	<b>GESTIÓN DE SERVICIOS ACADÉMICOS Y BIBLIOTECARIOS</b>		<b>CÓDIGO</b>	FO-GS-15	
			<b>VERSIÓN</b>	02	
	<b>ESQUEMA HOJA DE RESUMEN</b>			<b>FECHA</b>	03/04/2017
				<b>PÁGINA</b>	1 de 221
<b>ELABORÓ</b>		<b>REVISÓ</b>		<b>APROBÓ</b>	
Jefe División de Biblioteca		Equipo Operativo de Calidad		Líder de Calidad	

### RESUMEN TRABAJO DE GRADO

AUTOR(ES): NOMBRES Y APELLIDOS COMPLETOS

NOMBRE: ELIECER ALEJANDRO APELLIDOS: MOLINA VERGEL

FACULTAD: INGENERÍAS

PLAN DE ESTUDIOS: INGENERÍA DE SISTEMAS

DIRECTOR: MILTON JESÚS VERA CONTRERAS

TÍTULO DEL TRABAJO (TESIS): VPL++: PLATAFORMA WEB DE PRUEBAS UNITARIAS

AUTOMÁTICA EN MOODLE

VPL ++ ES UNA PLATAFORMA DE PRUEBAS UNITARIAS QUE EXTIENDE EL SISTEMA VIRTUAL PROGRAMMING LAB DE LA PLATAFORMA MOODLE. ESTE PROYECTO SE DESARROLLÓ EN 3 ETAPAS ITERATIVAS: 1) APROPIACIÓN Y ESTUDIO DE MOODLE Y VPL, 2) DESARROLLO Y 3) PRUEBAS. FINALMENTE SE DESARROLLÓ UNA PRUEBA PILOTO CON PROGRAMAS Y EJERCICIOS REALES QUE FUERON TOMADOS DE LA UVIRTUAL. VPL ++ ES CAPAZ DE LEER, EVALUAR Y GENERAR ESTADÍSTICAS DE VALOR QUE AYUDEN AL PROFESOR EN SU TAREA DE ENSEÑANZA DE LA PROGRAMACIÓN DE COMPUTADORES.

PALABRAS CLAVES: VPL, MOODLE, JUNIT, PRUEBAS UNITARIAS DE SOFTWARE

CARACTERÍSTICAS:

PÁGINAS: 221 PLANOS: \_\_\_ ILUSTRACIONES: \_\_\_ CD ROOM: \_\_\_

**VPL++: PLATAFORMA WEB DE PRUEBAS UNITARIAS  
AUTOMÁTICA EN MOODLE**

**ELIECER ALEJANDRO MOLINA VERGEL**

**UNIVERSIDAD FRANCISCO DE PAULA SANTANDER**

**FACULTAD DE INGENIERÍA**

**INGENIERÍA DE SISTEMAS**

**CÚCUTA**

**2020**

**VPL++: PLATAFORMA WEB DE PRUEBAS UNITARIAS  
AUTOMÁTICA EN MOODLE**

**ELIECER ALEJANDRO MOLINA VERGEL**

**PROYECTO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR POR EL  
TÍTULO DE INGENIERO DE SISTEMAS**

**DIRECTOR:**

**INGENIERO MILTON JESÚS VERA CONTRERAS**

**UNIVERSIDAD FRANCISCO DE PAULA SANTANDER**

**FACULTAD DE INGENIERÍA**

**INGENIERÍA DE SISTEMAS**

**CÚCUTA**

**2020**



**ACTA DE SUSTENTACIÓN DE TRABAJO DE GRADO**

**FECHA:** 24 de abril de 2020 **HORA:** 2:30 pm

**PLAN DE ESTUDIOS:** INGENIERÍA DE SISTEMAS

**TÍTULO DEL TRABAJO DE GRADO:**

**VPL++PLATAFORMA WEB DE PRUEBAS UNITARIAS AUTOMÁTICA EN MOODLE”**

**JURADOS:**

MSG. I.S. CLAUDIA YAMILE GÓMEZ LLAÑEZ

Ph.D. IS. MATÍAS HERRERA CÁCERES

MSc. I.S. JAIRO ALBERTO FUENTES CAMARGO

**DIRECTOR:** ING. MILTON JESÚS VERA CONTRERAS

<b>NOMBRE DEL ESTUDIANTE</b>	<b>CÓDIGO</b>	<b>CALIFICACIÓN</b>	<b>NÚMERO LETRA</b>
ELIECER ALEJANDRO MOLINA VERGEL	1151054	4,5	CUATRO, CINCO

**MERITORIA**

**FIRMA DE LOS JURADOS**

Phd. MATÍAS HERRERA CÁCERES

MSG. CLAUDIA YAMILE GÓMEZ LLAÑEZ

MSc. JAIRO ALBERTO FUENTES CAMARGO

Ph.D. JUDITH DEL PILAR RODRÍGUEZ TENJO  
Coordinadora Comité Curricular





Vigilada Mineducación



CARTA DE AUTORIZACIÓN DE LOS AUTORES PARA LA  
CONSULTA, LA REPRODUCCIÓN PARCIAL O TOTAL Y LA  
PUBLICACIÓN ELECTRÓNICA DEL TEXTO COMPLETO

Cúcuta,

Señores  
BIBLIOTECA EDUARDO COTE LAMUS

Ciudad

Cordial saludo:

Eliecer Alejandro Molina Vergel, identificado con la C.C. N° 1090446624, autor(es) de la tesis y/o trabajo de grado titulado: VPL++: PLATAFORMA WEB DE PRUEBAS UNITARIAS AUTOMÁTICA EN MOODLE, presentado y aprobado en el año 2020 como requisito para optar al título de Ingeniero de Sistemas, autorizo a la biblioteca de la Universidad Francisco de Paula Santander, Eduardo Cote Lamus, para que con fines académicos, muestre a la comunidad en general a la producción intelectual de esta institución educativa, a través de la visibilidad de su contenido de la siguiente manera:

- Los usuarios pueden consultar el contenido de este trabajo de grado en la página web de la Biblioteca Eduardo Cote Lamus y en las redes de información del país y el exterior, con las cuales tenga convenio la Universidad Francisco de Paula Santander.
- Permita la consulta, la reproducción, a los usuarios interesados en el contenido de este trabajo, para todos los usos que tengan finalidad académica, ya sea en formato CD-ROM o digital desde Internet, Intranet etc.; y en general para cualquier formato conocido o por conocer.

Lo anterior, de conformidad con lo establecido en el artículo 30 de la ley 1982 y el artículo 11 de la decisión andina 351 de 1993, que establece que “**los derechos morales del trabajo son propiedad de los autores**”, los cuales son irrenunciables, imprescriptibles, inembargables e inalienables.

ELIECER ALEJANDRO MOLINA VERGEL  
Cédula de ciudadanía: 1090446624

# Dedicatoria

Dedico este trabajo primero a Dios por ser quien inicia y culmina nuestros proyectos, a mi esposa por acompañarme y apoyarme durante el desarrollo de este proyecto y por último a mi familia quienes siempre estuvieron conmigo.

# Agradecimientos

Agradecimientos especiales a la Iglesia Casa Sobre la Roca en Cúcuta y al pastor en cabeza Enrique López Mayorga, quienes me ayudaron económicamente y moralmente a estudiar en la universidad.

A Amalia Valdez, quien me apoyó económicamente para continuar con mis estudios, aún sin conocerla desde Estados Unidos.

Al programa de Ingeniería de Sistemas por el esfuerzo y dedicación de los profesores, quienes permiten que los jóvenes que cursan y los que cursábamos permitan obtener una mejor calidad de vida.

Agradezco a la empresa Uruit por abrirme las puertas, darme buenas oportunidades y por el apoyo que recibí de todos en Medellín para terminar este trabajo.

Finalmente, agradezco mi director de tesis, el ingeniero Milton Jesús Vera Contreras, docente del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander por su paciencia y su guianza.

# Tabla de contenido

Introducción	20
1. Anteproyecto	22
1.1 Problema	22
1.2 Justificación	26
1.3 Objetivos	28
1.3.1 Objetivo General	28
1.3.2 Objetivos Específicos	28
1.4 Alcance	29
1.4.1 Alcance y delimitación funcional	29
1.4.2 Alcance y delimitación entregables	29
1.5 Metodología	31
1.5.1 Etapa 1: Apropiación de VPL y Moodle	31
1.5.2 Etapa 2: Desarrollo de software para extender VPL a VPL++	32
1.5.3 Etapa 3: Pruebas y validación	34
1.5.4 Etapa 4: Documentación y cierre	35
1.6 Cronograma	35
1.7 Presupuesto	38
1.7.1 Costos fijos	38
1.7.2 Costos mensuales	38
2. Marco Referencial	40
2.1 Antecedentes	40
2.1.1 Virtual Programming Lab en Uvirtual	40
2.2 Marco teórico	40
2.2.1 Arquitectura monolítica	40
2.2.2 Arquitectura orientada a microservicios	41

3. Virtual Programming Lab	43
3.1 Moodle	43
3.1.1 Arquitectura	43
3.2 VPL	44
3.2.1 Arquitectura	45
3.2.2 Instalación	47
3.3 Casos de uso	52
3.3.1 Profesor	52
3.3.2 Estudiante	62
3.4 Ventajas y limitantes	66
4. Pruebas unitarias en Java y librería para generación automática de pruebas	67
4.1 Generación automática de tests vs generación manual	68
4.2 Evosuite	69
4.3 Instalación	70
4.4 Uso	72
4.4.1 Crear clase ejemplo	72
4.4.2 Establecer classpath y generar pruebas unitarias a partir de una clase	73
4.5 Ejecutar pruebas unitarias generadas por Evosuite usando JUnit	75
4.6 Evosuite y VPL ++	76
5. Virtual Programming Lab ++	78
5.1 Metodología de desarrollo	79
5.2 Etapa 1: Apropiación de VPL y Moodle	79
5.2.1 Moodle y VPL Plugin	80
5.2.2 VPL Jaula de ejecución	80
5.2.3 Análisis de los componentes	82
5.2.4 Funcionamiento de la jaula de ejecución	83
5.2.5 Problema del alto acoplamiento en la idea inicial	92

5.2.6 Pensando “Out the box”	95
5.2.7 Conocimientos aprendidos	95
5.2.8 Alternativas de solución	96
5.2.9 Arquitectura orientada a microservicios	97
5.2.10 Extensión del proceso de ejecución de código en lugar de extender el código de VPL	102
5.2.11 Eligiendo la mejor arquitectura	104
5.2.12 Conclusiones de la etapa 1	105
5.3 Etapa 2: Desarrollo de software para extender VPL a VPL++	106
5.3.1 Identificación de actores	108
5.3.2 Identificación de componentes y piezas de software existentes	108
5.3.3 Definición de nuevos componentes y piezas de software	110
5.3.4 Tecnologías de desarrollo	114
5.3.5 Etapa de análisis	115
5.3.6 Etapa de desarrollo, test e integración	121
5.3.7 Estrategia de desarrollo	122
5.4 Funcionamiento de VPL ++	123
5.4.1 Terminología	123
5.4.2 Proyectos	125
5.4.3 Pruebas	129
5.4.4 Casos de prueba y tópicos	129
5.5 Ecuación matemática para medir el rendimiento de los estudiantes	130
5.6 Seguridad	133
5.7 Arquitectura	137
5.7.1 Orientada a Microservicios	137
5.7.2 S.O.A (Service Oriented Architecture)	138
5.7.3 Diagrama de Componentes del ecosistema	140
5.7.4 Diagrama de Secuencia del ecosistema	141
5.7.5 Jaula de ejecución y VPL ++ JLib Runner	142

5.7.6 VPL ++ API	151
5.8 Manual de usuario	157
5.8.1 Instalación y configuración	157
5.8.2 Iniciar sesión	157
5.8.3 Profesores	158
5.8.4 Administradores de Moodle	189
5.8.5 Recursos adicionales	194
5.8.6 F.A.Q	196
6. Prueba piloto	197
6.1 Selección de las actividades a comparar para el caso de estudio	197
6.2 Selección del estudiante para aplicar las pruebas con VPL ++ en el caso de estudio	204
6.2.1 Consideraciones	204
6.3 Ejecución de la prueba piloto	205
6.4 Resultados de la prueba piloto	207
6.4.1 Nivel de habilidad	207
6.4.2 Progreso del estudiante a través del tiempo	207
6.4.3 Dificultad	209
6.4.4 Tópicos	209
Conclusiones y recomendaciones	213
Recomendaciones	<b>¡Error! Marcador no definido.</b>
Anexos	216
Repositorios	216
Bibliografía	218

# Lista de tablas

Tabla 1 Costos fijos	38
Tabla 2 Costos mensuales	38
Tabla 3 Estrategias de modificación vs esfuerzo, acoplamiento y escalabilidad	94
Tabla 4 Conocimientos aprendidos	95
Tabla 5 Soluciones encontradas	96
Tabla 6 Microservicios y sus protocolos	97
Tabla 7 Estrategias de modificación con microservicios vs esfuerzo, acoplamiento y escalabilidad	105
Tabla 8 Ecuación matemática y sus variables para medir el nivel de habilidad de un estudiante	131
Tabla 9 Calificaciones promedio de las actividades	199
Tabla 10 Esfuerzo promedio	200
Tabla 11 Participación promedio	202
Tabla 12 Nivel de habilidad por mes del estudiante de la prueba piloto	208
Tabla 13 Repositorios	216



# Lista de ecuaciones

Ecuación 1 Casos no resueltos	131
Ecuación 2 Coeficiente negativo	132
Ecuación 3 Esfuerzo	132
Ecuación 4 Nivel de Habilidad	132

# Lista de figuras

Figura 1 Tasa de Aprobación/Reprobación Cursos Programación de Computadores IS-UFPS	22
Figura 2 Distribución de Calificaciones Cursos Programación de Computadores IS-UFPS 2 (Toward Unified Theory of Teaching and Learning Computer Programming : A Systematic Review of the Literature, 2017)	23
Figura 3 Arquitectura de aplicación monolítica tomado del libro Spring Microservices in Action p. 3	41
Figura 4 Arquitectura orientada a microservicios tomado del libro Spring Microservices in Action p. 4	42
Figura 5 Arquitectura común de Moodle	44
Figura 6 Diagrama de componentes Moodle y VPL	46
Figura 7 Diagrama de secuencia de ejecución de código en VPL	46
Figura 8 Vista administración de plugins en Moodle	48
Figura 9 Vista instalar plugin Moodle	48
Figura 10 Vista del Plugin Virtual Programming Lab en Moodle Plugins	49
Figura 11 Vista seleccionar instancia de Moodle en el directorio de plugins de Moodle	50
Figura 12 Vista confirmar instalación de VPL en Moodle	50
Figura 13 Vista confirmación de instalación exitosa de VPL	51
Figura 14 Vista actualizar base de datos de Moodle	51
Figura 15 Vista agregar ulr de la jaula de ejecución a VPL en Moodle	52
Figura 16 Vista activar edición en curso de Moodle	53
Figura 17 Vista añadir actividad o recurso a Moodle	53
Figura 18 Vista añadir actividad de VPL a curso en Moodle	54
Figura 19 Vista configurar nueva actividad de VPL en Moodle	55
Figura 20 Vista descripción de actividad de VPL en Moodle	55
Figura 21 Vista opciones de ejecución en actividad de VPL de Moodle	56
Figura 22 Vista de la configuración inicial de las opciones de ejecución de una actividad de VPL en Moodle	56
Figura 23 Vista actualización exitosa de las opciones de ejecución de VPL en Moodle	58
Figura 24 Vista seleccionar opción archivos de ejecución para configurar actividad de VPL en Moodle	59
Figura 25 Vista inicial del IDE de VPL	60

Figura 26 Vista modificar vpl_evaluate para activar las pruebas unitarias en las actividades de VPL en Moodle	60
Figura 27 Vista fraccionarioTest.java	62
Figura 28 Vista de la actividad de VPL desde el estudiante en Moodle	63
Figura 29 Vista agregar entrega a actividad de VPL en Moodle	64
Figura 30 Vista entrega guarda para actividad de VPL en Moodle	64
Figura 31 Vista evaluación en progreso de la respuesta de un estudiante en VPL en Moodle	65
Figura 32 Vista de la entrega del estudiante en el IDE de VPL en Moodle	65
Figura 33 Vista del estudiante de la opción editar entrega en la actividad de VPL en Moodle	66
Figura 34 Botones de guardar, correr y evaluar entrega en el IDE de VPL en Moodle	66
Figura 35 Ejecutar Evosuite	70
Figura 36 Evosuite comandos	71
Figura 37 Ejecutar Evosuite usando carpeta	73
Figura 38 Salida por consola al ejecutar Evosuite	74
Figura 39 test generado por Evosuite	75
Figura 40 Resultado de correr los tests generados or Evosuite	76
Figura 41 Diagrama de proceso de desarrollo de la etapa 1	81
Figura 42 Empaquetado de los archivos de ejecución que son enviados a la Jaula de Ejecución de VPL	84
Figura 43 listar archivos y carpetas dentro del sistema de archivos del proceso de ejecución de una actividad de VPL en Moodle	85
Figura 44 Contenido del archivo vpl_environment	86
Figura 45 Diagrama de secuencia para la etapa de conexión del IDE a la Jaula de Ejecución	87
Figura 46 salida por consola al correr pwd en la jaula de ejecución	88
Figura 47 Salida por consola al intentar graduar en etapa de compilación	89
Figura 48 Impresión de un comentario desde la Jaula de ejecución	90
Figura 49 Salida por consola de la etapa de compilación	91
Figura 50 Salida por consola de la etapa de ejecución	91
Figura 51 Diagrama de proceso de las etapas de ejecución de un programa en la Jaula de Ejecución	92
Figura 52 Respuesta de Juan Carlos a la solicitud de aportar al proyecto VPL	93

Figura 53 Diagrama de secuencia de la ejecución de una entrega de estudiante en VPL en Moodle	100
Figura 54 Diagrama de componentes de VPL con el sistema de archivos temporal	100
Figura 55 Diagrama de componentes de microservicios de VPL ++	102
Figura 56 Diagrama de secuencia de la ejecución de una actividad de VPL usando VPL ++	104
Figura 57 Diagrama de proceso de la etapa 2	107
Figura 58 Ejemplo de una prueba unitaria usando VPL ++	113
Figura 59 Relación de tablas para obtener el curso al cual está asignado una persona en Moodle	119
Figura 60 Distribución de la etapa de investigación, desarrollo y testing durante la etapa 2 del desarrollo del proyecto	121
Figura 61 Estrategia de desarrollo	122
Figura 62 Diagrama de proceso del ciclo de vida de la ejecución de un programa usando VPL ++	123
Figura 63 Proceso de nacimiento y ejecución de las pruebas de VPL ++	125
Figura 64 Diagrama de clases simple que explica la relación entre proyectos, pruebas y casos de pruebas de VPL++	126
Figura 65 JLib como librería y como software	127
Figura 66 Explicación de un test de java de VPL ++	128
Figura 67 Diagrama de clase simple para explicar la relación entre caso de prueba y tópico+	129
Figura 68 Seguridad de VPL ++	134
Figura 69 Relación entre políticas de acciones sobre recursos.	137
Figura 70 Diferencias entre SOA y Microservicios, tomado de Microservices: Flexible Software Architecture página 92	139
Figura 71 Distribución de los microservicios de VPL ++ por cada usuario	140
Figura 72 Diagrama de componentes de VPL ++	141
Figura 73 Diagrama de secuencia de VPL ++	142
Figura 74 Diagrama de proceso de la calificación automática de VPL ++	143
Figura 75 Casos de uso de VPL ++ JLib	143
Figura 76 Diagrama de secuencia del proceso de ejecución de pruebas unitarias de VPL ++ por JLib	144
Figura 77 Diagrama de clases general de JLib	145
Figura 78 Diagrama de clases de Test y TestCase	146

Figura 79 Diagrama de clases de una clase Parser	147
Figura 80 Diagrama de clase de VplReportSuite y VplReport	148
Figura 81 Diagrama de clases de VplRunner	149
Figura 82 Diagrama de clase de la clase ApiExporter y Printer	150
Figura 83 Diagrama de clases de un exporter de VPL ++	151
Figura 84 Caso de uso de JLib en el API	152
Figura 85 Diagrama de casos de uso del administrador de Moodle en el API	152
Figura 86 Diagrama de casos de uso del maestro en el API de VPL ++	153
Figura 87 Diagrama de uso del profesor de Moodle para ver sus estudiantes	153
Figura 88 Diagrama de casos de uso del profesor de Moodle para administrar sus proyectos	154
Figura 89 Diagrama de casos de uso del maestro sobre reportes	154
Figura 90 Arquitectura lógica de VPL ++ API	156
Figura 91 Distribución de las carpetas de VPL ++ API	156
Figura 92 Formulario inicio de sesión	158
Figura 93 Dashboard sin proyectos	159
Figura 94 Dashboard con proyectos	160
Figura 95 Menú al seleccionar proyecto que no se puede modificar	161
Figura 96 Advertencia al eliminar proyecto con respuestas de los estudiantes	162
Figura 97 Menú al seleccionar proyecto que si se puede modificar	162
Figura 98 Crear proyecto vista	163
Figura 99 Vista crear test	164
Figura 100 Vista test recién creado	164
Figura 101 Tarjeta de test	165
Figura 102 Test con su caso de prueba recién creado	166
Figura 103 Ventana para configurar caso de prueba	167
Figura 104 Caso de prueba - Información general	168
Figura 105 Editar código fuente del caso de prueba	168
Figura 106 Caso de prueba compilado	169

Figura 107 Pestaña para configurar la calificación y los tópicos del caso de prueba	169
Figura 108 Seleccionar topico	170
Figura 109 Configurar retroalimentación positiva	170
Figura 110 Configurar retroalimentación negativa	171
Figura 111 Vista final de configurar un caso de prueba	171
Figura 112 Seleccionar proyecto para exportar	172
Figura 113 Guardar archivo de restauración de actividad de Moodle generado por VPL++	173
Figura 114 Activar modo de edición de moodle	173
Figura 115 Menú restaurar en configuración de curso	174
Figura 116 Vista de archivo de restauración de actividad de Moodle cargado	175
Figura 117 Vista Seleccionar curso para restaurar la actividad de VPL ++	175
Figura 118 Vista que confirma que Moodle restauró la actividad normalmente	175
Figura 119 Actividad de VPL ++ restaurada satisfactoriamente	176
Figura 120 Pestaña actividad del proyecto	177
Figura 121 Dialogo de actividades de Moodle de VPL	178
Figura 122 Menú archivos de ejecución	179
Figura 123 Añadir ArrayList a la prueba unitaria de VPL ++	179
Figura 124 Tabla de estudiantes	180
Figura 125 Seleccionar estudiante	181
Figura 126 Seleccionar proyecto para ver reporte	182
Figura 127 Seleccionar estudiante para ver reporte	182
Figura 128 Vista de reportes si no hay reportes para mostrar	182
Figura 129 Botón de filtrar reportes	183
Figura 130 Cuadro de dialogo filtrar reportes	183
Figura 131 cuadro de dialogo filtrar reporte por fechas	184
Figura 132 Vista reportes	185
Figura 133 Pestaña Reporte de usuario	186
Figura 134 Información adicional del reporte de usuario	186

Figura 135 Pestaña progreso de los tópicos	187
Figura 136 Pestaña progreso de los tópicos expandida	187
Figura 137 Grafico progreso del estudiante	189
Figura 138 Administrador ventana inicial	190
Figura 139 Vista de administrar aplicaciones	191
Figura 140 Vista Administrar tópicos	192
Figura 141 Vista seleccionar tópico	193
Figura 142 Vista error al eliminar tópico	193
Figura 143 Grafico de calificaciones de Moodle promedio	200
Figura 144 Grafica de esfuerzo promedio	201
Figura 145 Grafico de participación promedio	202
Figura 146 Reporte de habilidad del estudiante tomado en la prueba piloto	207
Figura 147 Progreso del estudiante a través del tiempo	208
Figura 148 casos de prueba que resultaron más difíciles de resolver para el estudiante de la prueba piloto	209
Figura 149 Nivel de habilidad del estudiante de la prueba piloto por tópico	210
Figura 150 Leyendas del progreso del nivel de habilidad de los tópicos del estudiante de la prueba piloto	211
Figura 151 Gráfica del progreso del nivel de habilidad de los tópicos del estudiante de la prueba piloto	211

# Introducción

Desde hace cuatro décadas se han usado con éxito plataformas de pruebas automáticas de software en cursos de Programación de Computadores. Estas plataformas abordan tres aspectos: en primera medida disminuyen la carga de los profesores, pues la revisión de las actividades de los estudiantes requiere un gran esfuerzo. En segundo lugar, la retroalimentación personalizada y en tercer lugar el problema del plagio (Vera et al., 2018).

Una de dichas plataformas es Virtual Programming Lab (VPL), la cual comenzó a implementarse en el Programa de Ingeniería de Sistema de la Universidad Francisco de Paula Santander, en el primer semestre del año 2017, por iniciativa de los profesores de los cursos de programación de computadores de primer y segundo semestre. De dicha iniciativa surgió el presente proyecto, cuyo propósito es satisfacer requerimientos de lo profesores de la UFPS que no cubre la plataforma VPL original, en particular para los cursos de Programación Orientada a Objetos y reportes especializados del rendimiento de los estudiantes.

El proyecto estuvo centrado en los aspectos tecnológicos y de ingeniería. En paralelo, los profesores realizaron su práctica docente y de investigación aplicada, implementando estrategias y métodos pedagógicos y didácticos que no fueron del alcance de este proyecto. Principalmente, este proyecto se enfocó en adoptar la tecnología Cloud Computing, Docker y Microservicios para extender la funcionalidad de VPL, generando una segunda versión que se llamó VPL++. En conjunto, VPL ++ permite a los profesores formular ejercicios de Programación Orientada a Objetos en Java usando JUnit para que sean calificados automáticamente. Además, permite a los estudiantes aprovechar las funcionalidades de la versión original de VPL, pero dejando un registro detallado que le permite a VPL++ generar reportes especializados para una mejor retroalimentación del proceso educativo, tanto para estudiantes como profesores



A continuación, se presenta el informe final de este proyecto organizado en siete (7) capítulos: El capítulo 1 presenta el anteproyecto, omitiendo la sección de marco teórico y referencial, el cual aparece en el capítulo 2 y se detalla, según necesidad, a lo largo de los demás capítulos. El capítulo 3 trata lo referente a la plataforma VPL y Moodle. El capítulo 4 se refiere a las pruebas automáticas y su integración con VPL y Moodle. El capítulo 5 corresponde al producto final de este proyecto, la versión VPL++. El capítulo 6 detalla los resultados de la prueba piloto realizada en el grupo A de la asignatura Programación Orientada a Objetos I y finalmente se cierra con las conclusiones y recomendaciones.

Los resultados obtenidos en el proyecto, más allá de cumplir con los objetivos formulados, contribuyeron a una transformación en el Programa de Ingeniería de Sistema de la Universidad Francisco de Paula Santander, puesto que las actividades de formación y evaluación son consistentes con las prácticas de desarrollo de software en la vida real, la evaluación es mucho más objetiva e independiente del profesor, pues es automática y, además, se logra una mejor retroalimentación. Además, el proyecto mantiene una línea de innovación tecnológica en el sentido de adoptar Cloud Computing. Si bien no hay productos de divulgación científica en este proyecto, se tiene un producto de ingeniería que aportó a otros trabajos relacionados, bajo la dirección del mismo profesor. Aún queda mucho por seguir haciendo, por lo que se espera que este proyecto tenga continuidad en el tiempo y origine nuevos proyectos.

# 1. Anteproyecto

## 1.1 Problema

Aprender y enseñar programación de computadores exitosamente sigue siendo un problema difícil (Pea & Kurland, 1984; Weinberg, 1985; Kölling, 1999; Villalobos, Casallas, & Marcos, 2005; Villalobos & Calderón, 2009; Queirós & Leal, 2012; Scherer, 2016; Bosse & Gerosa, 2016; Kölling, 1999; Pea & Kurland, 1984; Queirós & Leal, 2012; ). En el caso particular del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (IS-UFPS), de acuerdo a las indagaciones preliminares de los profesores de los cursos de Fundamentos de Programación y Programación Orientada a Objetos, los resultados académicos indican que hay dificultades. La Gráfica 1 muestra el porcentaje de aprobación y reprobación de las cuatro (4) asignaturas básicas de Programación de Computadores durante el periodo 2012-I y 2016-I (Vera-Contreras & Herrera-Cáceres, 2017).

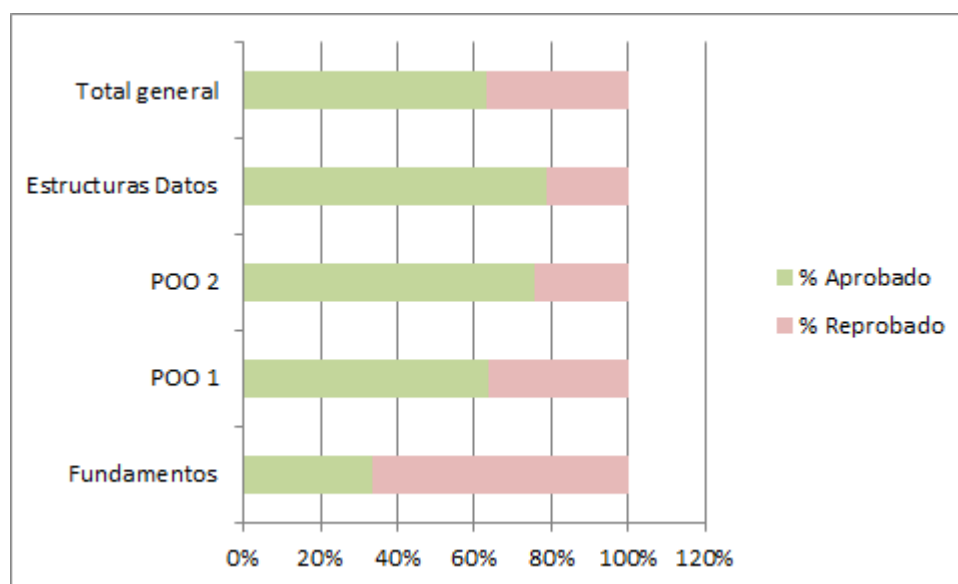


Figura 1 Tasa de Aprobación/Reprobación Cursos Programación de Computadores IS-UFPS

Según la gráfica, es evidente que la mayor dificultad está en el curso de primer semestre, fundamentos de Programación y se extiende a los cursos siguientes, lo cual se puede apreciar en la distribución de calificaciones que muestra la Gráfica 2.

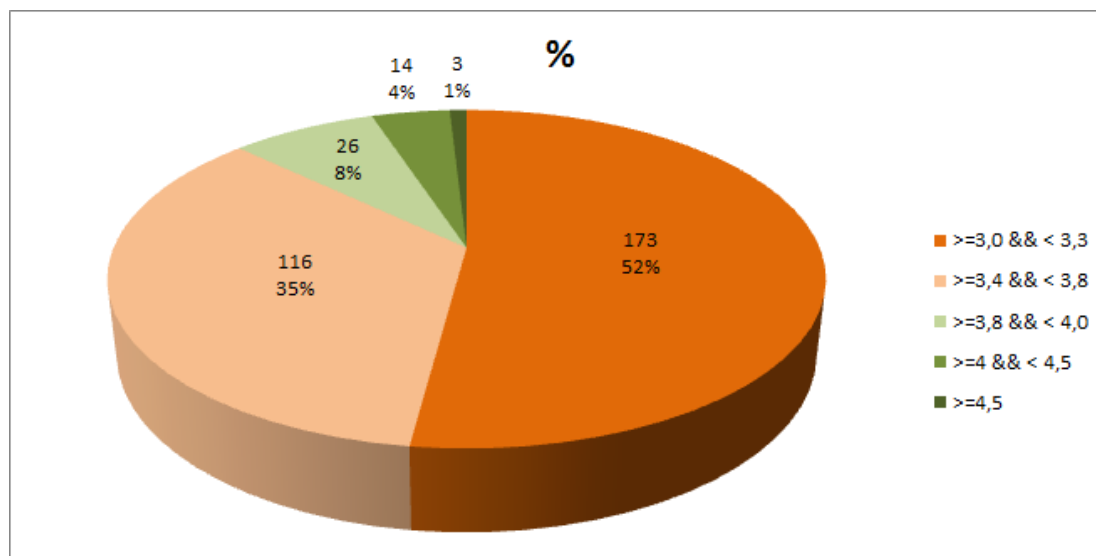


Figura 2. Distribución de Calificaciones Cursos Programación de Computadores IS-UFPS 2 (Toward Unified Theory of Teaching and Learning Computer Programming : A Systematic Review of the Literature, 2017)

De acuerdo a ésta información existen dos retos fundamentales: reducir la tasa de reprobación y mejorar las calificaciones, lo cual supone un mejoramiento en la calidad de enseñanza y aprendizaje de programación de computadores. Para enfrentar estos retos, la literatura ofrece diversas opciones, las cuales se detallan en el estado del arte de éste documento y a continuación se mencionan las más destacadas (Milton-Jesús Vera-Contreras & Herrera-Cáceres, 2017a):

Enfocarse en aspectos de alto nivel de Ingeniería de Software, como arquitecturas y metodologías con enfoque basado en problemas (J. A. Villalobos & Calderón, 2009) .

Usar lenguajes y entornos de desarrollo apropiados, IDE por su nombre en inglés Integrated Development Environment. (Patterson et al., 2003)

Enfocarse en los fundamentos matemáticos (Universidad Nacional de Colombia, Plataforma Inteligente De Aprendizaje Virtual)

Enseñar primero solución de problemas “Problem Solving” (Chu, 2009; Papaspyrou & Zachos, 2013).

Abordar las supuestas debilidades de los estudiantes en comprensión de lectura y razonamiento (Ariza, 2014; Rosales Sales, Euline esperanza; Suarez García, German Darío; Velazco Sanchez, n.d.), postura que es contraria a algunos planteamientos de la Psicología de la programación e inconsistente con los puntajes de pruebas ICFES / SABER 11 que sirven como filtro de admisión a la carrera.(Scherer, 2016)

Utilizar herramientas de visualización (Alhammad et al., 2016; J. A. Villalobos & Calderón, 2009).

Usar herramientas para evaluación automática (Carlos & Royo, n.d.; Douce et al., 2005; Fraser & Arcuri, 2011; Queirós & Leal, 2012) para: (i) minimizar la sobrecarga de los profesores, (ii) mejorar la motivación y reducir la ansiedad de los estudiantes y (iii) desarrollar integralmente competencias blandas y duras para programación de computadores.

En el presente trabajo se propone dar continuidad a la última opción: (g), herramientas para evaluación automática. Se busca aprovechar el Proyecto VPL Moodle de la Universidad de Las Palmas de Gran Canaria (Carlos & Royo, n.d.; Edith Lovos & Inés, n.d.; Rodríguez-del-Pino et al., 2012), considerando que, durante el primer semestre del año 2017 se experimentó con éste proyecto en el grupo de estudiantes de Fundamentos de Programación de IS-UFPS con buenos resultados(Milton-Jesús Vera-Contreras & Herrera-Cáceres, 2017b). Además, durante el segundo semestre de 2016 y primer semestre de 2017 se experimentó con el uso de Pruebas Unitarias

Automáticas en Java usando JUnit en el curso de POO I de IS-UFPS, también con buenos resultados (Milton-Jesús Vera-Contreras & Herrera-Cáceres, 2017b)

En consecuencia, se propone extender VPL para incorporar Pruebas Unitarias Automáticas usando JUnit, a fin de proporcionar a profesores y estudiantes de una herramienta que contribuya a enfrentar los desafíos de enseñanza y aprendizaje de programación de computadores. Los aspectos de investigación educativa sobre el uso y efectos de éste tipo de estrategias y herramientas no son del alcance del presente trabajo.

De acuerdo a lo expuesto las siguientes preguntas sintetizan el problema:

¿Cómo potencializar la plataforma VPL y las pruebas unitarias automáticas para enfrentar los retos de aprendizaje y enseñanza de programación de computadores en el programa de IS-UFPS?

¿Cómo integrar VPL con los cursos virtuales de la Uvirtual del Programa de Ingeniería de Sistemas de la UFPS,

¿Cómo incorporar Pruebas Unitarias Automáticas de Java tipo JUnit en VPL?

¿Cómo agilizar el diseño de ejercicios de programación de computadores con Pruebas Unitarias Automáticas en VPL?

## 1.2 Justificación

El Programa de Ingeniería de Sistemas es un programa acreditado de Alta Calidad. En ese sentido, como parte de la mejora continua, es de gran importancia abordar problemas académicos como la reprobación y nivel de calificación de los cursos de programación de computadores. Lo que propone éste proyecto es aprovechar una de las tendencias existentes como lo es el uso de evaluaciones automáticas, según el proyecto VPL y fusionarla con pruebas unitarias automáticas, una técnica de la industria que en contextos educativos genera buenos resultados.

La literatura indica que el entrenamiento, como una forma del aprendizaje activo, es una actividad fundamental en el aprendizaje de la programación de computadores (J. A. Villalobos & Calderón, 2009). No obstante, el entrenamiento demanda una alta carga docente en cuanto a preparación de ejercicios y retroalimentación al estudiante (Carlos & Royo, n.d.; Douce et al., 2005; Fraser & Arcuri, 2011; Queirós & Leal, 2012). En ese sentido, éste trabajo permitirá disminuir la carga laboral del docente en cuanto a la evaluación de los programas presentados por los estudiantes. Normalmente un curso de programación tiene más de treinta (30) estudiantes, si se estiman por curso dos previos, un examen final y cinco tareas, un profesor debe evaluar doscientos cuarenta (240) programas. Esta carga académica no permite que el profesor pueda focalizar sus esfuerzos en los problemas particulares del curso que dirige, pues cada curso presentará dificultades diferentes, ya sea la resolución de problemas, la comprensión del lenguaje de programación, o cualquier otra dificultad.

Al usar una plataforma de evaluación automática, ésta evaluaría según los criterios establecidos por el docente, con lo cual él conseguirá evaluar con mayor agilidad y objetividad. Además, el profesor podrá enfocar sus esfuerzos en reforzar áreas específicas de sus estudiantes y enfocar las clases con mayor aserción sin alejarse de su objetivo: mejorar el proceso de enseñanza y aprendizaje de la programación de computadores, en lugar de seguir un currículo lineal sin tener en cuenta los diferentes factores que dificultan la enseñanza de la misma. Por otra parte, el estudiante se beneficiará al tener retroalimentación de manera más inmediata,

aprovechando el tiempo presencial con el profesor, para clarificar y mejorar su proceso educativo.

El programa de Ingeniería de Sistemas podría mejorar su visibilidad al ser pionera a nivel regional y nacional en implementar una plataforma de éste tipo.

## 1.3 Objetivos

### 1.3.1 Objetivo general

Desarrollar una extensión de la Plataforma Virtual Programming Lab (VPL) para el uso de Pruebas Unitarias Automáticas con fines educativos VPL++.

### 1.3.2 Objetivos específicos

Implementar en ambiente de pruebas la Plataforma VPL en un servidor de manera integrada con la plataforma Uvirtual del Programa de Ingeniería de Sistemas de la UFPS.

Extender la Plataforma VPL para incorporar Pruebas Unitarias Automáticas en Java.

Incorporar a la Plataforma VPL una librería para la generación automática de Pruebas Unitarias en Lenguaje Java con JUnit.

Documentar una prueba piloto de VPL++ en los cursos de Fundamentos de Programación y POO I del Programa de Ingeniería de Sistemas de la UFPS.



## 1.4 Alcance

### 1.4.1 Alcance y delimitación funcional

Se entregará un plugin para Moodle en donde:

El estudiante puede resolver problemas de programación propuestos por su profesor, los cuales se evalúan de manera automática.

1. El profesor puede publicar ejercicios propuestos de programación para que los estudiantes los resuelvan y entreguen a través de la plataforma, indicando los detalles de objetivos e indicaciones.
2. El profesor puede definir pruebas unitarias en Java JUnit para los ejercicios propuestos, de acuerdo a los objetivos e indicaciones dadas al estudiante.
3. El profesor puede listar los resultados de los ejercicios entregados por los estudiantes y dar retroalimentación en línea.

### 1.4.2 Alcance y delimitación entregables

1. Software VPL ++
2. Manual de usuario para los estudiantes y profesores.
3. Documento de Arquitectura del Sistema
4. Código fuente documentado.

Por último, se entregará el documento de Trabajo de Grado optando por el título de Ingeniero de Sistemas, el cual contendrá todos los detalles del proyecto.

El desarrollo de la plataforma está fijado a los estudiantes y docentes de los primeros cursos de programación en el programa de Ingeniería de Sistemas en la Universidad Francisco de Paula Santander sede Cúcuta / Norte de Santander – Colombia

## 1.5 Metodología

Existen dos paradigmas en la investigación aplicable a la ingeniería: ciencia orientada al comportamiento y ciencia orientada al diseño (Esearch et al., 2004a). El primero busca verificar las teorías que explican o predicen el comportamiento humano individual o dentro de una organización, su objetivo es la verdad. La segunda busca extender las fronteras de las capacidades humanas y organizacionales creando e innovando artefactos, su objetivo es la utilidad. Éste proyecto se ubica en el segundo enfoque, pues no se desea probar ninguna teoría, en su lugar se quiere extender un artefacto existente, extender VPL a VPL++. La extensión permitirá apoyar la solución de un problema específico, para el caso la educación en programación de computadores.

Si bien el proyecto implica ciertas tareas de desarrollo de software, también implica otras tareas que no caen dentro de una metodología específica de desarrollo de software. Por lo tanto, metodológicamente se estructura el proyecto en cuatro etapas, las cuales determinan claramente el proceso a seguir y sus entregables:

### 1.5.1 Etapa 1: Apropiación de VPL y Moodle

En esta etapa se creará una máquina virtual donde se instalará como sistema operativo CentOS 7 en su versión minimalista. Después se instalarán el servidor web Apache, PHP como cgi y MariaDb como servidor de base de datos. Para administrar cómodamente el servidor de base de datos se instalará además phpMyAdmin. Luego, se descargará Moodle vía git:

Moodle: ([https://docs.moodle.org/33/en/Git\\_for\\_Administrators](https://docs.moodle.org/33/en/Git_for_Administrators))

VPL: ([https://github.com/jcrodriguez-dis/moodle-mod\\_vpl](https://github.com/jcrodriguez-dis/moodle-mod_vpl)).

Esta etapa nos ayudará a conocer los requerimientos para la instalación y configuración de Moodle junto a VPL, además se dejará una máquina virtual funcional con los servicios disponibles para ejecutar Moodle junto a VPL, lista para ser usado por los estudiantes de los cursos de Introducción a la programación y Programación Orientada a Objetos I en la Universidad Francisco de Paula Santander.

## 1.5.2 Etapa 2: Desarrollo de software para extender VPL a VPL++

No se sigue una metodología de desarrollo de software específica por tres razones fundamentales: (i) no hay un equipo de trabajo grande que lo justifique, sólo un patrocinador (el director del proyecto como sponsor) y un desarrollador (el estudiante). (ii) No es un proyecto de desarrollo nuevo sino un mantenimiento, personalización o extensión de un software existente. (iii) Este proyecto es un proceso creativo y cualquier metodología restringe la creatividad pues está pensada para entornos industriales. En consecuencia, se usará un Proceso Personal de Software PSP siguiendo un enfoque iterativo e incremental:

- a. Definición de los requerimientos funcionales y no funcionales:
- b. Indagación de las necesidades de software de los actores:  
profesores de y estudiantes de los diferentes cursos de programación de UFPS-IS.
- c. Análisis
- d. Estudiar la documentación de Moodle disponible en la web principal del proyecto [https://docs.moodle.org/33/en/Main\\_page](https://docs.moodle.org/33/en/Main_page), en ella se encuentra la documentación para el usuario final, usuario administrador y para el desarrollador.

e. Estudiar la documentación de VPL disponible en la web principal del proyecto: <http://vpl.dis.ulpgc.es/index.php/support>, además se cuenta además con las diferentes publicaciones que ha dejado el proyecto <http://vpl.dis.ulpgc.es/index.php/home/related-publications>.

f. Estudiar la arquitectura de VPL y los componentes que lo conforman (navegador, Moodle, y Jail Execution System) (Carlos & Royo, n.d.)

g. Diseño:

h. Diseño de librería en JAVA que usará JUnit, ésta será usada por VPL ++ y generará las calificaciones basadas en objetivos

i. Diseño de los diferentes diagramas UML (componentes, clases, etc) necesarios para la comprensión del sistema.

j. Diseño de interfaces gráficas.

#### 1. Programación:

a. Realizar pequeñas modificaciones a VPL como labels, posición de botones, colores, entre otros. de esta manera entender los componentes de software que intervienen en Moodle-VPL. En este paso se debe conocer grosso modo el funcionamiento de Moodle-VPL y el flujo de la información, desde el navegador, Moodle y la Jaula de ejecución de VPL y su retorno. Este punto dará soporte al objetivo específico número 2, ya que es necesario conocer el funcionamiento de VPL y su acoplamiento con Moodle. Esto es sumamente importante para ahorrar esfuerzos y tiempo al identificar los componentes, actores y eventos que intervienen en las funciones que ofrece la plataforma; y por ultimo identificar las clases o porciones de código que deberemos modificar para implementar las funcionalidades descritas en el segundo objetivo.

b. Codificación: después de identificar las clases, patrones de diseño, porciones de código o componentes lógicos en VPL procederemos a la codificación para extender la plataforma e implementar pruebas unitarias en

lenguaje Java y la integración de una librería para generar pruebas automáticas de software.

2. Pruebas:

a. Validación de requerimientos funcionales y no funcionales.

b. Pruebas de software: luego de la codificación se procede a hacer las pruebas que determinen el correcto funcionamiento de nuestras funcionalidades y su acoplamiento con el resto de la plataforma.

3. Iteración: Puesto que el desarrollo de software ya no se concibe como un proceso lineal y determinado sino iterativo y guiado por el usuario y el desarrollador, es necesario realizar varias iteraciones (iterando desde el numeral 4), las cuales se documentarán según el avance del proyecto.

4. Finalmente se creará dentro de VPL++ páginas de ayuda para docentes y estudiantes, explicando las funciones del sistema.

5. Aplicar un análisis automático de métricas de calidad al software final.

### 1.5.3 Etapa 3: Pruebas y validación

Para determinar el correcto funcionamiento de la plataforma VPL++ se desplegará en un servidor que será probado por los estudiantes y profesores de Fundamentos de Programación y Programación Orientada a Objetos I. Esta etapa se desarrollará en las siguientes sub-etapas:

En esta etapa se trabajará con los profesores para que:

a. El profesor publique en la plataforma varios ejercicios de programación para ser resueltos y sus pruebas unitarias correspondientes para cada ejercicio, o generarlas automáticamente.

b. El estudiante por su parte se encargará de resolverlos usando la plataforma. La plataforma por otro lado se encargará de ejecutar las pruebas unitarias que el profesor dispuso para el ejercicio







Fase	N°	Obj	Entregables	Actividades	Mes 1				Mes 2				Mes 3				Mes 4					
					1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4		
Desarrollo de Software	2	2	VPL ++	Desarrollo de software					x	x	x	x										
				Pruebas de software									x	x								
Pruebas y validación	3	3	VPL++ instalado en la Uvirtual	Instalación de VPL ++ en el Moodle Uvirtual de la universidad											x							
			.Ejercicios guardados en VPL ++ por los profesores y sus resultados (por los estudiantes)	Publicación de ejercicios de programación en los primeros cursos de programación (Fundamentos de programación y POO 1 )												x						
			Encuestas aplicadas	Creación de encuesta													x					
			tabuladas y analizadas	Aplicación de encuesta														x				
				Tabulación y análisis														x	x			
Documentación y cierre	4	4	Documentación	Redacción de la documentación															x	x		
				Redacción de los manuales de uso.																x		
				Redacción del artículo de investigación																x	x	

## 1.7 Presupuesto

### 1.7.1 Costos fijos

Tabla 1 Costos fijos

Activo	Descripción	Costo
Computador	Intel core i7 6500 8 Gb RAM Nvidia Geforce 920mx 1 Tb Hd	\$ 2'400.000,00
Mouse	Mouse óptico USB	\$ 25.000
Impresora	Impresora multifuncional Láser Samsung SL-M2070FW Samsung,	\$ 525.900
Papelería	Insumos necesarios para generar la documentación.	\$ 100.000
<b>Total:</b>		\$ 3'050.900,00

### 1.7.2 Costos mensuales:

Tabla 2 Costos mensuales

Activo	Descripción	Costo Mensual
Desarrollador de Software junior	Necesario para la instalación, configuración y despliegue del entorno de desarrollo.  Necesario para el desarrollo de VPL++  Necesario para la documentación del proyecto.	\$ 1'800.000,00

Electricidad	Gasto mensual de energía eléctrica	\$ 30.000,00
Agua	Gasto mensual	\$ 25.000,00
Internet	5 Mb banda ancha	\$ 99.800,00
Arriendo	Estación de trabajo	\$ 600.000,00
<b>Total:</b>		\$ 2'554.800,00 por Mes

## 2. Marco Referencial

### 2.1 Antecedentes

La Universidad Francisco de Paula Santander da a disposición de los estudiantes las siguientes plataformas.

#### 2.1.1 Virtual Programming Lab en Uvirtual

El programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander ofrece diferentes cursos como apoyo para las materias del programa. Estas usan el plugin V.P.L (Virtual Programming Lab), éste permite al maestro crear actividades de programación en los cursos, y los estudiantes resolver dichas actividades, esto se explicará con más profundidad más adelante.

### 2.2 Marco teórico

En esta sección definiremos algunos conceptos que se usarán durante el desarrollo de éste documento.

#### 2.2.1 Arquitectura monolítica

La arquitectura de una aplicación monolítica se refiere a una aplicación con una única base de código / repositorio grande que ofrece decenas o cientos de servicios utilizando diferentes interfaces como páginas HTML, servicios web y / y servicios REST (Villamizar et al., 2016).

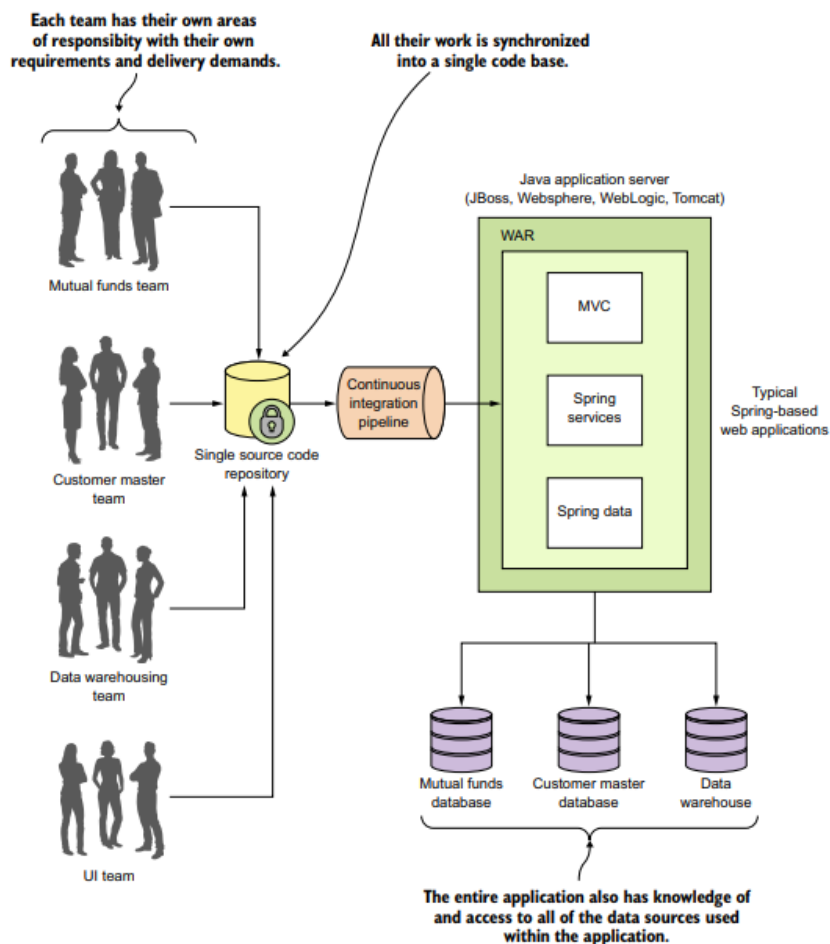


Figura 3 Arquitectura de aplicación monolítica tomado del libro *Spring Microservices in Action* p. 3

## 2.2.2 Arquitectura orientada a microservicios

El concepto de un microservicio se introdujo originalmente la comunidad de desarrollo de software alrededor de 2014 y fue una respuesta directa a muchos de los desafíos de tratar de escalar monolíticos, tanto técnicos como organizativos (Cosmina, 2017).

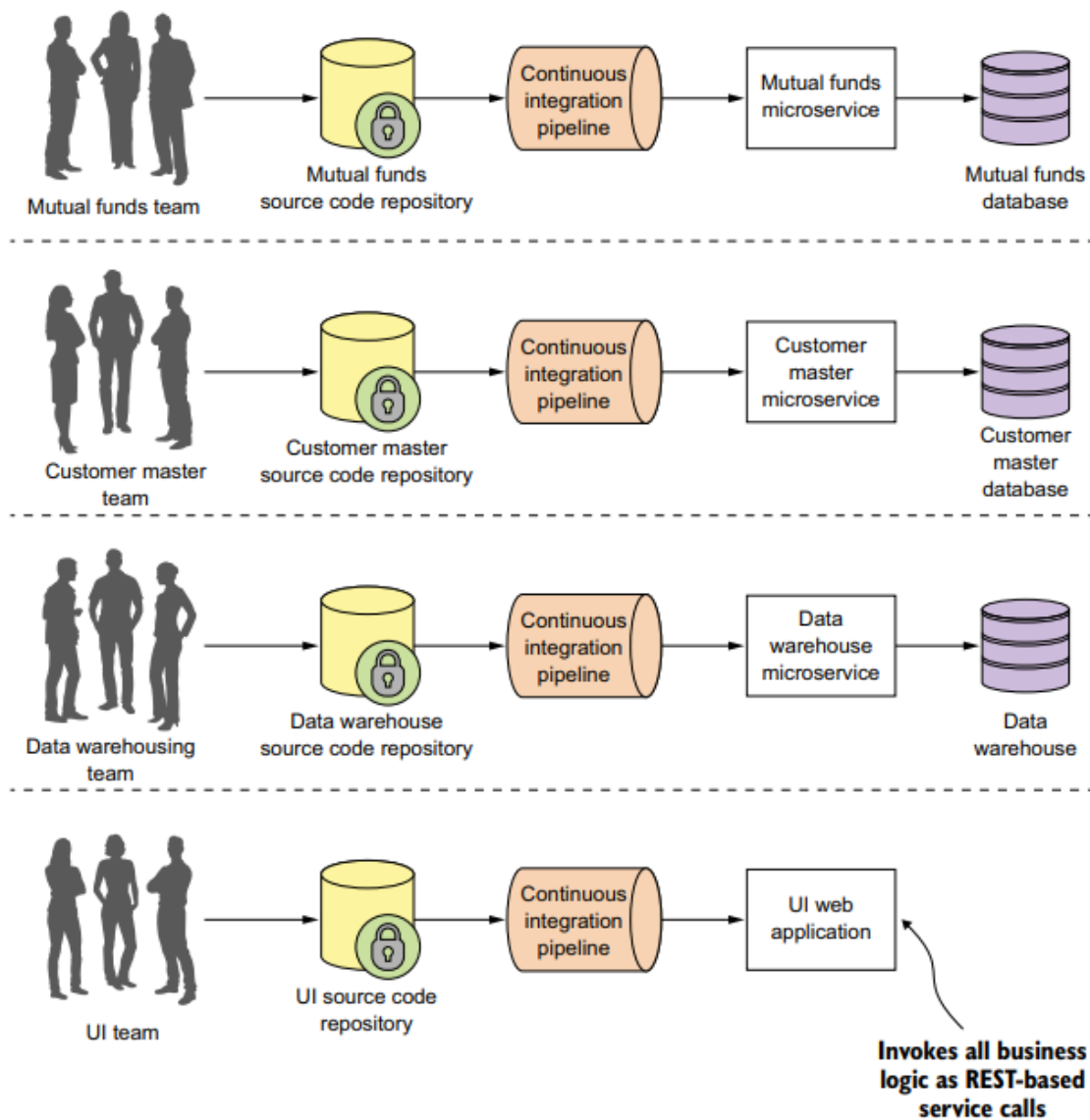


Figura 4 Arquitectura orientada a microservicios tomado del libro *Spring Microservices in Action* p. 4

## 3. Virtual Programming Lab

### 3.1 Moodle

Moodle es un sistema de gestión de aprendizaje, un sistema de gestión de cursos o un entorno de aprendizaje virtual, según el término que prefiera. Su objetivo es brindar a los profesores y estudiantes las herramientas que necesitan para enseñar y aprender. Moodle proviene de un contexto de pedagogía social constructivista, sin embargo, se puede utilizar para apoyar cualquier estilo de enseñanza y aprendizaje.

Más detalles sobre Moodle se pueden consultar desde acá:

[https://docs.Moodle.org/dev/Moodle\\_architecture#What\\_is\\_Moodle.3F](https://docs.Moodle.org/dev/Moodle_architecture#What_is_Moodle.3F)

Los detalles de la instalación y configuración pueden encontrarse en su web oficial:

<https://docs.Moodle.org/>

#### 3.1.1 Arquitectura

Moodle está escrito en PHP y es servido al cliente usando Apache como servidor web. La base de datos es Mysql y su estructura es creada por medio de migraciones de Moodle. Por otro lado,

Moodle se instala y almacena sus plugins en el sistema de archivos donde se instala.

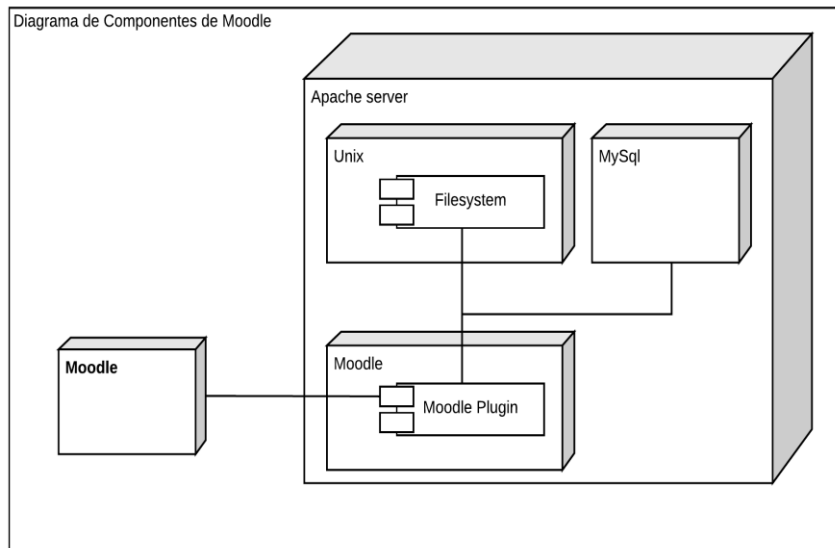


Figura 5 Arquitectura común de Moodle

## 3.2 VPL

V.P.L es la sigla de Virtual Programming Lab. Es un plugin para el LMS (Learning Management System) Moodle, está diseñado para evaluar de manera automática ejercicios en cursos de programación de computadores (Andrés et al., n.d.).

Características:

1. Crear y modificar código fuente dentro del navegador.
2. Ejecutar programas interactivamente en el navegador.
3. Estudiantes y profesores pueden ejecutar casos de prueba.
4. Los profesores pueden buscar código fuente similar dentro de la plataforma.
5. Compilación, depuración y ejecución de programas en múltiples lenguajes de programación
6. Control de ejecución de los programas



7. Control de envíos:
  - a. Límite de envíos por tareas.
  - b. Número máximo de archivos subidos y el tamaño máximo de cada uno de ellos.
8. Control de plagio

Actualmente es usado en los primeros cursos de programación de IS-UFPS como herramienta de estudio y evaluación, el profesor publica sus ejercicios mientras que los estudiantes los resuelve en el salón de clase o en su casa, el profesor lo estipula. Hasta ahora ha tenido buena aceptación entre estudiantes y profesores, sin embargo, el software es bastante transaccional, esto quiere decir que los programas de los estudiantes solo generan salidas y los profesores deben evaluar por aparte otras métricas y aspectos que son importantes cuando un estudiante aprende programación y pensamiento abstracto.

### 3.2.1 Arquitectura

Como software, VPL consta de dos componentes que funcionan en conjunto: el plugin de Moodle y la jaula de ejecución. La Jaula de ejecución es una instancia aparte, es decir, otro computador, servidor, máquina virtual o imagen virtualizada diferente a Moodle. En ella se instala el software de la jaula de ejecución de VPL, éste software es un servidor XMLRPC, que ejecuta procedimientos remotamente. De esta manera puede contenerse la ejecución de código malicioso sin afectar a Moodle, es decir, se recomienda instalar la jaula de ejecución en una instancia diferente a la instancia de Moodle.

#### *Diagrama de componentes*

En el siguiente diagrama se comprende mejor la distribución de Moodle, el plugin y la jaula de ejecución

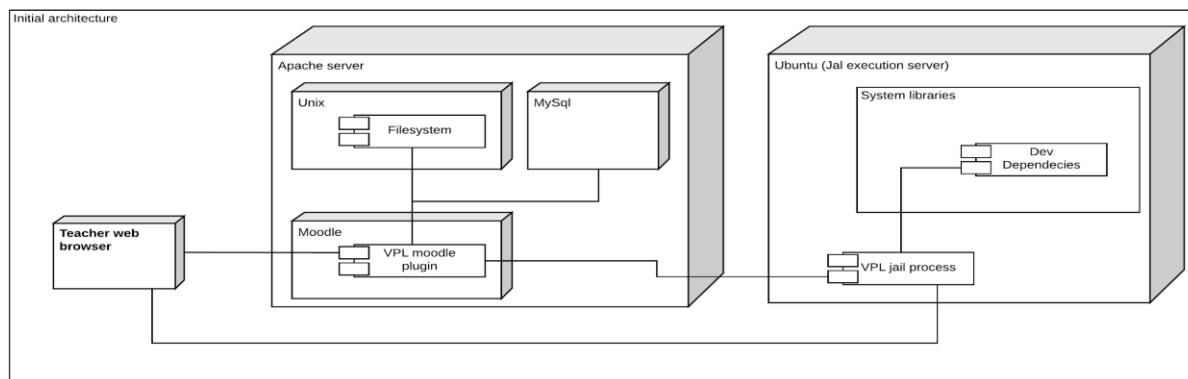


Figura 6 Diagrama de componentes Moodle y VPL

### Diagrama de secuencia

El siguiente diagrama muestra el proceso de ejecución del código de un estudiante, aplica igual para el profesor que ejecuta pruebas de su actividad.

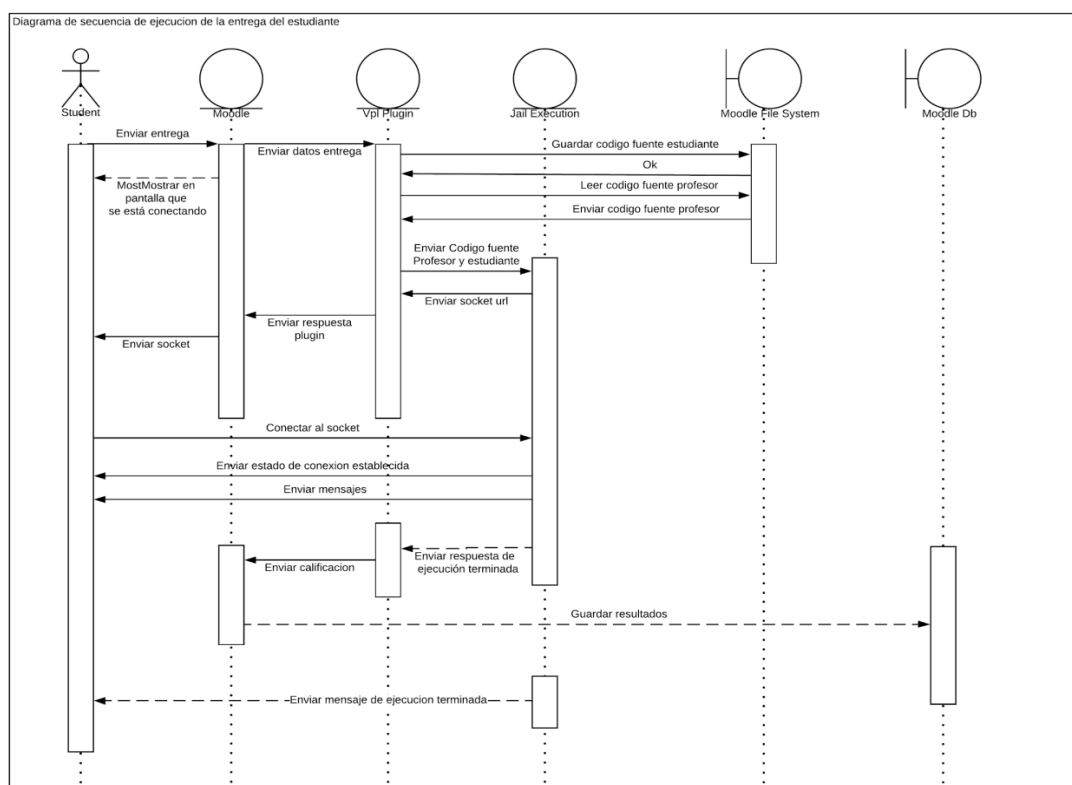


Figura 7 Diagrama de secuencia de ejecución de código en VPL

## 3.2.2 Instalación

### *Instalación de la jaula de ejecución de VPL*

Nota 1: La jaula de ejecución debe ser un componente aislado pues ejecutará, por lo que se recomienda usar una instancia (virtual o física).

Nota 2: se debe instalar sobre Ubuntu 16 según la documentación lo informa. En otros sistemas operativos no funciona correctamente.

Nota 3: Debe prestar atención al paso cuatro, ya que se le preguntará que software adicional desea instalar. Si usted quiere ejecutar pruebas unitarias con JUnit, deberá prestar atención, ya que se le preguntará si desea instalarlo

### *Instalación y configuración de VPL en Moodle*

Requerimientos:

1. Instancia registrada de Moodle en Moodle.net
2. Ser administrador en Moodle o tener permisos de administración (solo para instalar VPL usando el directorio de Moodle)

Nota: Según la versión y lenguaje configurado en Moodle, las imágenes podrían cambiar un poco. Consulte la documentación para su versión de Moodle.

Vaya al menú de administración de plugins en la ventana de administración del sitio

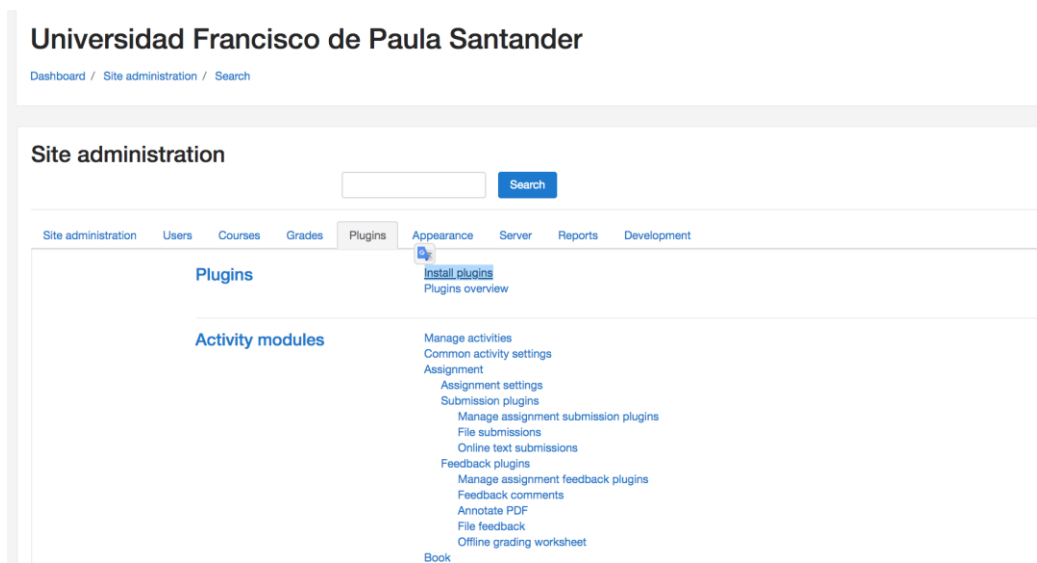


Figura 8 Vista administración de plugins en Moodle

Le mostrará la siguiente ventana:

## Plugin installer

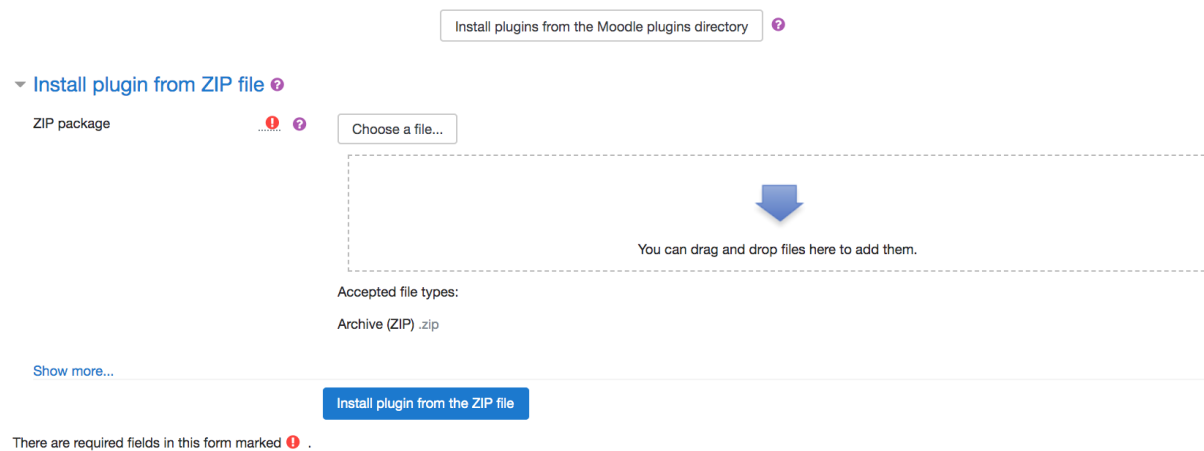


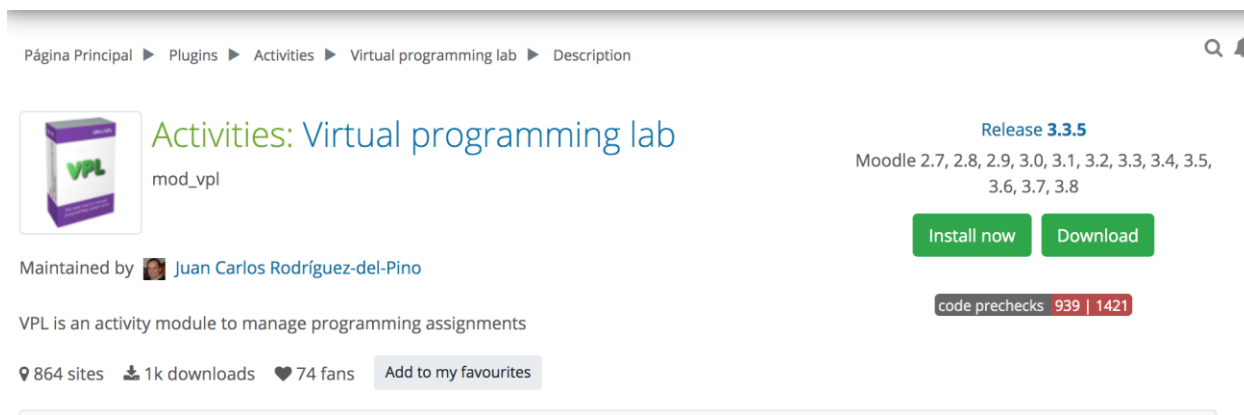
Figura 9 Vista instalar plugin Moodle

Si usted no está registrado en Moodle.net, entre al siguiente link:

[https://Moodle.org/plugins/view.PHP?plugin=mod\\_VPL](https://Moodle.org/plugins/view.PHP?plugin=mod_VPL)

Y de clic en “download”. Luego en la ventana de instalación del plugin, haga clic en el botón “Install plugin from zip file”

Además, podrá instalar el plugin desde el directorio de Moodle. Si su sitio de Moodle está registrado en Moodle.net haga clic en el botón superior, en el navegador le redirigirá al directorio de plugins de Moodle. Busque el plugin “Virtual Programming Lab” y siga el link. Le mostrará la siguiente ventana:



The screenshot shows the Moodle Plugins page for 'Activities: Virtual programming lab'. The breadcrumb trail is 'Página Principal > Plugins > Activities > Virtual programming lab > Description'. The plugin icon is a purple box with 'VPL' on it. The title is 'Activities: Virtual programming lab' with the code 'mod\_vpl' below it. It is maintained by 'Juan Carlos Rodríguez-del-Pino'. The description is 'VPL is an activity module to manage programming assignments'. There are 864 sites, 1k downloads, and 74 fans. A button 'Add to my favourites' is present. The release version is '3.3.5' and it is compatible with Moodle versions 2.7, 2.8, 2.9, 3.0, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, and 3.8. There are two buttons: 'Install now' and 'Download'. A badge shows 'code prechecks 939 | 1421'.

Figura 10 Vista del Plugin Virtual Programming Lab en Moodle Plugins

Dele clic en el botón “Install now” y seleccione la instancia donde se encuentra su instancia de Moodle

## My sites

Sites entered here are used when choosing to directly install plugins to your site. Moodle 2.5 or higher is required.

Virtual programming lab	Site name	Version	Site URL	+
<a href="#">Install now</a>	Universidad Francisco de Paula Santander	3.4	http://34.95.137.32:8080	 
<a href="#">Install now</a>	Universidad Francisco de Paula Santander	3.4	https://35.198.14.0	 
<a href="#">Install now</a>	Universidad Francisco de Paula Santander	3.4	http://localhost:8080	 
<a href="#">Install now</a>	Universidad Francisco de Paula Santander	3.4	http://34.95.252.116:8080	 

Figura 11 Vista seleccionar instancia de Moodle en el directorio de plugins de Moodle

Dele clic en “Install now” en la instancia de Moodle donde desea instalar VPL. Esto lo redirigirá a su instancia de Moodle y le solicitará aceptar los términos y condiciones. Deberá aceptar.

### Install plugins from the Moodle plugins directory

**Confirm**

There is a request to install plugin **Virtual programming lab** (mod\_vpl) version 2019102116 from the Moodle plugins directory on this site. If you continue, the plugin ZIP package will be downloaded for validation. Nothing will be installed yet.

Figura 12 Vista confirmar instalación de VPL en Moodle

Se empezará a instalar el plugin. Al finalizar haga clic en continuar

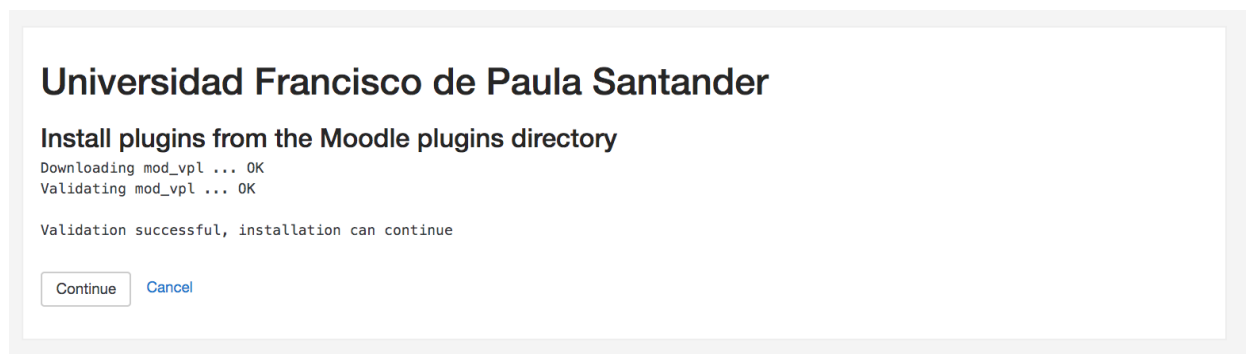


Figura 13 Vista confirmación de instalación exitosa de VPL

Moodle solicitará su permiso para actualizar la estructura de la base de datos. Por favor haga clic en el botón “upgrade database now”.

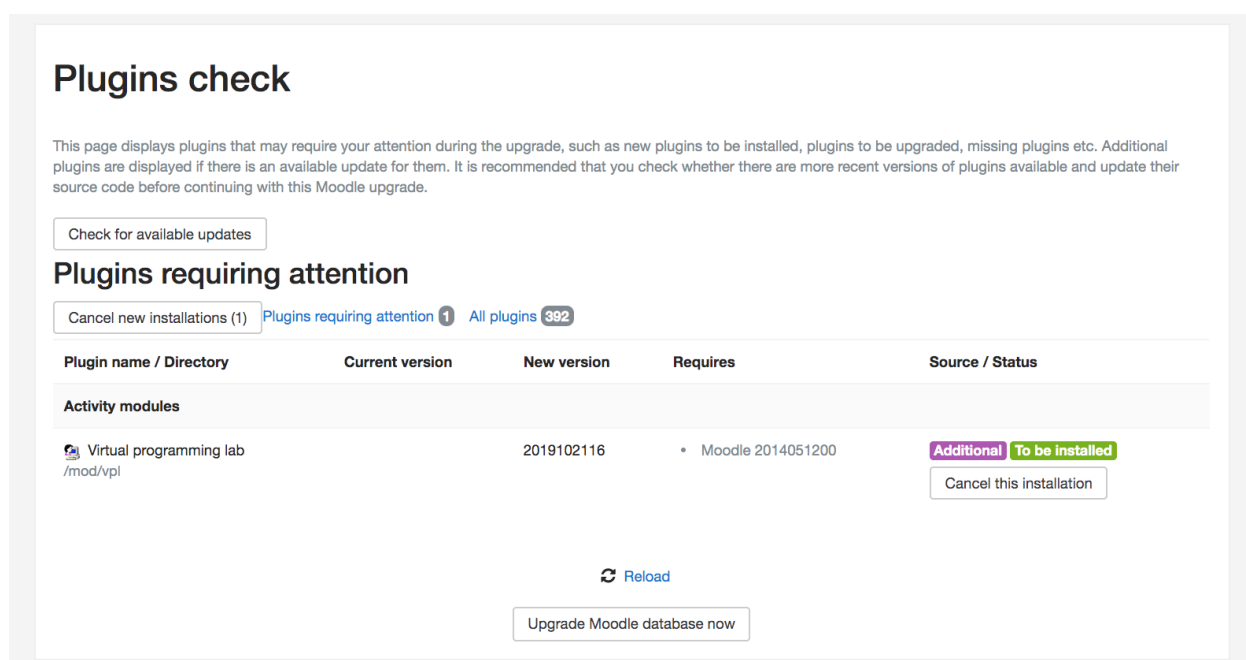


Figura 14 Vista actualizar base de datos de Moodle

Finalmente le mostrará la ventana de configuración del plugin. Hasta acá el plugin se ha instalado correctamente, sin embargo, deberemos indicarle donde se encuentra la jaula de ejecución para que pueda conectarse el plugin a ella.

En la ventana de configuración de Moodle VPL vaya a la sección llamada “Execution server list” y coloque la URL publica de su jaula de ejecución.

```
Execution servers list
mod_vpl | jail_servers
maximum default number of processes

# This server is only for test use.
# Install your own Jail server and remove the following line as soon as possible.
# http://demojail.dis.ulpgc.es

https://vpl.cloud.ufps.edu.co/

Default:
```

Figura 15 Vista agregar url de la jaula de ejecución a VPL en Moodle

## 3.3 Casos de uso

### 3.3.1 Profesor

*Crear actividad de VPL y configuración para calificar automáticamente*

Siga los siguientes pasos para crear una actividad de VPL

Vaya al curso en donde desea agregar una actividad de VPL, seleccione el modo de edición



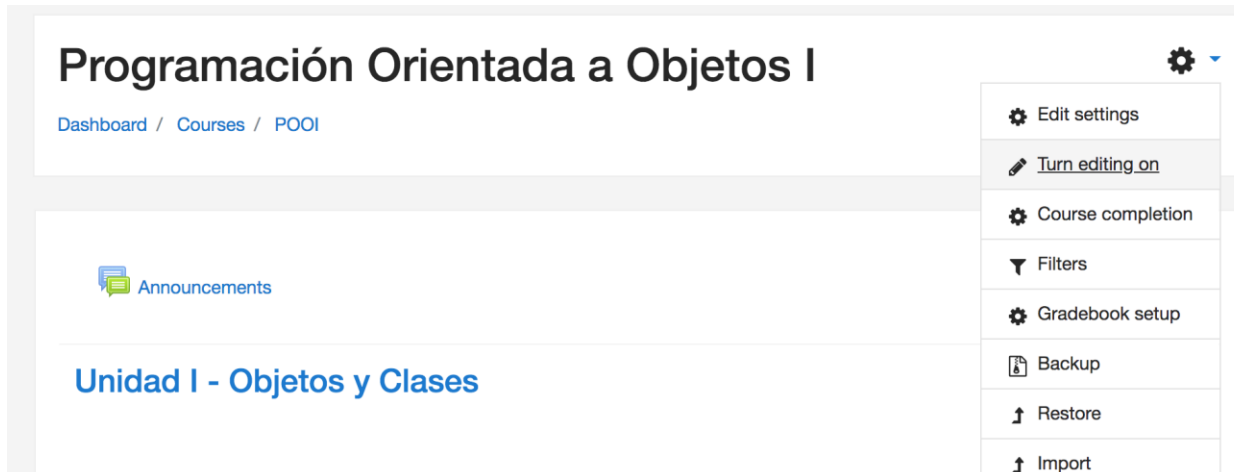


Figura 16 Vista activar edición en curso de Moodle

Haga clic en add activity or resource y seleccione Virtual Programming Lab

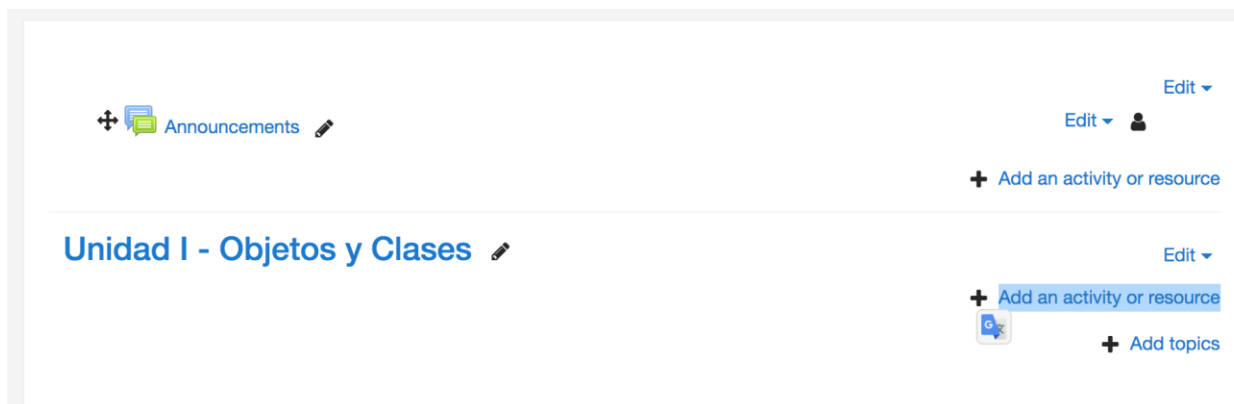


Figura 17 Vista añadir actividad o recurso a Moodle

Haga clic en el botón **add**

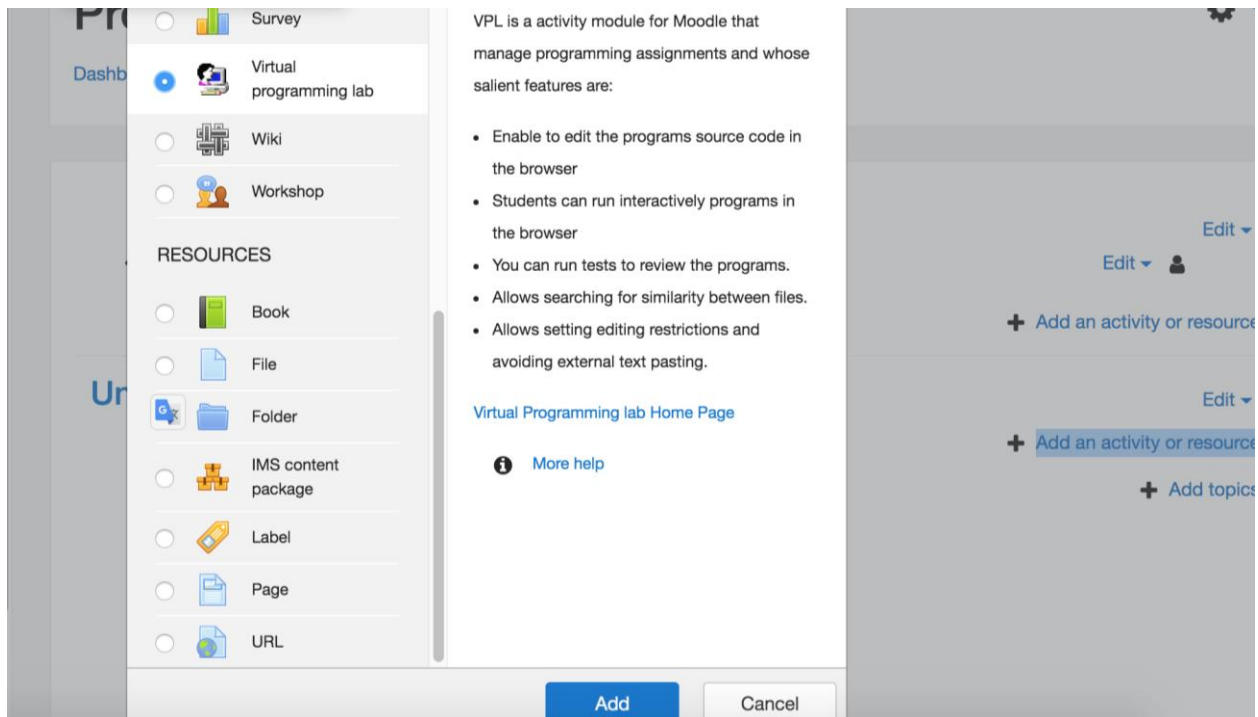


Figura 18 Vista añadir actividad de VPL a curso en Moodle

Coloque un nombre y una descripción, también puede configurar aspectos adicionales de las actividades de Moodle. Al finalizar haga clic en **save and display** para que al guardar lo redirija a la página de la actividad de VPL

▼ General

Name ! Tarea 5 - Calculadora de Fraccionarios en VPL

Short description

Full description

Display description on course page ?

▸ Submission period  
 ▸ Submission restrictions  
 ▸ Grade  
 ▸ Common module settings  
 ▸ Restrict access  
 ▸ Activity completion  
 ▸ Tags  
 ▸ Competencies

Figura 19 Vista configurar nueva actividad de VPL en Moodle

La actividad se verá de la siguiente manera:

## Programación Orientada a Objetos I

Dashboard / Courses / POOI / Unidad I - Objetos y Clases / Tarea 5 - Calculadora de Fraccionarios en VPL

Description Submissions list Similarity Test activity ⚙️

### Tarea 5 - Calculadora de Fraccionarios en VPL

**Due date:** Thursday, 6 February 2020, 12:00 AM  
**Maximum number of files:** 1  
**Type of work:** Individual work  
**Grade settings:** Maximum grade: 100  
**Run:** No. **Evaluate:** No

[Descargue el Proyecto en BlueJ de ejemplo propuesto de Calculador de Fraccionarios.](#)

Resuelva en VPL el problema.  
Adicionalmente implemente una solución completa usando JavaFX, usando la GUI de la Tarea anterior.

◀ Announcements
VPL

Jump to... ▾

Figura 20 Vista descripción de actividad de VPL en Moodle

Haga clic en el botón con forma de engrane, en la esquina superior derecha, y seleccione **execution options**



The screenshot shows the Moodle interface for a VPL activity titled "Tarea 5 - Calculadora de Fraccionarios en VPL". At the top right, there is a gear icon for settings. A dropdown menu is open, showing the following options: "Edit settings", "Test cases", "Execution options" (which is highlighted), "Requested files", "Advanced settings", "Execution files", and "Maximum execution resources limits".

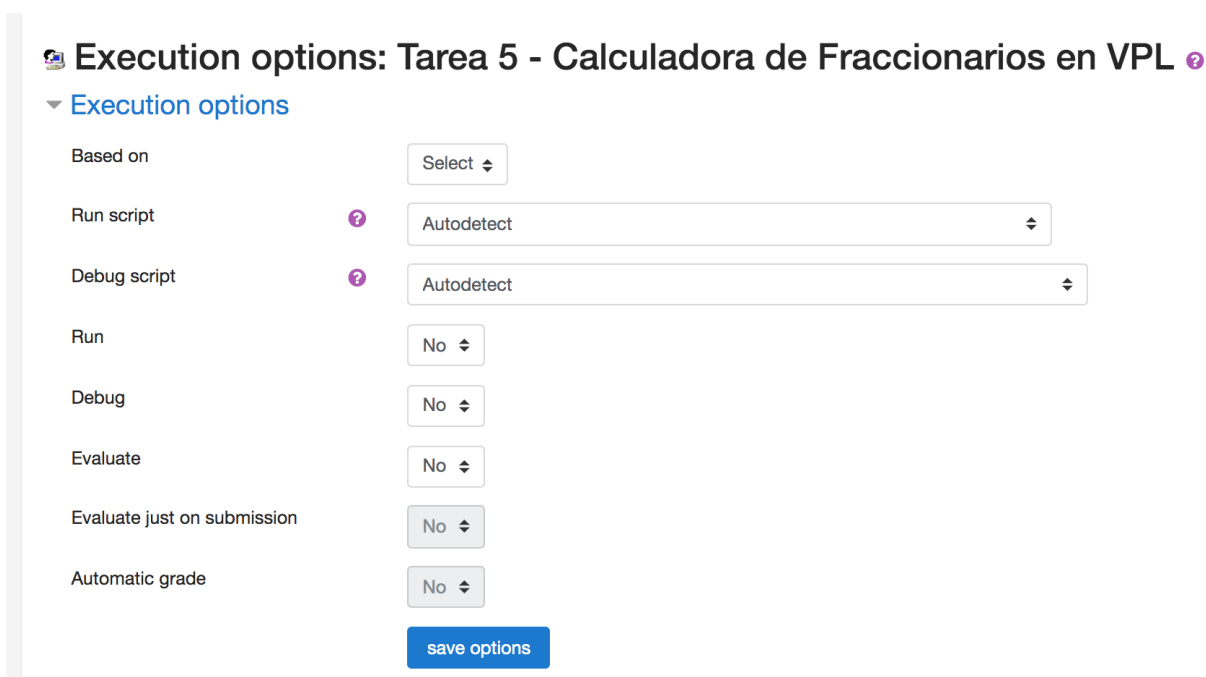
Below the menu, the activity details are visible:
 

- Due date:** Thursday, 6 February 2020, 12:00 AM
- Maximum number of files:** 1
- Type of work:** Individual work
- Grade settings:** Maximum grade: 100
- Run:** No. **Evaluate:** No

There is also a link: [Descargue el Proyecto en BlueJ de ejemplo propuesto de Calculador de Fraccionarios.](#)

Figura 21 Vista opciones de ejecución en actividad de VPL de Moodle

Se mostrará la siguiente ventana



The screenshot shows the "Execution options" configuration window for the activity "Tarea 5 - Calculadora de Fraccionarios en VPL". The window title is "Execution options: Tarea 5 - Calculadora de Fraccionarios en VPL". Below the title, there is a section for "Execution options" with the following settings:

- Based on:** Select
- Run script:** Autodetect
- Debug script:** Autodetect
- Run:** No
- Debug:** No
- Evaluate:** No
- Evaluate just on submission:** No
- Automatic grade:** No

At the bottom of the configuration, there is a blue button labeled "save options".

Figura 22 Vista de la configuración inicial de las opciones de ejecución de una actividad de VPL en Moodle

La opción **run** le permitirá al estudiante ejecutar sus programas en el IDE. La opción **debug** le permitirá al estudiante a hacer debug a sus programas. La opción **evaluate** le permitirá al estudiante evaluar sus programas. Esta última opción también permitirá calificar en Moodle automáticamente.

Seleccione **yes** en **run** y en **evaluate**. La última opción le permitirá activar los campos **evaluate just on submission** y **automatic grade**.

**Evaluate on submission** le permite ejecutar las pruebas automáticas cuando el estudiante suba sus programas o guarde sus programas en el IDE.

**Automatic grade** le indicará a VPL que califique automáticamente. En éste mismo capítulo se entrará en detalle cómo crear pruebas unitarias y su configuración para calificar automáticamente.

Seleccione **yes** en **evaluate just on submission** y en **automatic grade**.

Haga clic en guardar

## Execution options: Tarea 5 - Calculadora de Fraccionarios en VPL ?

Options have been saved ×

▼ Execution options

Based on

Run script

Debug script

Run

Debug

Evaluate

Evaluate just on submission

Automatic grade

Figura 23 Vista actualización exitosa de las opciones de ejecución de VPL en Moodle

*Crear pruebas unitarias que califiquen la actividad automáticamente*

Con VPL el profesor es capaz evaluar usando pruebas unitarias con JUnit sin hacer configuraciones adicionales.

Requerimientos:

1. Moodle
2. VPL Moodle
3. Jaula de ejecución instalada
4. JUnit instalado en la jaula de ejecución

Para generar una calificación automática deberá imprimir al final de la ejecución de los tests una línea que comience con `Grade :=>>` para generar un comentario durante la ejecución de los tests deberá imprimir líneas que comiencen con `Comment :=>>`, estas salidas en consola le indicarán a VPL cómo debe calificar y qué le mostrará al estudiante.

Devuélvase a la ventana de la actividad, y haga clic de nuevo en el engranaje, seleccione la opción `execution files`

Los archivos de ejecución son archivos que controlan la ejecución de las actividades de VPL dentro de la jaula de ejecución, aún si el profesor no crea archivos de ejecución, VPL enviará a la jaula de ejecución archivos de ejecución defaults dependiendo de la extensión de los archivos del estudiante. No se preocupe, capítulos abajo se hablará al respecto.

The screenshot shows the Moodle interface for an activity titled "Programación Orientada a Objetos I". The breadcrumb trail is: Dashboard / Courses / POOI / Unidad I - Objetos y Clases / Tarea 5 - Calculadora de Fraccionarios en VPL. The activity page has tabs for Description, Submissions list, Similarity, and Test activity. The main content area displays the activity title "Tarea 5 - Calculadora de Fraccionarios en VPL" and its details: Due date: Thursday, 6 February 2020, 12:00 AM; Maximum number of files: 1; Type of work: Individual work; Grade settings: Maximum grade: 100; Run: Yes, Evaluate: Yes, Evaluate just on submission: Yes; Automatic grade: Yes. A link is provided: "Descargue el Proyecto en BlueJ de ejemplo propuesto de Calculador de Fraccionarios." Below this, it says "Resuelva en VPL el problema. Adicionalmente implemente una solución completa usando JavaFX, usando la GUI de la Tarea anterior." On the right side, a settings menu is open, showing options like Edit settings, Test cases, Execution options, Requested files, Advanced settings, Execution files (which is highlighted), Maximum execution resources limits, and Files to keep when running.

Figura 24 Vista seleccionar opción archivos de ejecución para configurar actividad de VPL en Moodle

Verá la siguiente ventana

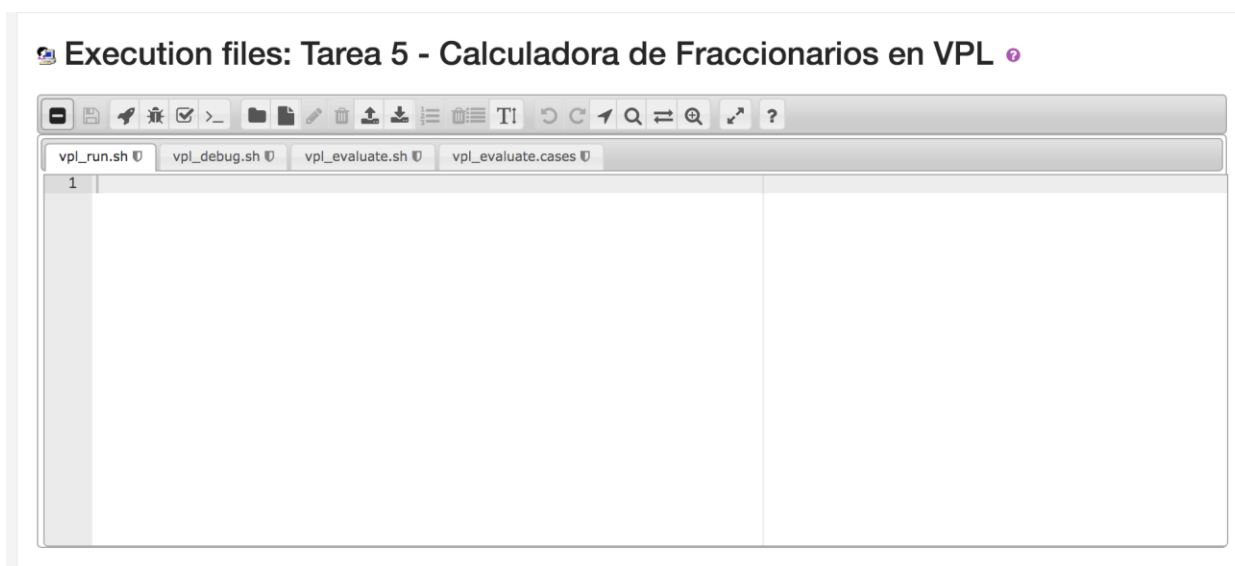


Figura 25 Vista inicial del IDE de VPL

Haga clic en `vpl_evaluate.sh` y solo agregue la línea: **`vpl_run.sh`** y de clic en el botón de guardar.

## Execution files: Tarea 5 - Calculadora de F

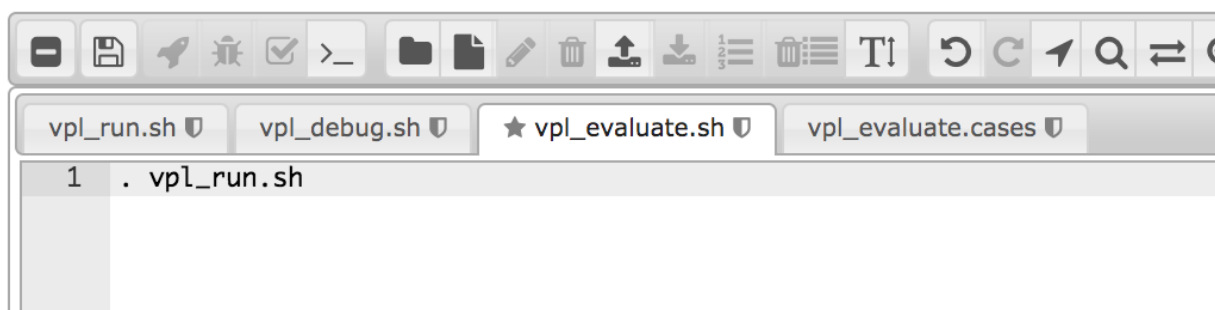


Figura 26 Vista modificar `vpl_evaluate` para activar las pruebas unitarias en las actividades de VPL en Moodle

Esa línea indica que cargaremos el archivo **`vpl_run.sh`**. Como usted notar, no modificaremos el archivo `vpl_run.sh`, con eso forzamos para que VPL envíe el archivo



vpl\_run.sh por defecto para archivos java. Este archivo está escrito en bash, y tiene las instrucciones necesarias para ejecutar JUnit.

Guarde el siguiente archivo en su disco duro, manteniendo el nombre del archivo y la extensión:

[https://raw.githubusercontent.com/alphonse92/VPLplusplus\\_composer/master/docs/original\\_Moodle/unidades/1/Tarea%205%20Calculadora%20de%20Fraccionarios%20en%20VPLLaboratorio%20virtual%20de%20programaci%C3%B3n/VPL/FraccionarioTest.java](https://raw.githubusercontent.com/alphonse92/VPLplusplus_composer/master/docs/original_Moodle/unidades/1/Tarea%205%20Calculadora%20de%20Fraccionarios%20en%20VPLLaboratorio%20virtual%20de%20programaci%C3%B3n/VPL/FraccionarioTest.java)

Este archivo controla la ejecución de los tests y captura las excepciones de los test, recuerde que JUnit arroja excepciones al fallar un test. Si el test se ejecuta correctamente suma a la variable, si no, crea un comentario.

Y este otro, también manteniendo el nombre del archivo y la extensión (java)

[https://raw.githubusercontent.com/alphonse92/VPLplusplus\\_composer/master/docs/original\\_Moodle/unidades/1/Tarea%205%20Calculadora%20de%20Fraccionarios%20en%20VPLLaboratorio%20virtual%20de%20programaci%C3%B3n/VPL/IntegralTest.java](https://raw.githubusercontent.com/alphonse92/VPLplusplus_composer/master/docs/original_Moodle/unidades/1/Tarea%205%20Calculadora%20de%20Fraccionarios%20en%20VPLLaboratorio%20virtual%20de%20programaci%C3%B3n/VPL/IntegralTest.java)

Este archivo contiene la clase con las pruebas unitarias. Es una prueba unitaria común y corriente. Si desea conocer más a profundidad sobre pruebas unitarias en JUnit visite:

<https://junit.org/junit4/>

Dentro del IDE busque el botón de subir archivo

Seleccione los dos archivos que descargó en el paso anterior y guarde.

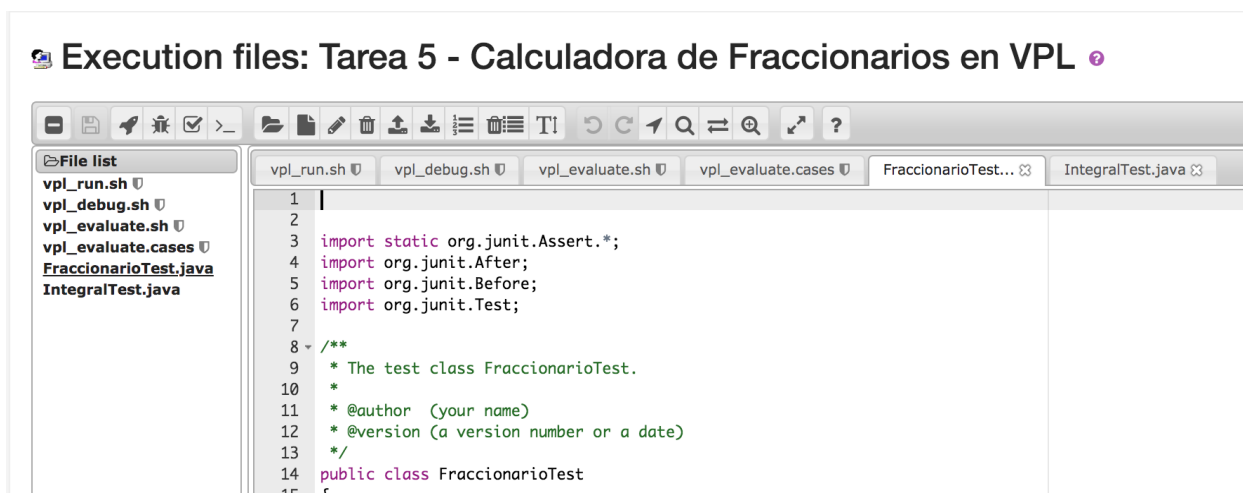


Figura 27 Vista fraccionarioTest.java

Puede inspeccionar el código y su funcionamiento, en el archivo IntegralTest.java tiene las salidas por consola para generar comentarios y la calificación

### 3.3.2 Estudiante

Los estudiantes podrán enviar sus programas desde su disco duro, o usar el IDE de VPL para crear sus programas, sea como sea, estos programas son enviados a la jaula de ejecución junto con los archivos de ejecución del profesor.

Como estudiante podrá acceder a las actividades de VPL

The screenshot shows a Moodle course page for 'Programación Orientada a Objetos I'. The breadcrumb trail is 'Dashboard / My courses / POOI / Unidad I - Objetos y Clases / Tarea 5 - Calculadora de Fraccionarios en VPL'. The page has four tabs: 'Description' (selected), 'Submission', 'Edit', and 'Submission view'. The main heading is 'Tarea 5 - Calculadora de Fraccionarios en VPL'. Below the heading, it lists: 'Due date: Thursday, 6 February 2020, 12:00 AM', 'Maximum number of files: 1', and 'Type of work: Individual work'. A blue link reads 'Descargue el Proyecto en BlueJ de ejemplo propuesto de Calculador de Fraccionarios.'. The instructions state: 'Resuelva en VPL el problema. Adicionalmente implemente una solución completa usando JavaFX, usando la GUI de la Tarea anterior.'

Figura 28 Vista de la actividad de VPL desde el estudiante en Moodle

Haga clic en submission, le mostrará una ventana para poder subir sus programas.

Descargue el siguiente programa desde:

[https://raw.githubusercontent.com/alphonse92/VPLplusplus\\_composer/master/docs/original\\_Moodle/unidades/entregas/muestra/1/Tarea%20%205%20-%20Calculadora%20de%20Fraccionarios%20en%20VPL/Quintero%20Reyes%20Johnny%20Alexander%201893/2019-09-07-23-17-33/Fraccionario.java](https://raw.githubusercontent.com/alphonse92/VPLplusplus_composer/master/docs/original_Moodle/unidades/entregas/muestra/1/Tarea%20%205%20-%20Calculadora%20de%20Fraccionarios%20en%20VPL/Quintero%20Reyes%20Johnny%20Alexander%201893/2019-09-07-23-17-33/Fraccionario.java)

Guárdelo en su disco duro manteniendo la extensión y el nombre.

Description Submission Edit Submission view

▼ Submission

Comments

Any file  Maximum size for new files: 2MB  
Fraccionario.java

◀ Announcements

Figura 29 Vista agregar entrega a actividad de VPL en Moodle

Luego de clic en el botón submit

Description Submission Edit Submission view

Saved

Figura 30 Vista entrega guarda para actividad de VPL en Moodle

Haga clic en continue



Figura 31 Vista evaluación en progreso de la respuesta de un estudiante en VPL en Moodle

El IDE le mostrará que empezó a evaluar la actividad automáticamente. Al finalizar podrá ver los resultados

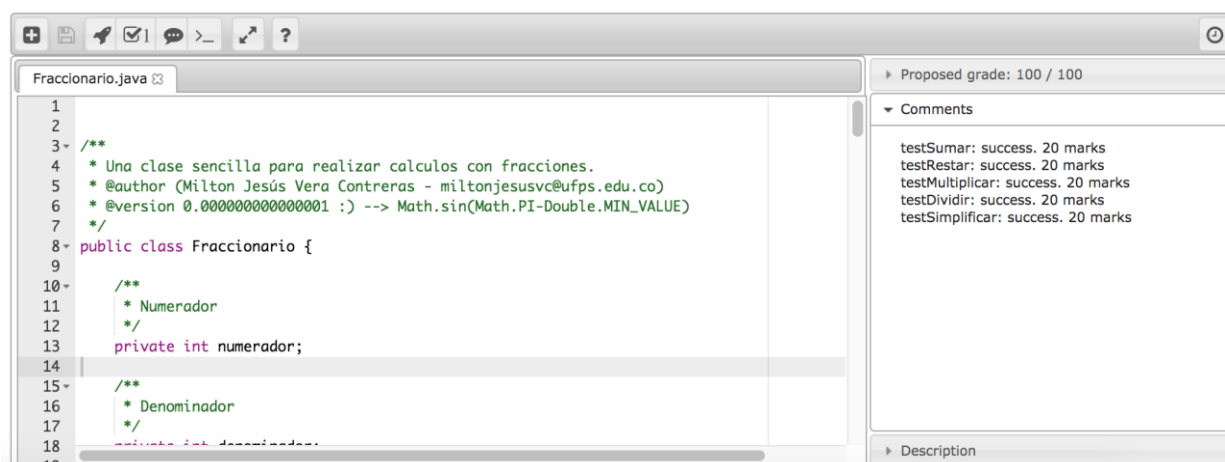


Figura 32 Vista de la entrega del estudiante en el IDE de VPL en Moodle

Si no desea enviar archivos desde su computador, puede optar por usar el IDE de VPL. En la página de la actividad de VPL haga clic en la pestaña edit.

Description   Submission   **Edit**   Submission view

Figura 33 Vista del estudiante de la opción editar entrega en la actividad de VPL en Moodle

En ella abrirá el IDE, al comenzar le pedirá el nombre del archivo que desea crear, recuerde colocarle la extensión.

Los botones (de izquierda a derecha) son:



Figura 34 Botones de guardar, correr y evaluar entrega en el IDE de VPL en Moodle

1. Guardar: le permite guardar los cambios sobre un programa.
2. Run: le permite ejecutar su código como si estuviera en su computador.
3. Evaluate: le permite correr las pruebas de evaluación de su programa.

## 3.4 Ventajas y limitantes

VPL es una herramienta útil para enseñar Programación como el profesor Milton lo ha probado(Vera et al., 2018).

1. Disminuye la carga laboral del profesor calificando automáticamente
2. Retroalimentación inmediata para el profesor y el estudiante
3. Permite combatir el plagio
4. Es agnóstico del lenguaje de programación
5. Permite hacer pruebas unitarias

Sin embargo, VPL solo ofrece resultados estáticos y de ejecución, es decir, solo informa si un caso de prueba de una prueba unitaria pasó o no, y dependiendo de esto califica. VPL no ofrece información de valor sobre el rendimiento del estudiante, o sobre qué casos de prueba falla más el estudiante.

Moodle y Virtual Programming Lab es un software que no es difícil de instalar. Además, no requiere muchos recursos, y es de fácil acceso para profesores y estudiantes.

Éste capítulo ayudó a entender como VPL funciona, y esto le permitirá a profesores y estudiantes extenderlo y fortalecerlo. El funcionamiento de VPL es relativamente sencillo y se pueden lograr grandes implementaciones como se pudo demostrar con las pruebas unitarias.

Se recomienda a los profesores tener en cuenta esta herramienta en las materias donde se enseñe programación.

## 4. Pruebas unitarias en Java y librería para generación automática de pruebas

En el área del software es imprescindible hacer uso de las pruebas del software, para poder encontrar defectos, fallas y observar el comportamiento del mismo cuando ejecutamos los casos de prueba. Básicamente planteamos entradas con salidas conocidas y observamos los resultados. Muchos factores pueden tomarse en cuenta para decir si una prueba falló o pasó:

1. Las salidas de las entradas que se plantean en los casos de prueba deben ser iguales o con un margen mínimo de error a las salidas conocidas.
2. El procesamiento de la información debe ser en un tiempo definido por el tester, si ese tiempo es excedido falla la prueba.
3. El caso de prueba no necesariamente debe ser afirmativo, es en pos de lo que espera el desarrollador; incluso errores o excepciones.

### 4.1 Generación automática de tests vs generación manual

Existen diferentes herramientas para generar casos de prueba con una tasa alta de cobertura de código. A falta de una especificación el desarrollador debe crear sus propios casos de pruebas donde una entrada conocida debe generar una salida conocida y aplicando los criterios anteriores puede inferir si falló o no falló la prueba. Se supone que las pruebas automáticas deberían (en teoría) facilitar la tarea al tester y aunque por décadas ha persistido esta suposición la realidad es diferente, la poca adopción de las herramientas de generación de casos de prueba en la industria es escaso, y hasta el 2014 no ha habido prueba concluyente de ésta afirmación



En el paper: *Does Automated Unit Test Generation Really Help Software Testers? A Controlled Empirical Study* se tomó un experimento que involucró a 49 sujetos y tres clases, extendiéndose a 48 sujetos y cuatro clases. En el estudio cada uno de los sujetos eligió una o dos clases y se pidió que crearan pruebas unitarias manualmente usando el framework JUnit a las clases elegidas. Usando ambos estudios se tomaron en total 145 combinaciones entre sujetos y clases. Se obtuvieron los siguientes resultados:

1. Se confirmó que las herramientas generadoras de test automáticos son efectivos y hacen lo que deben hacer. Comparado con los tests generados por los sujetos se encontró que tuvieron un alto índice de cobertura del código.
2. El estudio no confirma que las herramientas de tests automáticos encuentran fallas. Ambos (herramientas y sujetos) encontraron el mismo número de fallas.
3. Adicionalmente se encontró que el desarrollador gasta más tiempo analizando la prueba generada por una herramienta que creándola. Si la prueba generada por la herramienta es errónea claramente es perjudicial para la prueba

## 4.2 Evosuite

En el estudio anterior fue usada la herramienta para generación automática de pruebas de software en el lenguaje de programación Java junto a JUnit 4. Evosuite usa un algoritmo evolutivo para lograrlo, genera una semilla basada en la hora del sistema y se ejecuta iterativamente.

Evosuite usa un enfoque basado en búsqueda de ejecución simbólica dinámica, transformaciones de estabilidad y búsqueda híbrida (Fraser & Arcuri, 2011), Evosuite es capaz de generar pruebas para clases sencillas o proyectos enteros sin intervención de persona alguna, además es capaz de usar cualquier tipo de objeto complejo, llamar a métodos, usar constructores

y datos primitivos, sin embargo está limitado a su capacidad de manejar aplicaciones de un solo thread.

## 4.3 Instalación

Evosuite se puede descargar desde <http://www.Evosuite.org/>

o en su repositorio listo para ser descargado como Jar o Maven Project listo para su compilación en git: <https://github.com/Evosuite/Evosuite/>

Para su ejecución puede usar el prompt del sistema ejecutando el entorno de ejecución de java (JRE) junto al parámetro -jar seguido del nombre del jar que hemos descargado.

```
usuario@DESKTOP-3A59NKC /k/documentos/Development/librerias/java/evosuit/jar
$ java -jar evosuite-1.0.3.jar
* Evosuite 1.0.3
```

Figura 35 Ejecutar Evosuite

Ejecute Evosuite usando el parámetro -help podemos observar los diferentes parámetros que acepta junto a una breve descripción de los mismos.

```

usuario@DESKTOP-3A59NKC MINGW64 /k/documentos/Development/librerias/java/evosuit/jar
$ java -jar evosuite-1.0.3.jar -help
* EvoSuite 1.0.3
usage: EvoSuite
  -base_dir <arg>           Working directory in which tests and reports
                             will be placed
  -class <arg>              target class for test generation. A fully
                             qualifying needs to be provided, e.g.
                             org.foo.SomeClass
  -continuous <arg>        Run Continuous Test Generation (CTG). Valid
                             values are: [EXECUTE, INFO, CLEAN]
  -criterion <arg>          target criterion for test generation. Can
                             define more than one criterion by using a ':'
                             separated list
  -D <property=value>       use value for given property
  -evosuiteCP <arg>         classpath of EvoSuite jar file(s). This is
                             needed when EvoSuite is called in plugins like
                             Eclipse/Maven
  -extend <arg>             extend an existing test suite
  -generateMOSuite          use many objective test generation (MOSA).
  -generateNumRandom <arg> generate fixed number of random tests
  -generateRandom           use random test generation
  -generateSuite            use whole suite generation. This is the
                             default behavior
  -generateTests            use individual test generation (old approach
                             for reference purposes)
  -heapdump                 Create heap dump on client VM out of memory
                             error
  -help                     print this message
  -inheritanceTree          Cache inheritance tree during setup
  -junit <arg>              junit prefix
  -libraryPath <arg>        java library path to native libraries of the
                             project under test
  -listClasses              list the testable classes found in the
                             specified classpath/prefix
  -listParameters           list all the parameters that can be set with
                             -D
  -measureCoverage          measure coverage on existing test cases
  -mem <arg>                heap size for client process (in megabytes)
  -prefix <arg>             target package prefix for test generation. All
                             classes on the classpath with the given
                             package prefix will be used, i.e. all classes
                             in the given package and sub-packages.
  -printStats              print class information (coverable goals)
  -projectCP <arg>         classpath of the project under test and all
                             its dependencies
  -regressionSuite          generate a regression test suite
  -regressionTests         generate a regression test suite of individual
                             tests
  -seed <arg>               seed for random number generator
  -setup <arg>              Create evosuite-files with property file
  -startedByCtg            Determine if current process was started by a
                             CTG process
  -target <arg>             target classpath for test generation. Either a
                             jar file or a folder where to find the .class
                             files
  -writeDependencies <arg> write the dependencies of a target class to

```

Figura 36 Evosuite comandos

## 4.4 Uso

### 4.4.1 Crear clase ejemplo

Cree una carpeta llamada “trying” al mismo nivel en el árbol de directorios en el que se encuentra Evosuite-\*.jar, en trying se crea una carpeta donde almacenaremos las clases que habremos creado, la nombraremos “src”. Dentro de ella debe crear una carpeta llamada “main”, esta será un paquete de la aplicación básica que queremos construir. Por último, cree un archivo .java llamado Numeric.java que contendrá el siguiente código:

```
package main;
public class Numeric {

public int multiply(int i1,int i2)
{
    return i1*i2;
}

public int add(int i1,int i2)
{
    return i1+i2;
}

}
```

Esta clase sólo contiene dos métodos para multiplicar y sumar dos números ingresados en los parámetros y devolverá como salida el resultado.

Finalmente obtendrá la siguiente ruta:

trying/src/main/Numeric.java

Usando su IDE favorito o la consola compilamos ésta clase y guarde el archivo generado (Numeric.class) al mismo nivel del árbol de directorio que Numeric.java

#### 4.4.1 Establecer classpath y generar pruebas unitarias a partir de una clase

Aunque Evosuite admite más parámetros los más básicos son:

1. `-class [path to class]`: el parámetro debe ser la ruta de nuestra clase dentro del proyecto usando un nombre cualificado (qualifying name) en éste caso será: `main.Numeric`, omitimos la extensión.
2. `-projectCP [path to project folder]` En este caso será `trying/src/`

Finalmente se verá así nuestra consola:

```
usuario@DESKTOP-3A59NKC MINGW64 /k/documentos/Development/librerias/java/evosuit/jar
$ java -jar evosuite-1.0.3.jar -class asdasd -projectCP trying/src/
```

Figura 37 Ejecutar Evosuite usando carpeta

Evosuite le informará paso a paso el progreso y la evolución durante la creación de los tests

```

usuario@DESKTOP-3A59NKC MINGW64 /k/documentos/Development/librerias/java/evosuit/jar
$ java -jar evosuite-1.0.3.jar -class main.Numeric -projectCP trying/src/
* EvoSuite 1.0.3
* Going to generate test cases for class: main.Numeric
* Starting client
* Connecting to master process on port 9716
* Analyzing classpath:
- trying/src/
* Finished analyzing classpath
* Generating tests for class main.Numeric
* Test criteria:
- Line Coverage
- Branch Coverage
- Exception
- Mutation testing (weak)
- Method-Output Coverage
- Top-Level Method Coverage
- No-Exception Top-Level Method Coverage
- Context Branch Coverage
* Setting up search algorithm for whole suite generation
* Total number of test goals:
- Line 2
- Branch 3
- Exception 0
- MutationFactory 24
- Output 6
[Progress:>                                0%] [Cov:>                                0%] - Method 3
- MethodNoException 3
- CBranchFitnessFactory 3
* Using seed 1494384917523
* Starting evolution
,

```

Figura 38 Salida por consola al ejecutar Evosuite

Al finalizar le diré **done**, y podrá encontrar los tests en una carpeta que Evosuite habrá creado al mismo nivel del árbol de directorios donde se encuentra Evosuite-\*.jar

Para cada caso de prueba Evosuite genera un método @test, junto a un registro de la trazabilidad durante la creación del assert

```

//Test case number: 1
/*
 * 22 covered goals:
 * Goal 1. main.Numeric.multiply(II)I: root-Branch
 * Goal 2. main.Numeric.<init>()V: root-Branch
 * Goal 3. Branch main.Numeric.<init>()V: root-Branch in context: main.Numeric:<init>()V
 * Goal 4. Branch main.Numeric.multiply(II)I: root-Branch in context: main.Numeric:multiply(II)I
 * Goal 5. [Output]: main.Numeric.multiply(II)I:Negative
 * Goal 6. main.Numeric.multiply(II)I: Line 9
 * Goal 7. [METHOD] main.Numeric.<init>()V
 * Goal 8. [METHOD] main.Numeric.multiply(II)I
 * Goal 9. [METHODNOEX] main.Numeric.<init>()V
 * Goal 10. [METHODNOEX] main.Numeric.multiply(II)I
 * Goal 11. Weak Mutation 0: main.Numeric.multiply(II)I:9 - ReplaceVariable i1 -> i2
 * Goal 12. Weak Mutation 1: main.Numeric.multiply(II)I:9 - InsertUnaryOp Negation of i1
 * Goal 13. Weak Mutation 2: main.Numeric.multiply(II)I:9 - InsertUnaryOp IINC 1 i1
 * Goal 14. Weak Mutation 3: main.Numeric.multiply(II)I:9 - InsertUnaryOp IINC -1 i1
 * Goal 15. Weak Mutation 4: main.Numeric.multiply(II)I:9 - ReplaceVariable i2 -> i1
 * Goal 16. Weak Mutation 5: main.Numeric.multiply(II)I:9 - InsertUnaryOp Negation of i2
 * Goal 17. Weak Mutation 6: main.Numeric.multiply(II)I:9 - InsertUnaryOp IINC 1 i2
 * Goal 18. Weak Mutation 7: main.Numeric.multiply(II)I:9 - InsertUnaryOp IINC -1 i2
 * Goal 19. Weak Mutation 8: main.Numeric.multiply(II)I:9 - ReplaceArithmeticOperator * -> +
 * Goal 20. Weak Mutation 9: main.Numeric.multiply(II)I:9 - ReplaceArithmeticOperator * -> %
 * Goal 21. Weak Mutation 10: main.Numeric.multiply(II)I:9 - ReplaceArithmeticOperator * -> -
 * Goal 22. Weak Mutation 11: main.Numeric.multiply(II)I:9 - ReplaceArithmeticOperator * -> /
 */

@Test(timeout = 4000)
public void test1() throws Throwable {
    Numeric numeric0 = new Numeric();
    int int0 = numeric0.multiply((-703), 978);
    assertEquals((-687534), int0);
}

```

Figura 39 test generado por Evosuite

## 4.5 Ejecutar pruebas unitarias generadas por Evosuite usando JUnit

Requerirá las siguientes dependencias para poder ejecutar los tests que Evosuite produjo

1. JUnit, disponible en <http://junit.org/junit4/>, se probó la versión 4.12
2. Simple Logging Facade for Java disponible en <https://www.slf4j.org/>, específicamente las librerías `slf4j-api-*.jar` y `slf4j-simple-*.jar`
3. Hamcrest core, se puede descargar desde la web de JUnit, se probó la versión `hamcrest-core-1.3`

4. Evosuite standalone runtime, descargada junto a Evosuite.

Puede ejecutar esos tests desde línea de comandos o usando su IDE favorito.

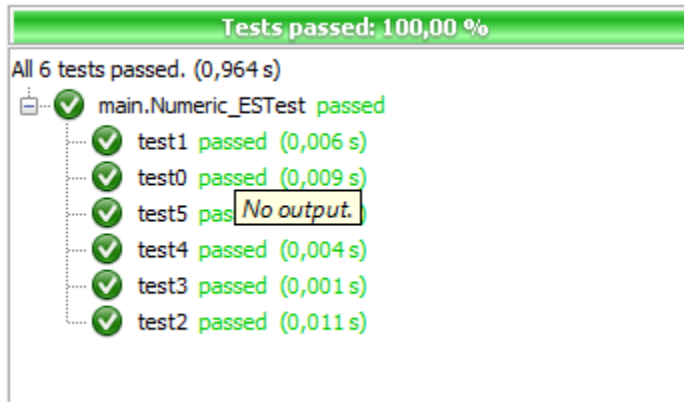


Figura 40 Resultado de correr los tests generados or Evosuite

## 4.6 Evosuite y VPL ++

Evosuite es una herramienta poderosa que permite generar una gran cantidad de test automáticamente a partir de una clase. Sin embargo, se decidió que su integración a VPL ++ depende del profesor. Pues los tests generados por Evosuite dependen exclusivamente a partir de una clase original. Si la clase original no cumple con las reglas de negocio Evosuite no generará casos de prueba para evaluar reglas de negocio.

Por otro lado, los casos de prueba de VPL ++ deben tener como objetivo evaluar aspectos del estudiante de una manera atómica. Por ejemplo, si el profesor quiere conocer si el estudiante sabe herencia, él deberá crear casos de prueba que validen esto. Es el profesor quien debe asignar un tópico/tema/conocimiento a cada caso de prueba, Evosuite es incapaz de esto.

Evosuite puede ayudar al profesor a diseñar las pruebas para las diferentes actividades de VPL. Por otro lado, el profesor puede aprovechar que VPL soporta múltiples lenguajes de



programación como Python, JavaScript, Perl, PHP, etc... junto con sus herramientas para hacer testing.

## 5. Virtual Programming Lab ++

Virtual Programming Lab ++ o VPL ++ es un conjunto de software que extiende el proceso de ejecución de software en VPL. VPL ++ ejecuta pruebas unitarias a los programas de los estudiantes y profesores, y genera reportes sobre la ejecución de éstos programas. Los reportes son generados según tópicos (Objetivos específicos) vinculados a cada caso de prueba en una suite de pruebas unitarias de VPL.

En conjunto, las siguientes piezas de software solucionan una de las limitantes de VPL:

1. JLib Runner: es una librería para crear pruebas unitarias de VPL ++ y además es un software que ejecuta dichas pruebas.
2. VPL ++ api: es un microservicio que expone una API web escrita en JavaScript, ofrece servicios al cliente web (frontend) y a JLib Runner.
3. VPL ++ Cliente: es un microservicio que contiene el frontend de VPL ++ y ofrece servicios al administrador de Moodle y a los profesores.
4. VPL ++ Gateway: es un microservicio que hace como punto único de entrada de las peticiones de profesor y estudiantes, y redirigirlas al microservicio adecuado.
5. Sin embargo, y como se planteó en el anteproyecto, era extender el código fuente de VPL, en los capítulos siguientes describiremos el proceso de análisis que llevó a la solución actual.

## 5.1 Metodología de desarrollo

Existen dos paradigmas en la investigación aplicable a la ingeniería: ciencia orientada al comportamiento y ciencia orientada al diseño (Esearch et al., 2004b) . El primero busca verificar las teorías que explican o predicen el comportamiento humano individual o dentro de una organización, su objetivo es la verdad. La segunda busca extender las fronteras de las capacidades humanas y organizacionales creando e innovando artefactos, su objetivo es la utilidad. Éste proyecto se ubica en el segundo enfoque, pues no se desea probar ninguna teoría, en su lugar se quiere extender un artefacto existente, extender VPL a VPL++. La extensión permitirá apoyar la solución de un problema específico, para el caso la educación en programación de computadores.

Si bien el proyecto implica ciertas tareas de desarrollo de software, también implica otras tareas que no encajan dentro de una metodología específica de desarrollo de software. Por lo tanto, metodológicamente se estructura el proyecto en dos etapas, las cuales determinan claramente el proceso a seguir y sus entregables.

## 5.2 Etapa 1: Apropiación de VPL y Moodle

Para lograr el éxito de éste proyecto fue necesario conocer cada pieza de software que se involucra en VPL. Lo principal fue conocer, como Moodle funciona, su arquitectura, sistema de roles y qué interfaces expone Moodle.

Se necesitó conocer lo siguiente para poder continuar en el análisis y poder implementar una solución adecuada:

## 5.2.1 Moodle y VPL Plugin

1. Conocer la arquitectura de Moodle lo suficiente para saber su rol principal en éste proyecto
2. Conocer cómo VPL se integra Moodle dio claridad para entender cómo y dónde debería modificar VPL para añadir las nuevas funcionalidades
3. Entender los casos de uso del profesor en una actividad de VPL
4. Entender los casos de uso del estudiante en una actividad de VPL
5. Entender las opciones avanzadas de VPL
6. Entender lo que procesos se ejecutan cuando se crea una actividad de VPL
7. Entender cómo el plugin se comunica con la jaula de ejecución

## 5.2.2 VPL Jaula de ejecución

1. Conocer el proceso de instalación de la jaula de ejecución.
2. Conocer el proceso de ejecución del código del estudiante, y los archivos de ejecución del profesor.

El siguiente diagrama describe las actividades y productos obtenidos de cada una de ellas para ésta etapa

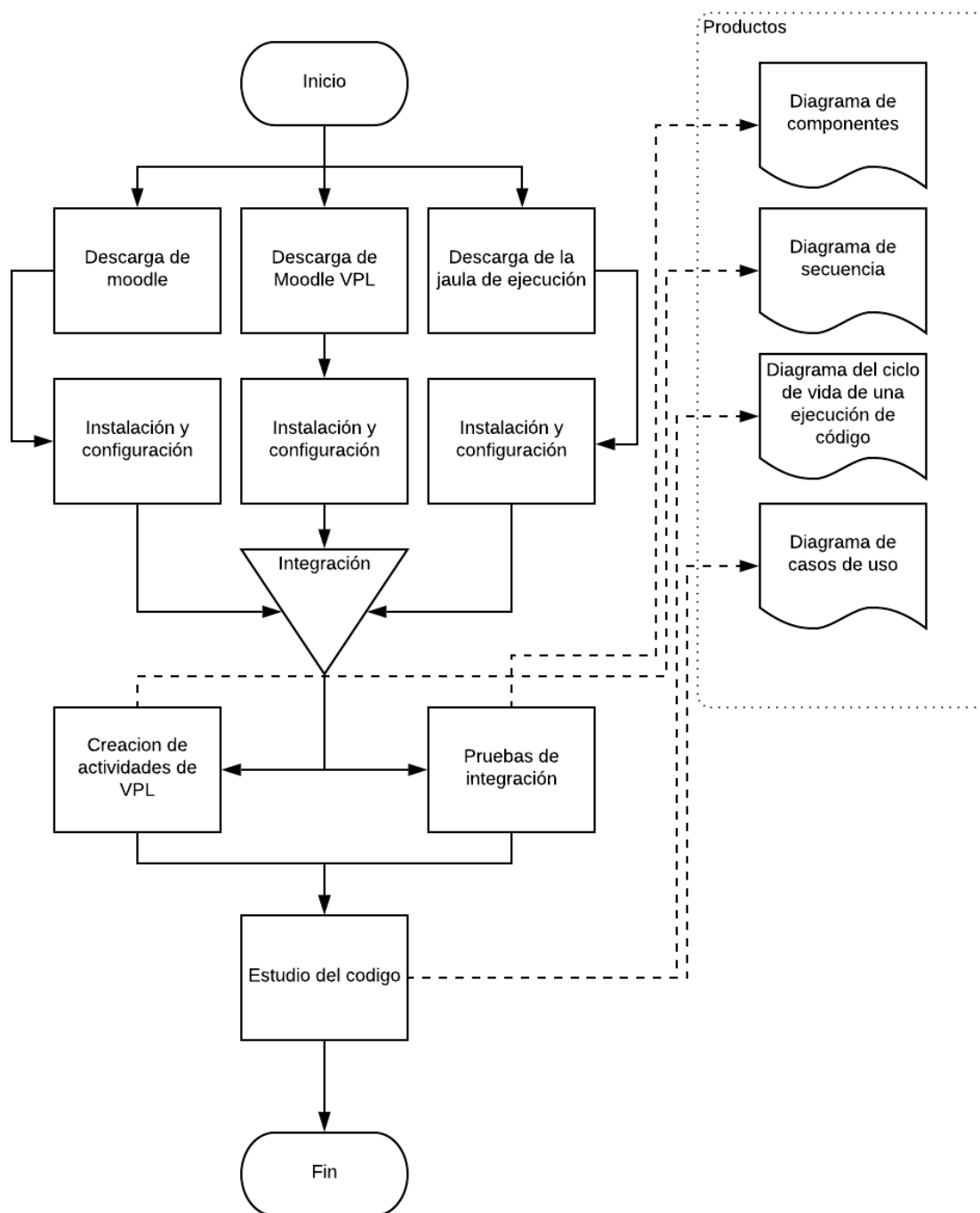


Figura 41 Diagrama de proceso de desarrollo de la etapa 1

### 5.2.3 Análisis de los componentes

Lo primero que se hizo fue instalar Moodle y plugin, la base de datos y la jaula de ejecución de VPL. Este proceso ayudó a identificar los componentes que hace posible la ejecución de VPL, además los requerimientos de hardware necesarios para su instalación. Se identificaron cuatro componentes que permiten el funcionamiento de VPL:

1. Servidor Apache + PHP: Este componente es el encargado de servir Moodle al cliente, está compuesto por un servidor HTTP (Apache en éste caso) y PHP como cgi del servidor.
2. Servidor de base de datos Mysql: actúa como motor de base de datos para Moodle y sus componentes
3. Plugin VPL: es el núcleo de VPL, está escrito en PHP al igual que Moodle e implementa el Framework que Moodle dispone para sus plugins. Como cualquier plugin de VPL, éste tiene sus carpetas específicas, nombradas semánticamente, de manera que pueda distribuir por carpeta cada una de las responsabilidades. Por ejemplo, la carpeta form contiene las clases/funciones con el código necesario para imprimir los diferentes formularios, en la carpeta db tiene los scripts que incluyen las clases/funciones que permiten instalar la base de datos para el plugin, mientras que la carpeta CSS contiene todos los archivos css que incluyen los diferentes estilos del plugin. Más adelante se analizará su arquitectura lógica.
4. Jaula de ejecución: la jaula de ejecución es un componente fundamental para VPL pues el plugin de Moodle trabaja en conjunto con ella. Como su nombre lo indica, el término jaula de ejecución (comúnmente llamado sandbox en otra literatura) hace referencia a un componente de software capaz de ejecutar código sin riesgo alguno, éste es capaz de separar de un contexto de producción la ejecución de cualquier programa.

En términos más sencillos: es un servidor que es responsable de ejecutar todo el código de los estudiantes y profesores que usan VPL sin riesgo alguno, si algún estudiante hace algo inapropiado es posible contener la amenaza aislada de la base de datos y del plugin. Esta jaula de ejecución es un software también creado por los desarrolladores de VPL, éste actúa como un servidor de llamadas a procedimientos remotos o RPC (remote procedure call) siendo éstas las siglas de su traducción al inglés.

Durante su instalación la jaula de ejecución instala (valga la redundancia) todo el software y herramientas necesarias para la compilación y ejecución de código en múltiples lenguajes, puede tomar varias horas y requerirá de la atención del usuario. La jaula de ejecución implementa además su propio Firewall de manera que adquiere una protección extra ante cualquier intento de conexiones entrantes o salientes no autorizadas.

## 5.2.4 Funcionamiento de la jaula de ejecución

Explicaremos algunos aspectos de la jaula de ejecución. Es muy importante conocer esto, ya que, si usted desea modificar el comportamiento de una ejecución, agregar o quitar software, librerías etc., deberá tenerlo claro. Por ejemplo, si quiere agregar módulos globales de npm para archivos de node.js o librerías de desarrollo de java como Apache commons o cualquier otro software.

### *Archivos de ejecución*

VPL envía los archivos necesarios para ejecutar el programa del estudiante. Internamente el plugin VPL almacena en la carpeta “jail” scripts escritos en bash, donde cada script es responsable de compilar y ejecutar código según el lenguaje de programación que se intente ejecutar, es decir hay scripts para ejecutar java, Python, PHP, JavaScript entre otros lenguajes.

Dichos scripts se envían junto al código que el estudiante o el profesor intenta ejecutar a la jaula de ejecución.

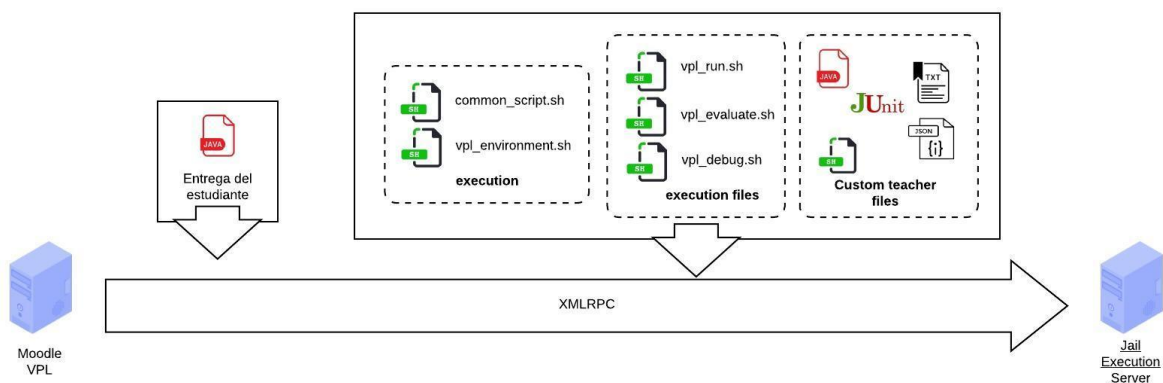


Figura 42 Empaquetado de los archivos de ejecución que son enviados a la Jaula de Ejecución de VPL

Cada archivo de ejecución está relacionado a un caso de uso del estudiante o profesor

1. vpl\_run: está relacionado con el caso de uso ejecutar código (sin evaluar)
2. vpl\_evaluate: está relacionado con el caso de uso ejecutar código y evaluar la actividad
3. vpl\_debug: está relacionado con el caso de uso hacer debug de código

Por tanto, podemos observar dentro de la ejecución de una actividad los archivos enviados a la jaula de ejecución.

Para hacer este ejercicio vaya a una actividad en VPL, en el archivo vpl\_evaluate.sh agregue la línea:

```
ls -la
```



Y guarde. Esta línea es un comando en UNIX para listar todos los archivos, sin parámetros, toma como el directorio actual desde donde se ejecutó. Después de guardar, haga clic en el botón de evaluar. Obtendrá la siguiente salida

### Execution files: Tarea 1 - Ejercicio 8 - Casa de Cambio Tio Rico VPLLaboratorio virtual de programación

```

1 ls -la
2
total 40
drwx----- 2 p15153 p15153 200 Jan 29 15:38 .
drwx--x--x 3 root root 60 Jan 29 15:38 ..
-rw----- 1 p15153 p15153 9285 Jan 29 15:38 CasaDeCambioTest.java
-rw----- 1 p15153 p15153 6 Jan 29 15:38 asdasd.java
-rwx----- 1 p15153 p15153 4971 Jan 29 15:38 common_script.sh
-rwx----- 1 p15153 p15153 168 Jan 29 15:38 vpl_debug.sh
-rwx----- 1 p15153 p15153 481 Jan 29 15:38 vpl_environment.sh
-rw----- 1 p15153 p15153 0 Jan 29 15:38 vpl_evaluate.cases
-rwx----- 1 p15153 p15153 19 Jan 29 15:38 vpl_evaluate.sh
-rwx----- 1 p15153 p15153 234 Jan 29 15:38 vpl_run.sh

```

Figura 43 listar archivos y carpetas dentro del sistema de archivos del proceso de ejecución de una actividad de VPL en Moodle

A la derecha se observan los 4 archivos de ejecución para cada caso de uso:

1. vpl\_debug.sh
2. vpl\_evaluate.sh
3. vpl\_debug.sh

También se listan los archivos de ejecución propios del profesor como por ejemplo CasaDeCambioTest.java

El archivo common\_script.sh es enviado desde el plugin siempre, éste script es usado por cada archivo de ejecución para java, Python y otras tecnologías.

También se envía un archivo llamado `vpl_environment.sh`, este archivo contiene metadatos sobre la actividad. Modifique el archivo de ejecución `vpl_evaluate` y agregue al final

```
cat vpl_environment.sh
```

La salida para este comando es la siguiente

```
#!/bin/bash
export VPL_LANG='en_US.UTF-8'
export VPL_MAXTIME=240
export VPL_MAXMEMORY=268435456
export VPL_MAXFILESIZE=67108864
export VPL_MAXPROCESSES=200
export MOODLE_USER_ID='2'
export MOODLE_USER_NAME='Eliecer Alejandro Molina Vergel'
export VPL_GRADEMIN='0.00000'
export VPL_GRADEMAX='100.00000'
export VPL_COMPILATIONFAILED='The compilation or preparation of execution has failed'
export VPL_SUBFILE0='asdasd.java'
export VPL_SUBFILES="asdasd.java "
export VPL_VARIATION=''
```

Figura 44 Contenido del archivo `vpl_environment`

Como podrá observar, es un script bash común y corriente, pero exporta variables de entorno con información del estudiante, como el id y el nombre.

### *Ciclo de vida de la ejecución*

Ahora bien, la ejecución no es tan sencilla como parece, la jaula tiene unas etapas de ejecución de código cuando recibe la actividad y sus archivos de ejecución.

### *Conexión entre VPL plugin y la Jaula de Ejecución*

Al enviar la actividad y los archivos de ejecución desde el navegador hasta Moodle, el plugin de VPL envía a la jaula de ejecución una petición de ejecución con los archivos que se

ejecutarán, la jaula confirmará recibido con una url a un socket que la jaula de ejecución abre. Esta url es enviada desde Moodle VPL hasta el cliente por medio de una respuesta JSON.

Al recibir la url del socket de la jaula de ejecución, el cliente web intentará conectarse con la jaula por medio del socket, y por este medio recibir el resultado de la ejecución del código.

Esto se puede apreciar mejor en este video: <https://youtu.be/skUaW1rQd6o>

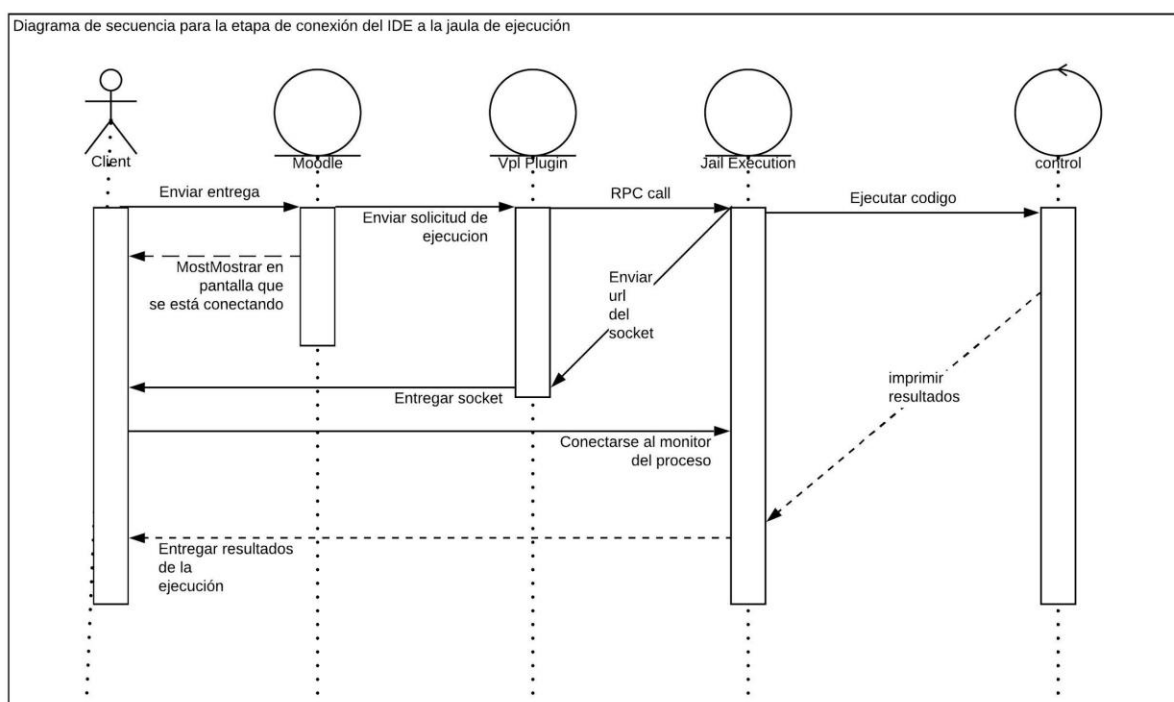


Figura 45 Diagrama de secuencia para la etapa de conexión del IDE a la Jaula de Ejecución

### *Creación de un sistema de archivos temporal para la ejecución*

Al recibir una petición de ejecución, la jaula crea un proceso y monta un sistema de archivos temporal para encapsular la ejecución y proteger la jaula (código fuente de

<https://github.com/jcrodriguez-dis/vpl-XMLRPC-jail/blob/ef6786fb203a315eb76278f444d21583afacf44a/vpl-jail-system.initd#L289>.

Este comportamiento se puede observar haciendo lo siguiente: elimine todas las líneas que agregamos al archivo `vpl_evaluate.sh` y agregue

```
pwd
```

Esta línea es un comando de UNIX para conocer la ubicación actual de la consola. Luego ejecute evaluar. Obtendrá un resultado parecido a esto



Figura 46 salida por consola al correr `pwd` en la jaula de ejecución

Observe que montó un sistema de archivos temporal en la carpeta `/home` para el proceso 12970.

### *Etapa de compilación*

Después de que la jaula de ejecución montó el sistema de archivos temporal para el proceso que ejecutará la actividad, entra en una etapa de compilación de código. Ésta etapa no hace referencia a la ejecución del programa del estudiante, en ella se ejecuta los archivos de ejecución de debug, `run` y `evaluate`, recuerde que según el caso de uso. En esta etapa **no se puede generar ni calificaciones ni comentarios** (en el capítulo dos hablamos de cómo calificar automáticamente). Pruebe lo siguiente, elimine toda línea de `vpl_evaluate.sh` y agregue: `echo "Grade :=>> 100"` luego haga clic en el botón de evaluar, obtendrá la siguiente salida



Figura 47 Salida por consola al intentar graduar en etapa de compilación

Como verá, generó solo una salida en consola, pero no se informó que se haya calificado la actividad.

### *Etapa de ejecución*

En esta etapa, la jaula de ejecución ejecuta el código del estudiante y genera las salidas requeridas por VPL para mostrar comentarios y calificar automáticamente. A la derecha del IDE, después de ejecutar evaluar, o ejecutar el programa, se muestran diferentes pestañas que informan sobre las salidas del ciclo de vida de la ejecución del programa. En la imagen anterior, se muestra solo la pestaña llamada “compilation” esto quiere decir que *Grade :=>> 100* fue la salida por consola durante la etapa de compilación.

Al finalizar la etapa de ejecución tratará de ejecutar el archivo `vpl_execution`

La jaula solo entrará en etapa de ejecución sólo si puede ejecutar un script llamado `vpl_execution`.

Por defecto este script no está creado, normalmente cuando se envían los archivos de ejecución por defecto para ejecutar, evaluar y debuggear código según el lenguaje, éstos ejecutan el archivo de ejecución `common_script.sh` y éste crea el archivo `vpl_execution`. Si el archivo `vpl_execution` existe, la jaula entra a la etapa de ejecución e intenta ejecutar el archivo `vpl_execution`.

Intente lo siguiente, elimine toda línea del archivo `vpl_evaluate` y agregue las siguientes líneas:

```
ls -la
echo 'echo "Grade :=>> 100" ' >> vpl_execution
echo 'echo "Comment :=>> este es un comentario!" ' >> vpl_execution
chmod +x vpl_execution
cat vpl_execution
```

Y haga clic en evaluar. Observará que aparece la pestaña de calificación donde se le informa al usuario que pudo calificar con un valor de 100. Además, aparece una pestaña llamada comentarios como se ve a continuación.

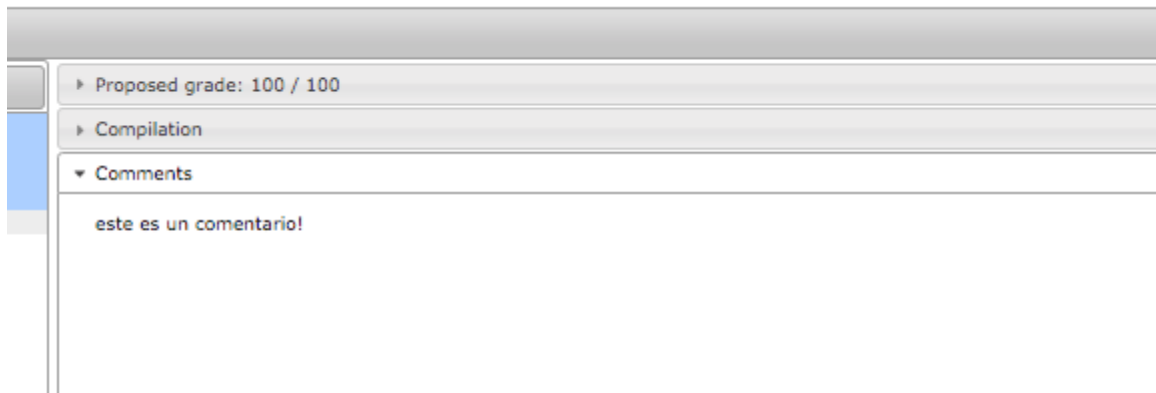


Figura 48 Impresión de un comentario desde la Jaula de ejecución

Para la etapa de compilación la salida fue:

```

▶ Proposed grade: 100 / 100
▼ Compilation

total 40
drwx----- 2 p13369 p13369 200 Jan 29 16:28 .
drwx--x--x 3 root root 60 Jan 29 16:28 ..
-rw----- 1 p13369 p13369 9285 Jan 29 16:28 CasaDeCambioTest.java
-rw----- 1 p13369 p13369 44 Jan 29 16:28 asdasd.java
-rwx----- 1 p13369 p13369 4971 Jan 29 16:28 common_script.sh
-rwx----- 1 p13369 p13369 168 Jan 29 16:28 vpl_debug.sh
-rwx----- 1 p13369 p13369 481 Jan 29 16:28 vpl_environment.sh
-rw----- 1 p13369 p13369 0 Jan 29 16:28 vpl_evaluate.cases
-rwx----- 1 p13369 p13369 106 Jan 29 16:28 vpl_evaluate.sh
-rwx----- 1 p13369 p13369 234 Jan 29 16:28 vpl_run.sh
echo "Grade :=> 100"

▶ Execution

```

Figura 49 Salida por consola de la etapa de compilación

Para la etapa de ejecución la salida fue:

```

▶ Proposed grade: 100 / 100
▶ Compilation
▶ Comments
▼ Execution

/bin/bash: warning: setlocale: LC_ALL: cannot change locale (en_US.utf8)
Grade :=> 100
Comment :=> este es un comentario!

```

Figura 50 Salida por consola de la etapa de ejecución

La siguiente imagen ayuda a comprender mejor las etapas

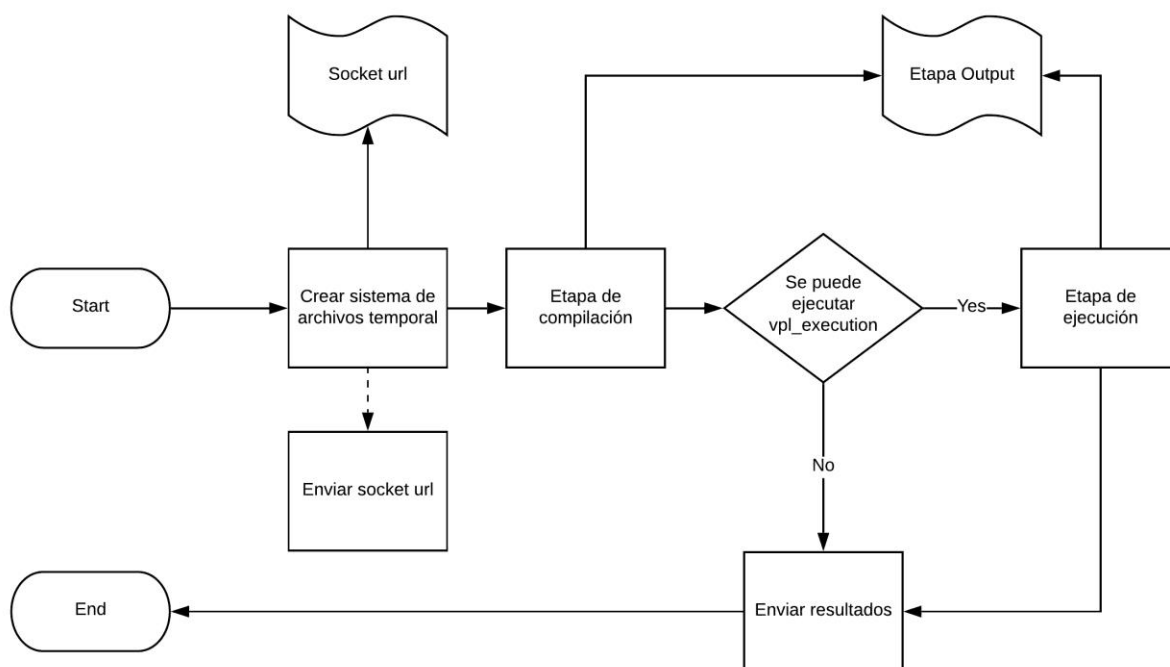


Figura 51 Diagrama de proceso de las etapas de ejecución de un programa en la Jaula de Ejecución

## 5.2.5 Problema del alto acoplamiento en la idea inicial

La identificación de los componentes y su interacción fue de suma importancia, pues permitió conocer las responsabilidades y entidades que actúan durante la ejecución de actividades VPL.

Ahora bien, existen muchas formas de extender VPL e implementar las funcionalidades deseadas, estas deben ser evaluadas a la luz de ciertos aspectos técnicos y/o métricas de



aceptación del software que de tenerse en cuenta pueden ayudar a alcanzar el éxito del mismo a través del tiempo. Se identificó dos aspectos técnicos fundamentales:

1. Facilidad de mantenimiento: La solución debe ser fácilmente mantenible. En éste aspecto no entra el mantenimiento a VPL ni a Moodle. Depende del nivel de acoplamiento de VPL++ a VPL y Moodle
2. Bajo acoplamiento: El bajo acoplamiento reduce la complejidad del mantenimiento, permite la facilidad de reutilización de componentes de software.

Durante la búsqueda de la solución más adecuada, que además satisfaga los dos aspectos anteriores se encontraron algunas dificultades. A la primera de ellas se denominó “dilema del código abierto”. Siendo VPL un software de código abierto, se corre el riesgo de que el proyecto se privatice o se abandone, ya que no está en nuestras manos darle continuidad al proyecto. Se plantearon diferentes soluciones:

Opción 1: Intentar ser un colaborador del proyecto, pero al intentar esta solución el autor del proyecto, Juan Carlos, el creador, respondió que por cuestiones laborales no estaban aceptando colaboraciones por el momento:

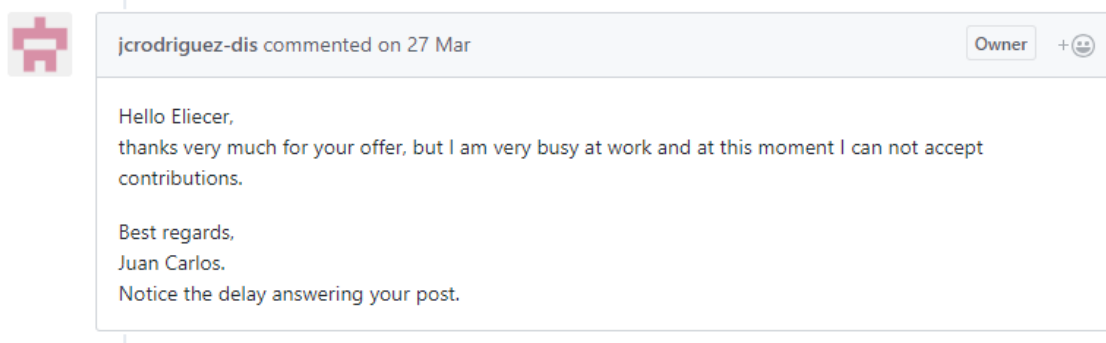


Figura 52 Respuesta de Juan Carlos a la solicitud de aportar al proyecto VPL

Se podría modificar el plugin en un proyecto local y luego actualizarlo desde el proyecto original, sin embargo, tendríamos que resolver conflictos.

Opción 2: Hacer un fork del proyecto y darle continuidad en otra línea del proyecto. Sin embargo, la Universidad deberá dar soporte a VPL y su extensión por sí misma, pues estaríamos por fuera de la línea base del proyecto. Si el programa de IS-UFPS desea actualizar VPL tendría que actualizar manualmente el fork con la línea base del proyecto y si no, quedarse fuera de las futuras actualizaciones, con todas las desventajas que esto supondría, por ejemplo, la corrección de errores o el soporte para versiones futuras de Moodle. Si la universidad decide dar soporte manualmente al fork, usando el ejemplo anterior, tendremos el siguiente flujo de eventos:

1. Se modifica el archivo A.PHP
2. La línea base genera otro release donde modifica el archivo A.PHP. Desde el fork hacemos un git pull a la línea base del proyecto
3. El encargado del mantenimiento de VPL++ resolverá conflictos.
4. El encargado del mantenimiento de VPL++ hará un commit y luego push al repositorio del fork

En este caso el problema se hubiera resuelto, sin embargo, requiere que el desarrollador que dé soporte conozca en gran medida el proyecto, VPL y Moodle para poder mantener actualizado el repositorio, es decir la universidad deberá entrenar a un desarrollador o conseguir a la persona adecuada, lo que generaría costos y esfuerzos extras, algo nada deseable.

Tabla 3 Estrategias de modificación vs esfuerzo, acoplamiento y escalabilidad

		Métricas		
Solución	Viable?	Esfuerzo de mantenimiento	Acoplamiento	Escalabilidad

Ser colaborador de VPL	no	Alto	Alto	Bajo
Fork del proyecto original sin actualizaciones	Si	Alto	Alto	Bajo
Fork del proyecto + actualizaciones manuales	Si	Alto	Alto	Bajo

Se identificó el acoplamiento como uno de los principales problemas. El alto acoplamiento entre VPL y Moodle obstaculiza su extensión, VPL no cuenta con protocolos o interfaces para ser extendido lo que lo hace aún más difícil, por lo que estaríamos a merced del proyecto original y el rumbo que tome.

### 5.2.6 Pensando “Out the box”

Out the box es una metáfora que invita a pensar diferente, no convencionalmente, tomando en cuenta diferentes aspectos, contextos y parámetros, para encontrar una solución a un problema. En el desarrollo de software existen problemas cuyas soluciones no son satisfactorias como el caso descrito en el párrafo superior. Para explorar otras soluciones se hizo el siguiente cuadro comparativo sobre los conocimientos que hasta éste punto se obtuvo.

### 5.2.7 Conocimientos aprendidos

Tabla 4 Conocimientos aprendidos

Pregunta	Si / No	¿Qué sé al respecto?
¿Conozco el problema?	Si	Enseñar y aprender programación de computadores es aún una tarea difícil

¿Definí una solución?	Si	Extender una herramienta que ayude al profesor a enfocar sus clases.
¿Es viable?	Si	Se cuenta con los recursos adecuados para el desarrollo del proyecto.
¿Incluye todos los actores?	Si	Si, profesor, estudiante, administrador del sistema y desarrollador de mantenimiento.

## 5.2.8 Alternativas de solución

Tabla 5 Soluciones encontradas

Pregunta	Si / No	Qué sé al respecto?
¿Es necesaria esta extensión?	No	La solución no necesariamente implica extender VPL
¿Trae alguna ventaja extender VPL?	SI	Si, VPL es un proyecto sólido, la universidad de las Palmas de España continúa activamente el proyecto, además la comunidad de software libre reporta y corrige frecuentemente problemas en VPL vía git pull request. Además VPL cuenta con buenas funcionalidades.
¿Conoces sus componentes?	Si	Son 4, el servidor Apache + PHP + Moodle, el servidor de base de datos, el plugin de VPL y la jaula de ejecución
¿Conoces su arquitectura?	Si	Es una arquitectura orientada a microservicios
¿Las opciones para extender VPL que identificaste son orientadas a microservicios?	No	Las opciones evaluadas no son orientadas al desarrollo de un micro servicio sino que modifica uno de los componentes de los microservicios, en este caso el plugin de Moodle VPL
¿Evaluaste una solución orientada a microservicios?	No	No, aún no.
¿Entonces no todas las opciones han sido evaluadas?	No	No, es hora de evaluar esta opción

## 5.2.9 Arquitectura orientada a microservicios

Después de haber analizado la información que se obtuvo se logró identificar que no había evaluado todas las soluciones posibles, se notó que cada componente actúa como un microservicio y cumplen con responsabilidades específicas. Por ejemplo, el microservicio web está compuesto por cuatro componentes: Apache, PHP, Moodle y VPL. Este microservicio (MS en adelante) es el encargado de visualizar e integrar los demás, éste actúa como interfaz entre el ecosistema y el stakeholder. Por otro lado, el microservicio de base de datos es el encargado de gestionar la base de datos, mientras que la jaula de ejecución es la encargada de la ejecución de los programas.

Se evaluó entonces la posibilidad de una solución orientada a microservicios que junto a Moodle extendería el proceso de ejecución y no el software existente de VPL. La experiencia laboral ayudó a idear una solución orientada a microservicios, esto fue una ventaja pues ya que se estaba familiarizado con los conceptos, la forma de desarrollar microservicios y la forma de distribuirlos. El análisis de las responsabilidades de cada uno de los componentes de software de VPL ayudó a valorar la probabilidad de una solución basada en microservicios.

Tabla 6 Microservicios y sus protocolos

Microservicio	Protocolo	Función
Web	Http	Servir la aplicación Moodle y sus plugins
Mysql	Mysql Protocol	Permitir conexiones seguras a la base de datos
Jail Execution Server	RPC	Ejecutar llamadas a procedimientos remotos
Jail Execution Server	RFC 6455	Permitir la conexiones full dúplex TCP entre el cliente y la jaula, el cliente entonces es capaz de recibir respuesta del estado de la ejecución de los programas que se envían allí.

Se centró la atención en la jaula de ejecución pues la jaula de ejecución es esclava en la relación maestro-esclavo, es decir no obedece ni actúa por sí sola, pues es el microservicio Moodle VPL es quien inicia la ejecución y la jaula de ejecución solo envía respuestas de afirmación y de estatus, ni siquiera se encarga de la ejecución del programa esa es la labor del sistema operativo y las librerías instaladas en él quien ejecuta el código recibido. En pocas palabras VPL pierde control de la ejecución del programa. Más adelante, en el capítulo de funcionamiento, incluso se observa que es el plugin de VPL de Moodle quien envía los archivos para ejecutar el código en la jaula de ejecución.

El siguiente ejemplo puede ayudar a entender el origen de ésta conclusión

El profesor Fundamentos de Programación deja como actividad de VPL escribir un programa que imprima la famosa frase “hola mundo”, la entrega de un estudiante podría ser el siguiente código (lo llamaremos hola.py):

```
print("Hola Mundo");
```

El siguiente código es un script llamado Python\_run.sh (VPL execution file), este envía las instrucciones en bash para ejecutar código en Python:

```
#!/bin/bash
# This file is part of VPL for Moodle - http://VPL.dis.ulpgc.es/
# Script for running Python language
# Copyright (C) 2014 onwards Juan Carlos Rodríguez-del-Pino
# License http://www.gnu.org/copyleft/gpl.html GNU GPL v3 or later
# Author Juan Carlos Rodríguez-del-Pino <jcrodriguez@dis.ulpgc.es>
```

```

#load common script and check programs
. common_script.sh
check_program Python
if [ "$1" == "version" ] ; then
    echo "#!/bin/bash" > vpl_execution
    echo "Python --version" >> vpl_execution
    chmod +x vpl_execution
    exit
fi
cat common_script.sh > vpl_execution
echo "Python $vpl_SUBFILE0 \@>" >>vpl_execution
chmod +x vpl_execution
grep -E "Tkinter" $vpl_SUBFILE0 &> /dev/null
if [ "$?" -eq "0" ] ; then
    mv vpl_execution vpl_wexecution
fi

```

Se espera que la ejecución del programa del estudiante generaría la salida “Hola Mundo”. VPL plugin no ejecutará irresponsablemente el código del estudiante en el mismo contexto, sino que delegará ésta responsabilidad a la jaula de ejecución. Cuando la jaula de ejecución reciba los archivos (Python\_run.sh y hola.py) creará una carpeta asignada para la actividad y el estudiante (\$vpl\_data/\$id\_actividad/\$id\_student), luego copiará los archivos allí. Estos archivos finalmente serán enviados a la jaula de ejecución mediante el protocolo XMLRPC, los recibirá y ejecutará.

El siguiente diagrama de secuencia ayuda al entendimiento de los párrafos anteriores:

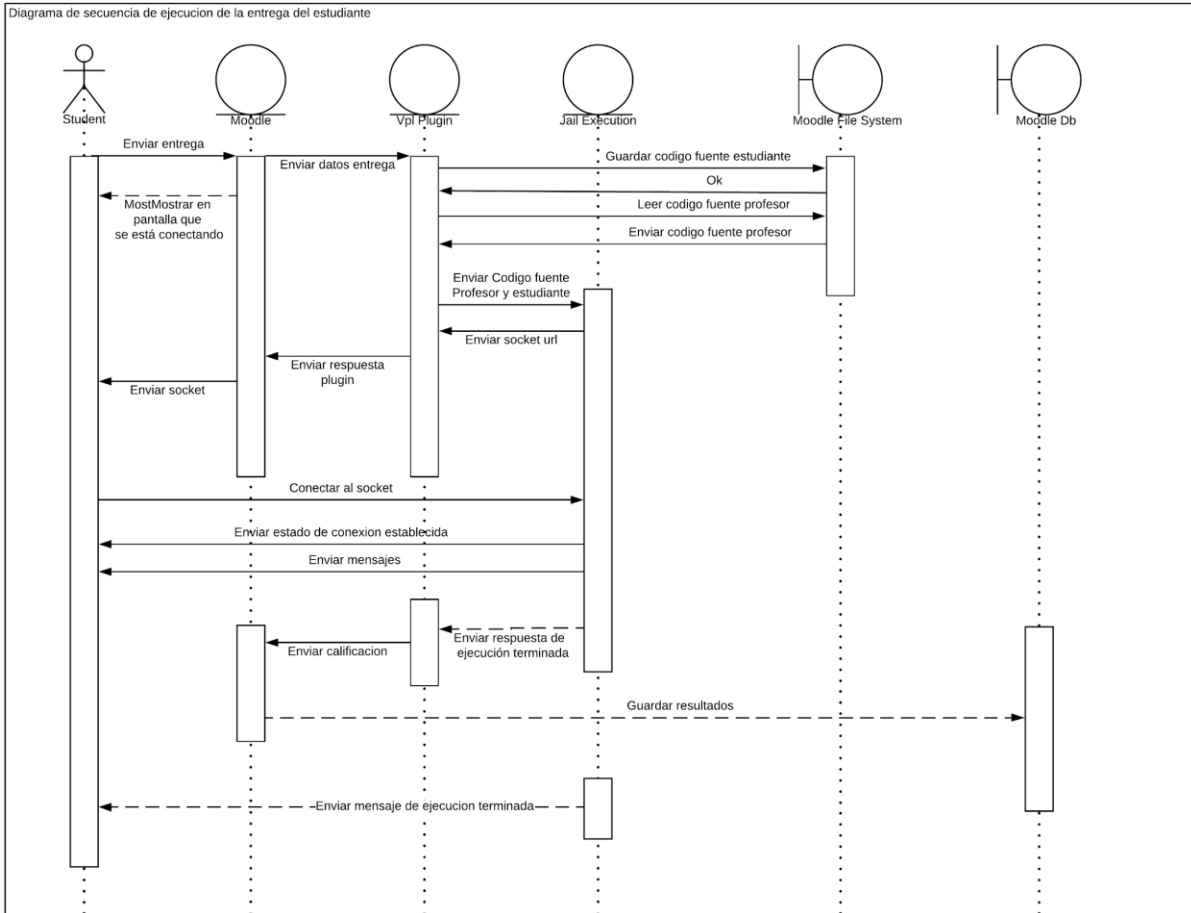


Figura 53 Diagrama de secuencia de la ejecución de una entrega de estudiante en VPL en Moodle

Internamente, los componentes de la jaula de ejecución lucen así:

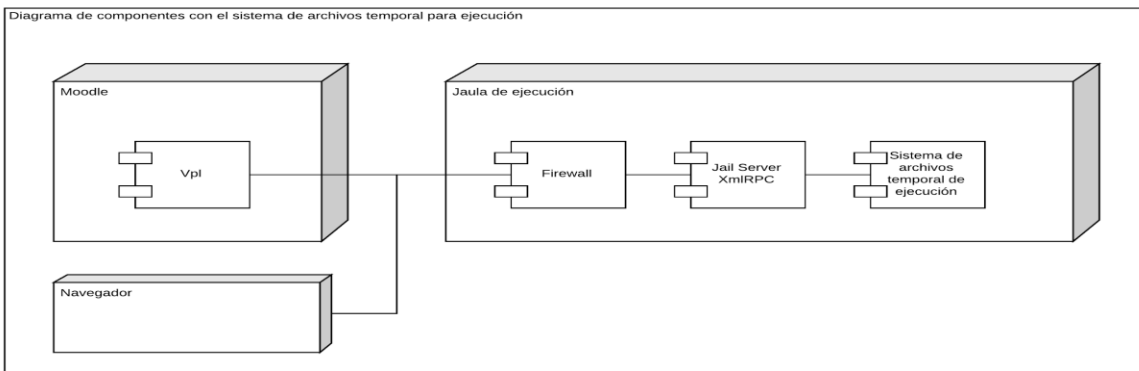


Figura 54 Diagrama de componentes de VPL con el sistema de archivos temporal



Se pudo concluir lo siguiente:

1. El servidor de la jaula de ejecución no interviene en la ejecución de los programas que se envían a él, sino que el plugin VPL envía las instrucciones y el sistema operativo junto a las librerías instaladas en él ejecutan el código que es enviado a la jaula.
2. Modificando los archivos de ejecución de VPL obtenemos control de la ejecución del programa antes que el sistema operativo compile/interprete el programa del estudiante.
3. La ejecución del programa depende del sistema operativo, si removemos java del sistema operativo donde se encuentra la jaula de ejecución, la jaula de ejecución no podrá ejecutar ningún programa en java.
4. Si añadimos una librería al sistema operativo podemos tomar control de la ejecución del programa del estudiante.
5. Esto se puede confirmar analizando el archivo de ejecución de java, éste busca en los archivos de ejecución para el programa del estudiante si incluye JUnit, si JUnit es hallado entonces en vez de ejecutar JAVA ejecuta el runner de JUnit.
6. Podemos entonces controlar la ejecución antes de la compilación/interpretación del programa de estudiante, durante y después.
7. Esperando las mismas entradas y generando las mismas salidas en la jaula de ejecución, el microservicio web no se dará por enterado que la jaula fue intervenida y no afectará su funcionamiento.

## 5.2.10 Extensión del proceso de ejecución de código en lugar de extender el código de VPL

Según las conclusiones anteriores, se desarrolló un diagrama donde se definieron algunos componentes de software:

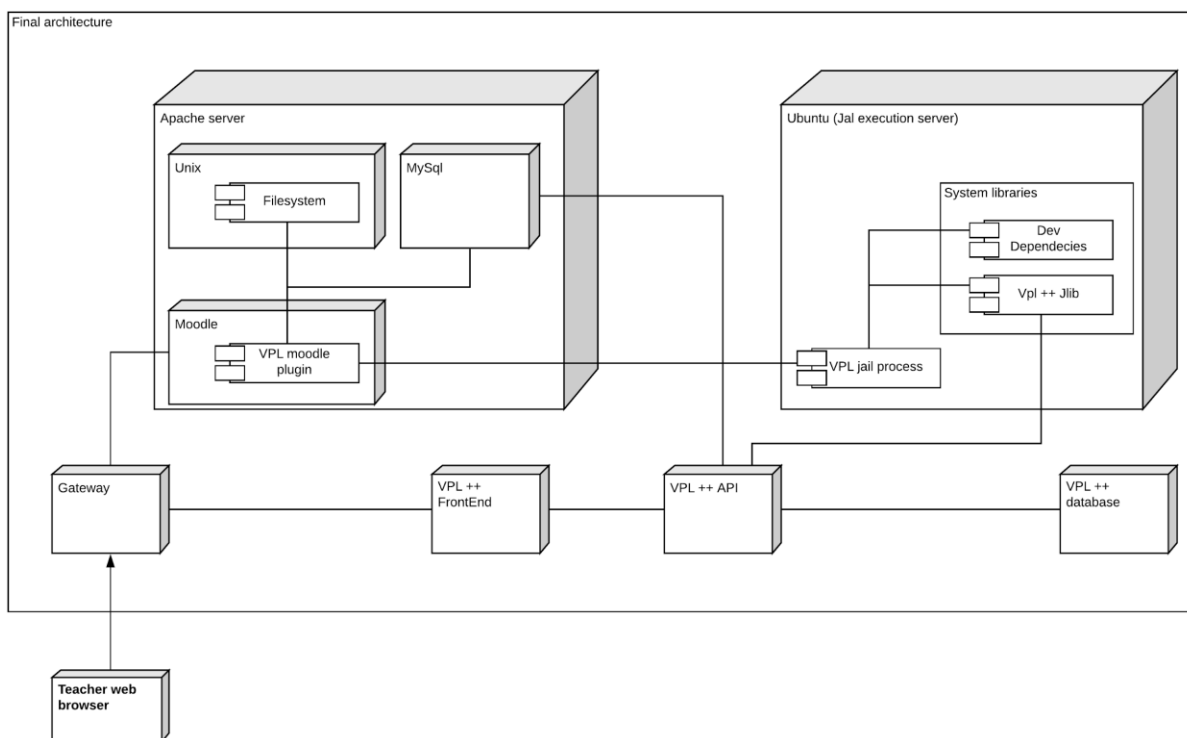


Figura 55 Diagrama de componentes de microservicios de VPL ++

Como podrá observar, no sólo limitamos la solución a un solo microservicio, sino que añadimos tres microservicios más:

1. **Gateway:** este microservicio nos dará acceso al ecosistema de microservicios.

2. VPL ++ Frontend: éste, al igual que Moodle servirá de interfaz entre VPL++ API, el frontend (FE) consumirá los servicios del VPL ++ API
3. VPL ++ API: éste actuará como backend (BE), su responsabilidad será toda la lógica de los procesos del negocio, transformación de datos, generación de reportes, etc. Además, para obtener información de credenciales, actividades, profesores y demás de Moodle, éste se conectará a la base de datos.
4. VPL ++ Database: como su nombre lo indica, éste microservicio será el responsable de la base de datos para VPL ++ (como solución general), en ella se almacenará los resultados e información relevante en el proceso de aprendizaje del estudiante.

Como resultado de ésta investigación hemos podido descubrir una tercera opción: expandir VPL usando microservicios.

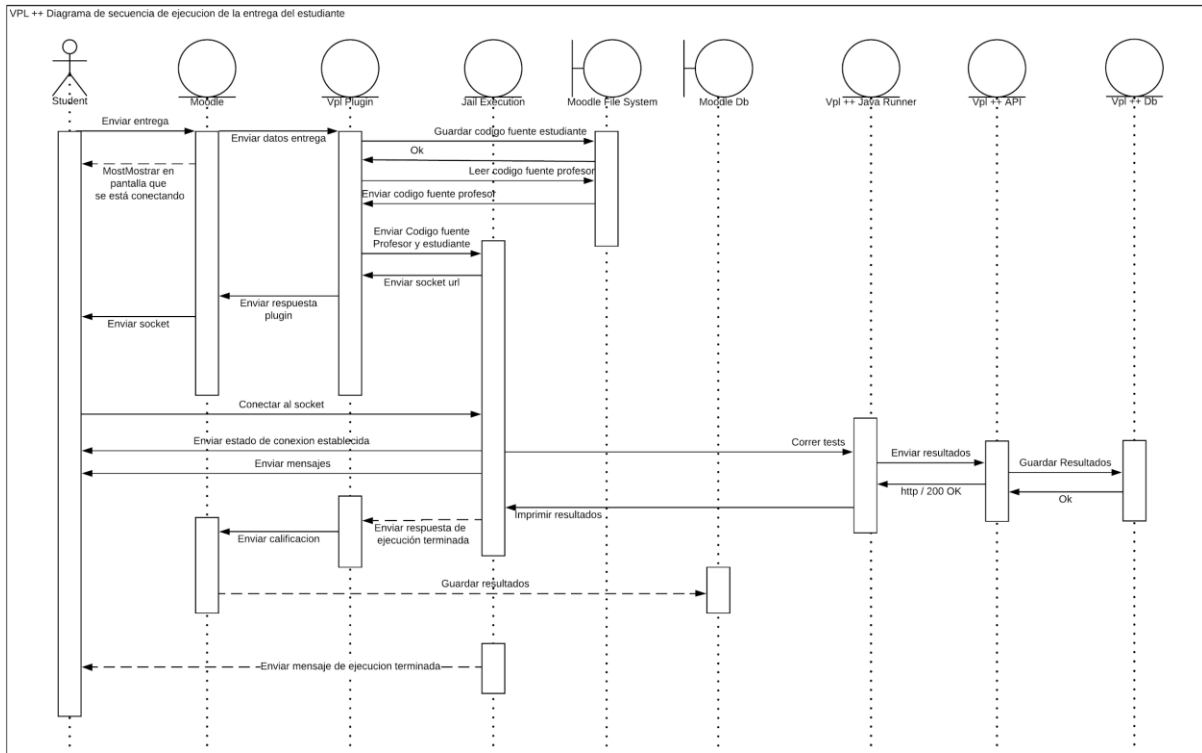


Figura 56 Diagrama de secuencia de la ejecución de una actividad de VPL usando VPL ++

## 5.2.11 Eligiendo la mejor arquitectura

Se evaluó la arquitectura orientada a microservicios ante las métricas con la que se evaluaron las opciones anteriores

### *Mantenibilidad*

Esta opción fue mucho más mantenible, como el código del proyecto VPL no será modificado, y además estará 100% desacoplado de VPL++ no se tendría problemas con el Rolling release, Moodle y VPL podrán actualizarse sin problema alguno. Por otro lado, el desarrollador que brinde soporte solo debe conocer la arquitectura lógica y el lenguaje en que esté escrito VPL ++, no será necesario que sepa PHP, bash o el framework de Moodle.

## *Escalabilidad*

Es una de las grandes ventajas de pensar orientado a microservicios, los microservicios pueden reutilizarse y añadirse a otros ecosistemas. Por ejemplo, supongamos que el profesor no desea usar Moodle, pero si VPL++, él podría descargar VPL++JLib, instalarla en su computadora, configurarla para que se conecte al servidor donde está el microservicio VPL++API, y ejecutarla sin problemas (ver anexos).

Actualizando la tabla de soluciones obtenemos que:

Tabla 7 Estrategias de modificación con microservicios vs esfuerzo, acoplamiento y escalabilidad

		Métricas		
Solución	Viable?	Esfuerzo de mantenimiento	Acoplamiento	Escalabilidad
Ser colaborador de VPL	no	Alto	Alto	Bajo
Fork del proyecto original sin actualizaciones	Si	Alto	Alto	Bajo
Fork del proyecto + actualizaciones manuales	Si	Alto	Alto	Bajo
<b>Microservices</b>	<b>si</b>	<b>Bajo</b>	<b>Muy bajo</b>	<b>Muy Alto</b>

### 5.2.12 Conclusiones de la etapa 1

1. Existe más de una manera de extender software.
2. La extensión no modificará el código de VPL plugin, la jaula de ejecución, Moodle o cualquier software usado en VPL.
3. VPL puede existir en una arquitectura orientada a microservicios

4. VPL ++ puede extender VPL sin verse afectado por Moodle o los cambios en VPL
5. Los componentes de VPL++ podrán usarse fuera del ecosistema de microservicios

## 5.3 Etapa 2: Desarrollo de software para extender VPL a VPL++

En este capítulo se describirán las etapas 2, 3 y 4 de la metodología propuesta en el anteproyecto, dado que éstas están estrechamente relacionadas.

Se obtuvo un panorama general de la aplicación, los actores, procesos y componentes involucrados; resta empezar a desarrollar el software necesario para llevar a cabo el desarrollo de la extensión de VPL, VPL ++.

El siguiente gráfico describe a grandes rasgos las diferentes actividades y sus productos ejecutadas en ésta etapa.

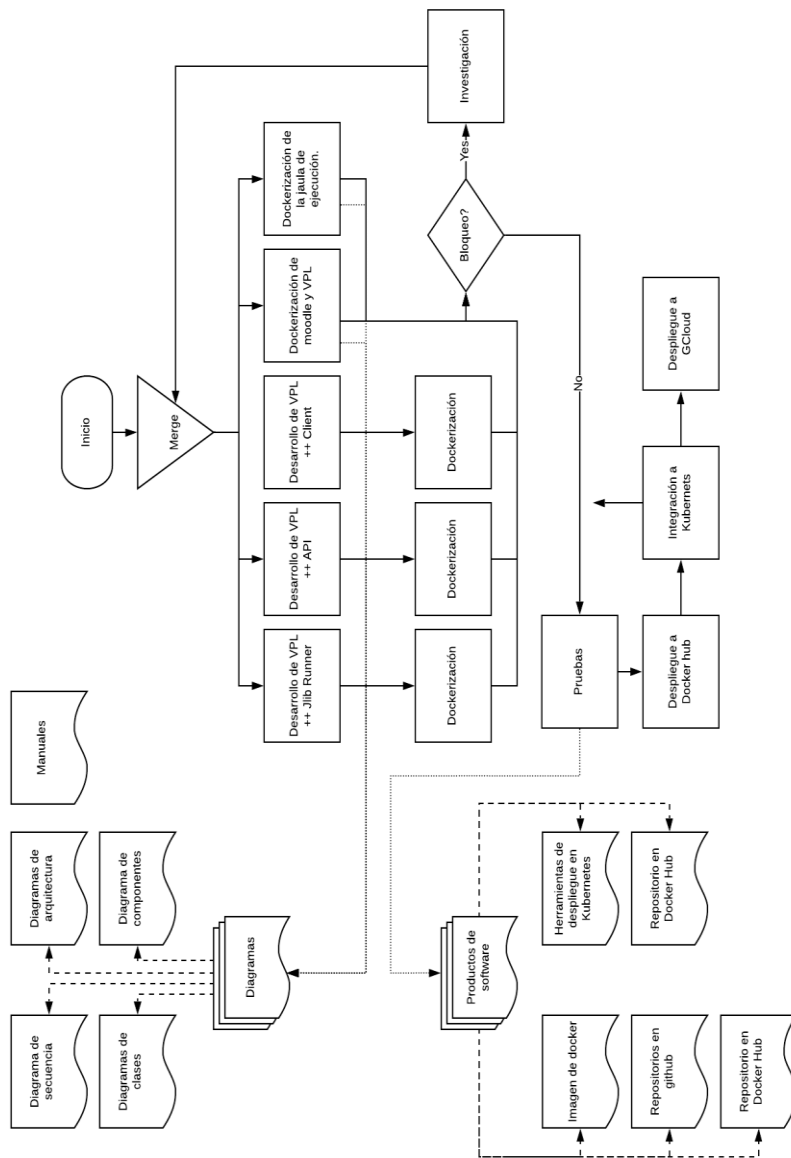


Figura 57 Diagrama de proceso de la etapa 2

### 5.3.1 Identificación de actores

Los actores son aquellos que podrán usar VPL++. Cualquiera que pueda tener acceso a Moodle podrá usar éstos componentes, en mayor o en menor medida; dependiendo de los roles que el actor posea en Moodle.

Sin embargo, los actores principales de ésta extensión son los que tengan los siguientes arquetipos en Moodle:

1. Administrator
2. Teacher
3. Editing teacher

Los demás roles podrán acceder a la extensión, sin embargo, solo verán un botón para desloguearse.

### 5.3.2 Identificación de componentes y piezas de software existentes

Antes de extender, se identificaron los siguientes componentes

#### *Moodle*

#### *Tecnologías*

1. PHP
2. Mysql
3. Apache



## *VPL Plugin*

### *Tecnologías*

1. PHP
2. Moodle interfaces
3. Moodle webservice interfaces

### *Responsabilidades*

1. Proveer funcionalidades para administrar actividades del laboratorio virtual de programación
2. Proveer un editor de código para que profesores y estudiantes puedan escribir y ejecutar código desde Moodle
3. Calificar actividades en donde los estudiantes resuelvan problemas de programación

## *VPL Jail execution server*

### *Tecnologías*

1. XMLRPC
2. C++
3. Unix
4. Iptables

### *Responsabilidades*

1. Exponer un endpoint para ejecutar código usando el protocolo XMLRPC

2. Administrar un firewall para aceptar/rechazar tráfico TCP
3. Ofrecer un ambiente seguro para ejecutar código
4. Ofrecer un socket para informar el estado de la ejecución de una porción de código

### 5.3.3 Definición de nuevos componentes y piezas de software

#### *VPL Gateway*

##### *Tecnologías*

1. Nginx
2. Alpine Linux

Este microservicio permitirá el acceso al ecosistema de microservicios que contendrá VPL y VPL++.

##### *Responsabilidades*

1. Enrutar tráfico entrante a los diferentes microservicios (reverse proxy)
2. Exponer un endpoint para que el usuario pueda acceder a Moodle, VPL++ frontend, VPL++ api
3. Balancear cargas

#### *VPL ++ frontend (VPL++ client)*

##### *Tecnologías*

1. Node Js

2. React Js
3. Redux
4. Flux
5. Babel
6. Webpack

### *Responsabilidades*

Cada actor del sistema podrá:

1. Iniciar sesión usando las credenciales de Moodle o la cuenta de google asociada a Moodle

Este componente es un cliente web, y permitirá al profesor:

1. Crear actividades de VPL

Generar automáticamente pruebas unitarias para ser usadas en las actividades creadas en VPL++

Vincular cada método de las pruebas unitarias a un tópico/habilidad

Ver el progreso de sus estudiantes respecto a un tópico/habilidad

Este componente es un cliente web, y permitirá al administrador de Moodle:

1. Crear tokens de aplicación para conectarse al API
2. Crear tópicos que el profesor podrá usar para crear sus pruebas

## *VPL ++ Backend (VPL ++ API):*

### *Tecnologías*

1. Node Js
2. Express Js
3. Mongoose
4. Babel
5. Mongo database

### *Responsabilidades*

Este microservicio es un api que dará servicios a VPL++ client y VPL JLib. Además, se comunicará con la base de datos de Moodle para autenticación y acceso al usuario de acuerdo al arquetipo del rol del usuario.

### *VPL ++ Test*

Un test de VPL++ es una prueba clásica de JUnit, que además importará el api de VPL JLib.

### *Tecnologías*

1. Java
2. JUnit

Una prueba de VPL++ se luce de la siguiente manera:

```

1 // Esta es una prueba unitaria de VPL++ Importará primero la API de vpl JLib.
2 import VPLPlusCore.annotations.VplPlusPlus;
3 import VPLPlusCore.annotations.VplTest;
4 import VPLPlusCore.annotations.VplTestCase;
5 //Además importará la API de JUnit, tal cual * se construyen las pruebas unitaras de JUnit clásicas
6 import static org.junit.Assert.assertEquals;
7 import org.junit.Test;
8 import org.junit.Before;
9 // ésta etiqueta indicará que es un test de Vpl++
10 @VplPlusPlus
11 // Esta etiqueta indicará a qué proyecto pertenece
12 @VplTest(project = "5dcdd9d9bf2717001ce0c96c")
13 public class CalculadoraTest{
14 // tal cual lo hacemos usando JUnit declaramos la clase que aplicaremos los tests
15     private Calculadora test;
16     @Before
17     public void setUp(){
18         // instanciamos la clase que probaremos
19         test = new Calculadora();
20     }
21
22 // esta etiqueta indicará a que caso de prueba pertenece un caso de prueba está vinculado a un tópico/tema/habilidad que se desea medir
23 @VplTestCase(id = "5dcdd9d9bf2717001ce0c96e")
24 // tal cual lo hacemos con JUnit agregamos la etiqueta @test que indica que ese método será ejecutado por JUnit
25 @Test()
26 public void SumarTest(){
27     assertEquals(2, test.sumar(1, 2));
28 }
29 // También podemos asignar una calificación a este caso de prueba
30 @VplTestCase(id = "5dcdd9d9bf2717001ce0c96f" , grade="1")
31 @Test()
32 public void MultiplicarTest(){
33     assertEquals(2, test.multiplicar(1, 2));
34 }
35 }

```

Figura 58 Ejemplo de una prueba unitaria usando VPL ++

## *VPL ++JLib (VPL ++ Java runner):*

### *Tecnologías*

1. Java
2. JUnit

## *Responsabilidades*

Ejecutado por línea de comandos, es un software que corre pruebas de VPL ++. Además, funciona como librería que se podrá importar desde otra clase java.

1. Expondrá una API de java que será usada en las pruebas unitarias de VPL ++. Este api proveerá decoradores a las pruebas unitarias de JUnit. Estas permitirán asociarlas a los proyectos creados por el profesor.
2. Será el encargado de ejecutar las pruebas unitarias de VPL++ en vez de usar el clásico JUnit

Este componente será instalado en la jaula de ejecución y reemplazará a JUnit al ejecutar las pruebas unitarias. Nos dará más control sobre la ejecución de las pruebas unitarias, a su vez, nos permitió separar los tests que pasaron satisfactoriamente de los que no. Finalmente esta información será enviada a VPL++ API

## 5.3.4 Tecnologías de desarrollo

Se decidió usar Docker en los diferentes componentes (actuales y nuevos), y usar Kubernetes como gestor de contenedores por las siguientes razones (Bashari Rad et al., 2017):

1. Portabilidad: una de las ventajas de usar Docker es su portabilidad, pues ésta no depende del entorno del host. Por ejemplo, los microservicios se pueden desplegar en un host Windows o Linux, o en proveedores cloud diferentes, por ejemplo: Aws, Gcloud o IBM cloud

2. Escalabilidad: Kubernetes permite escalar horizontalmente los contenedores sin mucha intervención y cuando ocurran situaciones particulares en el host o los contenedores.

3. Rendimiento: El rendimiento de Docker es más rápido que las máquinas virtuales, ya que no tiene invitados operando sistema y menos sobrecarga de recursos.(Bashari Rad et al., 2017)

4. Entrega continua: el administrador solo requerirá tener desplegada la imagen de la última versión del microservicio, con unos comandos básicos, sin importar si es Docker puro, Docker composer o Kubernetes

5. Agnosticismo de la red: Kubernetes se encarga del tráfico de microservicios que fueron automáticamente escalados. Es decir, no nos tendremos que preocupar por configuraciones de red on demand, o en proveedores de cloud.

6. Independencia de la tecnología: cada microservicio podrá ser desarrollado independientemente de la tecnología en que fue desarrollado los otros microservicios

### 5.3.5 Etapa de análisis

La etapa de desarrollo permitió conocer las limitaciones de los componentes de VPL. Estas son:

1. Limitaciones técnicas
2. Limitaciones ambientales
3. Limitaciones de networking

Aunque no todos los componentes y piezas de software tienen las mismas limitaciones, cada componente requirió su propio esfuerzo investigativo. A continuación, se expone los principales temas de investigación.

### *Protocolos de comunicación de Moodle y VPL*

Se requirió varias etapas para conocer los protocolos de comunicación Moodle con otros sistemas. El principal objetivo de ésta etapa fue conocer cómo se podría integrar los nuevos componentes a Moodle usando los protocolos de comunicación o estándares que Moodle implementa

Moodle soporta los siguientes protocolos de comunicación

1. Restful
2. Soap
3. export/import Moodle backup files

También soporta los siguientes estándares de aprendizaje electrónico:

1. Scorm
2. Lti

VPL también soporta:

1. Restful para web services



Se descartó el uso de webservices dada la precariedad de la documentación y el poco control del servicio web sobre Moodle. También se descartó usar estándares como SCORM o LTI por su complejidad (ver recomendaciones).

Con esta investigación se descubrió:

2. Conocer que cuando se exporta una actividad de Moodle, éste genera un archivo comprimido con varios archivos en XML que definen las propiedades de la actividad exportada.
3. Moodle no genera algún tipo de seguridad que valide ésta actividad, sus propiedades, etc.
4. Fue posible modificar éste comprimido y sus archivos; de manera que se pudo crear una nueva actividad recreando el comprimido.

Finalmente se decidió que la mejor opción para añadir una actividad creada por VPL++ era crear una actividad falsa de backup y luego restaurarla desde Moodle.

### *Reflexión en java*

Se requirió investigar en la documentación oficial de JUnit para conocer lo siguiente:

Requerimientos:

1. Se requirió conocer el proceso de compilación de las clases y la ejecución de las pruebas unitarias usando JUnit

Conocimiento adquirido:

Para ejecutar las pruebas unitarias se debe compilar (es decir generar el archivo .class) las clases que se desean probar y su prueba unitaria.

Por línea de comandos se especifica la ubicación de estas clases y se ejecuta JUnit.

2. Se requirió conocer cómo se puede extender JUnit

Conocimiento adquirido:

Para extender JUnit se debe usar la API de JUnit en java.

Para esto se importa la clase JUnitCore, ésta provee un método estático que permite correr las clases de las pruebas unitarias. Link de referencia:

<https://junit.org/junit4/javadoc/latest/org/junit/runner/JUnitCore.html>

Existen otras clases de la API de JUnit para java que fueron usadas como, por ejemplo:

Result: clase que abstrae los resultados de la ejecución de las pruebas unitarias. Link de referencia:

<https://junit.org/junit4/javadoc/4.12/org/junit/runner/Result.html>

Failure: clase que abstrae una prueba que falló. Link de referencia:

<https://junit.org/junit4/javadoc/4.12/org/junit/runner/notification/Failure.html>

Description: clase que abstrae un método de la prueba unitaria. Link de referencia: <https://junit.org/junit4/javadoc/4.12/org/junit/runner/Description.html>

3. Se necesitó conocer cómo añadir clases dinámicamente al contexto de ejecución de VPL++ JLib

Conocimiento adquirido:

Se conoció sobre reflexión en java. En informática, reflexión (o reflexión computacional) es la capacidad que tiene un programa para observar y opcionalmente modificar su estructura de alto nivel.

Se conoció que Java provee un api para hacer reflexión y cargar dinámicamente clases al contexto (classpath) donde se ejecuta VPL JLib. La clase

java.net.URLClassLoader permite cargar archivos .class desde una url. Después de cargar los tests como archivos .class, se envían al core de JUnit para que se ejecuten

## *Dockerización de Moodle*

Se requirió conocer como Moodle se instala, que dependencias requiere del sistema operativo y de PHP. Además, también se necesitó conocer la estructura lógica, ya que Moodle almacena algunas cosas en archivos, como por ejemplo los plugins.

## *Estructura de la base de datos de Moodle*

Se requirió una etapa de investigación de la estructura de la base de datos de Moodle, para poder hacer consultas sobre información de las actividades (módulos de curso), contextos, profesores, estudiantes de profesores, cursos a los que puede acceder el profesor, entre otros.

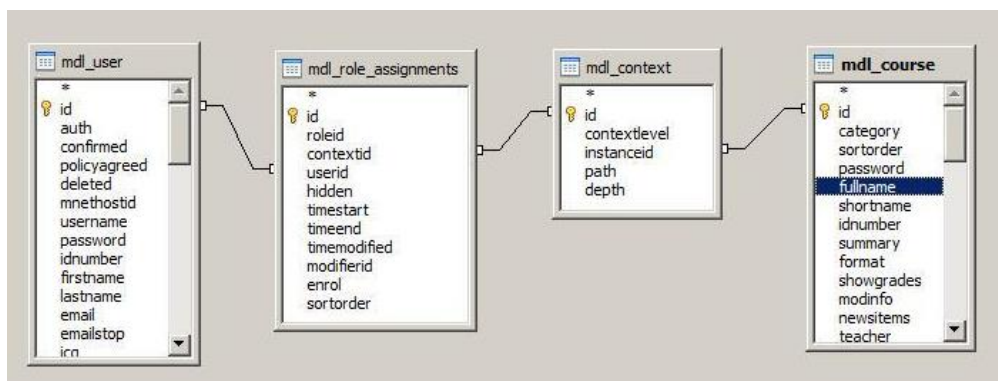


Figura 59 Relación de tablas para obtener el curso al cual está asignado una persona en Moodle

## *Dockerización de la jaula de ejecución*

Se requirió una etapa de investigación de Docker, ya que muchos de los componentes tienen requerimientos especiales para funcionar. Se requirió mucho esfuerzo para Dockerizar la jaula de ejecución exitosamente. Sin embargo, nos permitió conocer un poco más de ella. Se descubrió que:

1. La jaula de ejecución requiere capacidades especiales para ejecutar la jaula de ejecución. En Docker se puede usar la opción `CAP=ALL` para proveerles al contenedor.
2. Docker no virtualiza completamente el contenedor, es imposible usar `systemctl` o cualquier administrador de servicios de Linux dentro del contenedor.
3. El script que inicia la jaula de ejecución requiere usar un administrador de servicios de Linux, al estar en Docker en el servicio de la jaula nunca es ejecutado.
4. Es necesario iniciar manualmente la jaula de ejecución con el siguiente comando: `etc/VPL/VPL-jail-system start` (`etc/VPL` es el directorio donde se instaló la jaula de ejecución)
5. Al recibir una solicitud de ejecución de código, la jaula monta un sistema de archivos solo para éste proceso.
6. Cada sistema de archivos no es persistido y los archivos ejecutados en el desaparecen.
7. Si se requiere añadir scripts, dependencias o cualquier otra forma de información durante la ejecución de una solicitud de ejecución de VPL, se debe añadir al esquema del sistema de archivos. Este esquema se puede encontrar en la ruta `/jail`

## 5.3.6 Etapa de desarrollo, test e integración

El siguiente diagrama muestra las etapas de desarrollo:

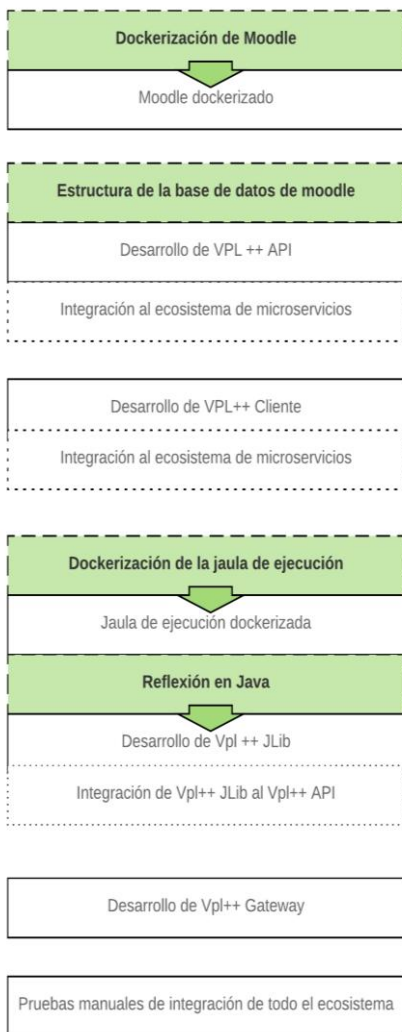


Figura 60 Distribución de la etapa de investigación, desarrollo y testing durante la etapa 2 del desarrollo del proyecto

Cada etapa de investigación está señalada con un color y su contorno son líneas cortadas. Las etapas de desarrollo tienen un fondo claro y líneas continuas. Las etapas de integración tienen fondo blanco y líneas punteadas.

### 5.3.7 Estrategia de desarrollo

A pesar de que se apropió Moodle y VPL, se necesitaron varias etapas de investigación. Estas etapas fueron iterativas a medida que cada microservicio y componente de software fue desarrollado, probado e integrado.

El siguiente diagrama muestra el ciclo de vida

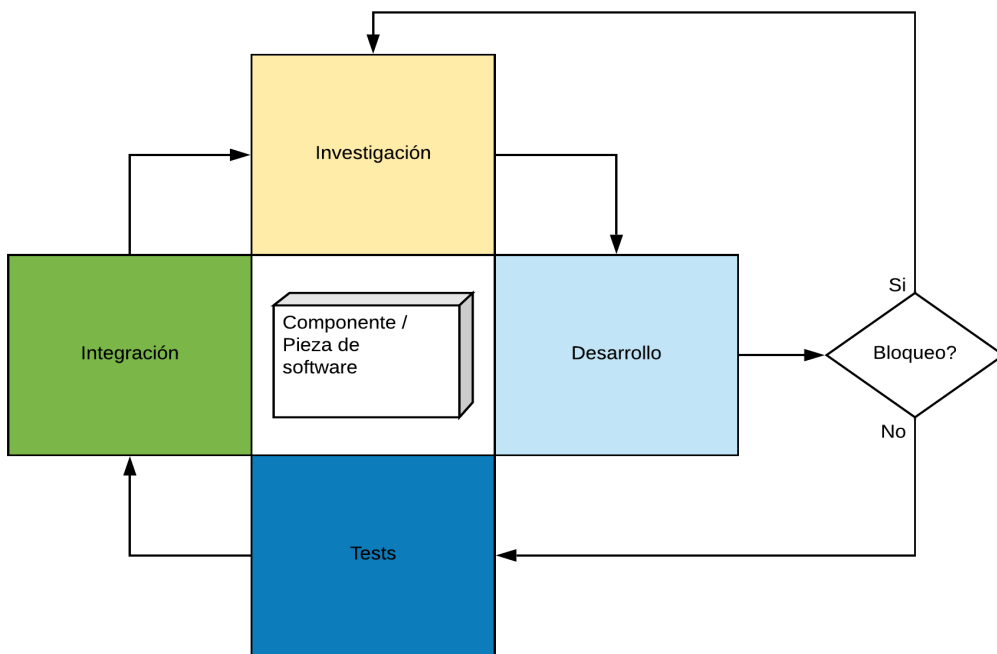


Figura 61 Estrategia de desarrollo

A pesar que la metodología no es estrictamente en cascada, hubo momentos durante el desarrollo de cada componente o pieza de software que se desarrolló de esa manera, ya sea por nuevos requerimientos técnicos, incertidumbre, desconocimiento o bloqueos que obligaron varias veces a repetir la etapa de investigación.

## 5.4 Funcionamiento de VPL ++

El siguiente diagrama describe el proceso desde que el profesor crea una actividad de VPL ++ y los estudiantes terminan de resolverlo.

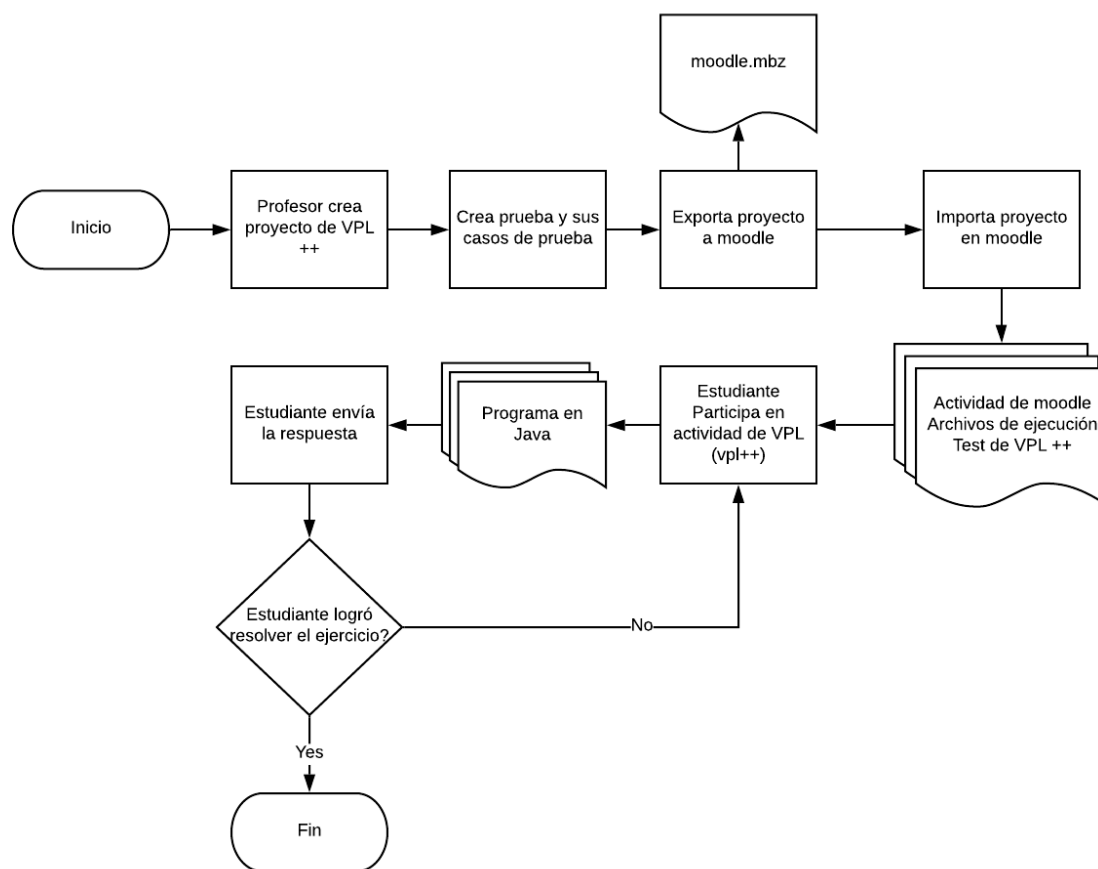


Figura 62 Diagrama de proceso del ciclo de vida de la ejecución de un programa usando VPL ++

### 5.4.1 Terminología

1. VPL: son las siglas de Virtual Programming Lab
2. VPL ++: Siglas de Virtual Programming Lab ++, nombre del producto de esta tesis

3. Proyecto de VPL ++: un proyecto corresponde a una actividad de Moodle VPL, este contiene pruebas que a su vez contienen casos de pruebas
4. Prueba: corresponde a una clase de Java que hace uso de JUnit, y cada método tiene un decorador @test
5. Caso de prueba: dentro de una prueba, un caso de prueba es un método que implementa el decorador @test
6. Tópico: hace referencia a un tema en específico, por ejemplo “Manejo de operadores aritméticos”. El nivel de especificidad depende del profesor y es algo que el profesor desea evaluar.
7. Nivel de Habilidad: es un valor comprendido entre cero y cien. Este valor indica que tan habilidoso es un estudiante en cierto tópico.
8. Esfuerzo: es el número de entregas que necesitó un estudiante enviar para pasar un caso de prueba
9. Coeficiente de penalización: es una variable usada para calcular el nivel de habilidad, éste es un número mayor o igual a uno. La penalización es 1 si el estudiante logró pasar todos los casos de prueba respecto a un tópico, y mayor que uno dependiendo de cuántos casos de prueba no logró pasar respecto a la cantidad de casos de prueba que logró pasar para un tópico. La penalización es directamente proporcional al número de casos de prueba no pasados respecto a un tópico.
10. Archivo de ejecución: Son archivos que VPL envía a la jaula de ejecución para controlar la ejecución de un código
11. Casos de prueba difíciles: son aquellos que requieren más esfuerzo para resolverse.
12. VPL ++ JLib Runner: Importado desde una clase de Java, ofrece una serie de decoradores para construir pruebas de VPL ++. Además, es un software para ser instalado en la jaula de ejecución de VPL. Éste es una implementación del runner de JUnit, de manera que VPL ++ toma el control de la ejecución de las pruebas unitarias.



13. VPL ++ es un conjunto de elementos de software que funcionan entre sí para ejecutar pruebas unitarias, almacenar los resultados y sacar reportes. El proceso empieza desde que el profesor crea un proyecto de VPL ++.

El siguiente diagrama muestra el proceso de nacimiento y ejecución de las pruebas de VPL ++

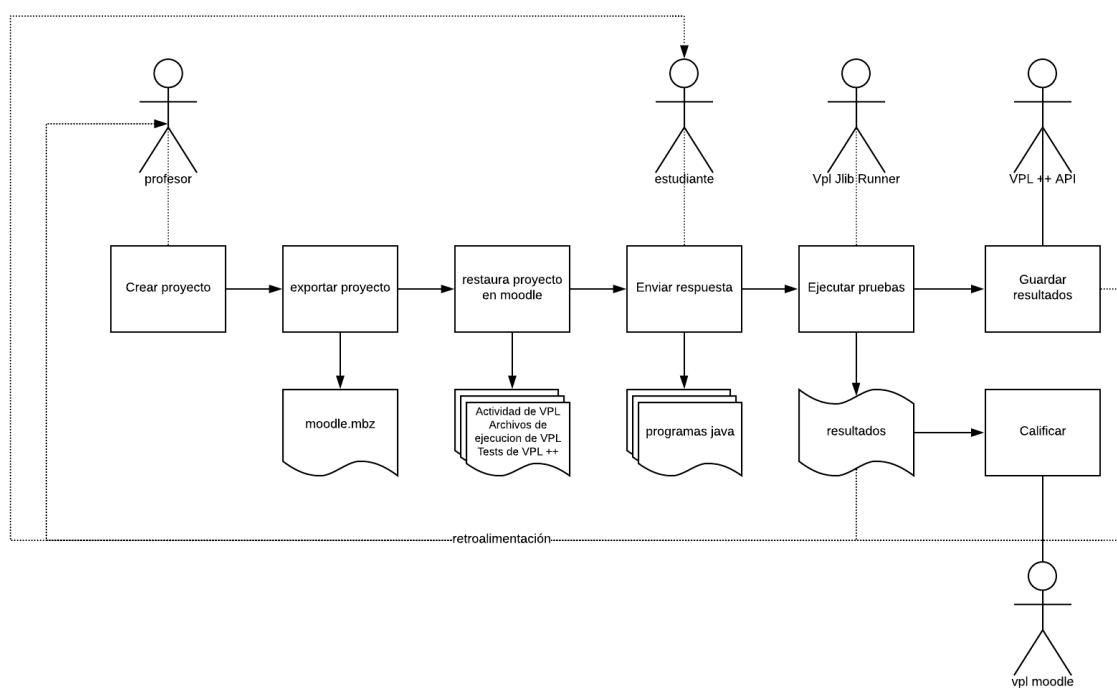


Figura 63 Proceso de nacimiento y ejecución de las pruebas de VPL ++

## 5.4.2 Proyectos

Un proyecto de VPL ++ genera un archivo de restauración de Moodle, y este es restaurado en Moodle para crear la actividad de VPL.

La actividad de VPL generada por VPL ++ es el conjunto de archivos de ejecución de VPL y de unas pruebas unitarias, que implementan la librería de VPL++ para java y que tiene uno o varios casos de pruebas. Esta librería es ofrecida por VPL ++ JLib Runner y es un superset de JUnit 4.

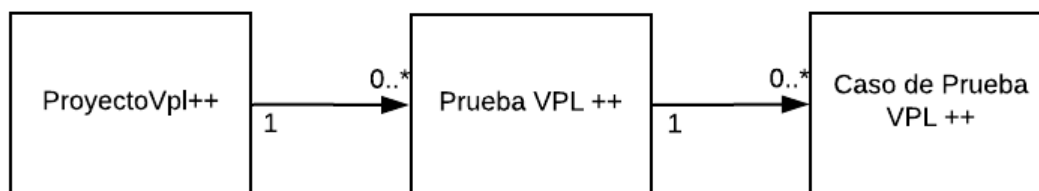


Figura 64 Diagrama de clases simple que explica la relación entre proyectos, pruebas y casos de pruebas de VPL++

Cada prueba de VPL ++ es simplemente una clase de java que implementa los decoradores @VPLPlusPlus y @VPLTest de VPL ++ JLib. Éstas clases además tienen métodos que implementan el decorador @test de JUnit y el decorador @VPLTestCase de VPLJlib.

VPL ++ VPLJlib Runner omitirá los tests que no implementen en sus clases las etiquetas de @VPLPlusPlus y @VPLTest y los métodos que no implementen del decorador @test de JUnit y el decorador @VPLTestCase

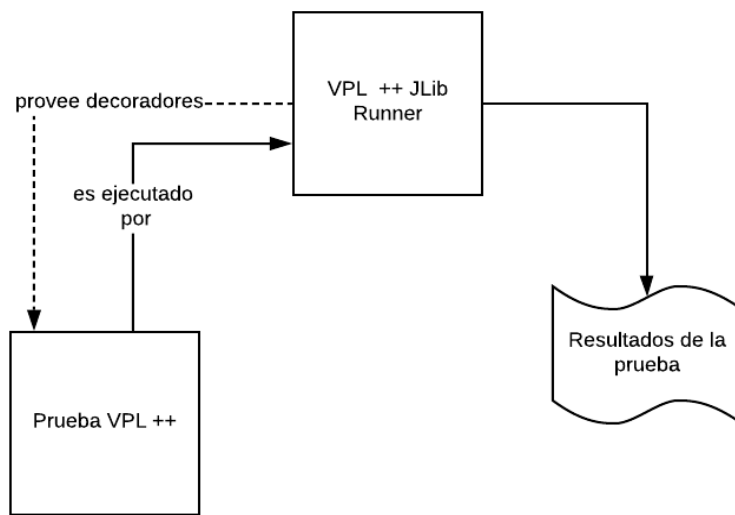


Figura 65 JLib como librería y como software

La siguiente imagen muestra como una prueba de VPL ++ es creada.

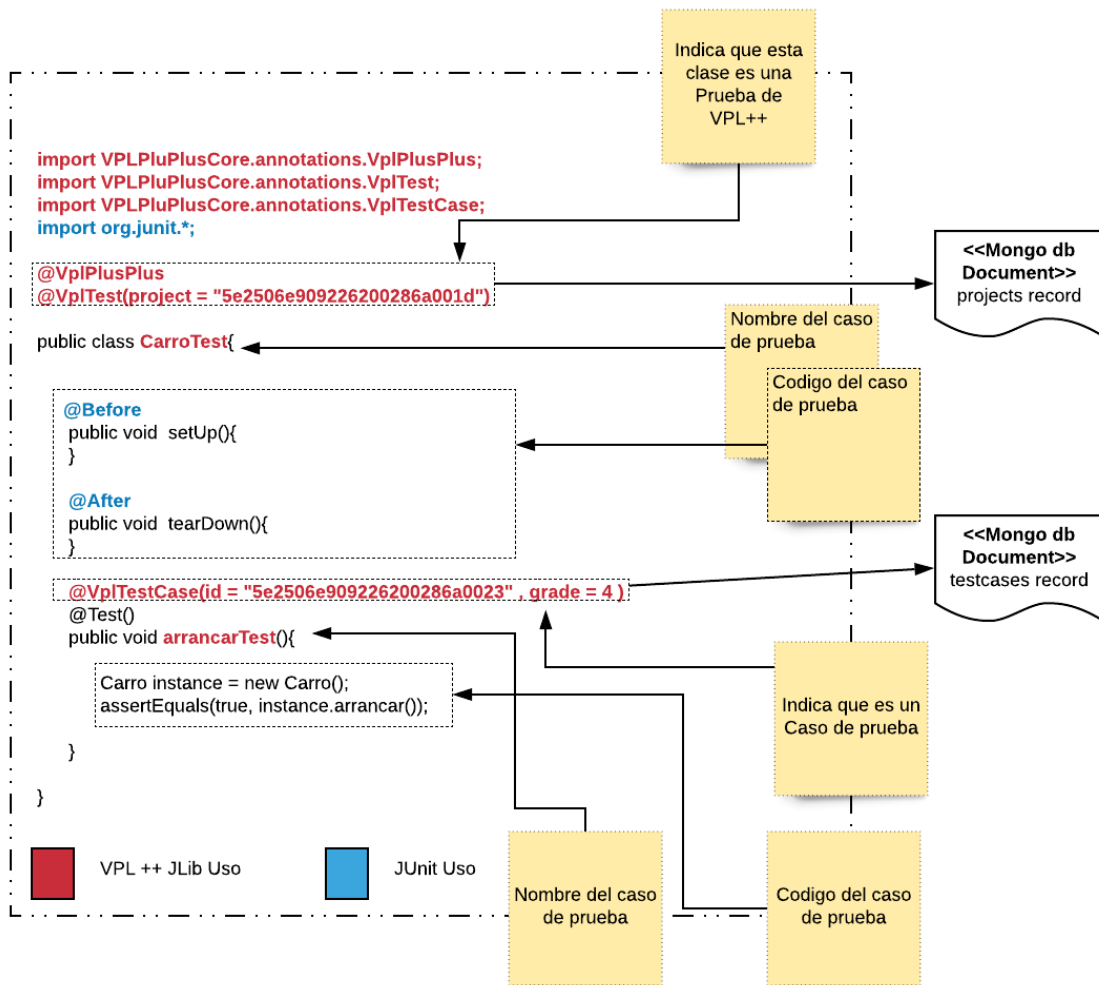


Figura 66 Explicación de un test de java de VPL ++

Los decoradores de JLib permiten al JLib Runner identificar el proyecto, y el caso de prueba a la que pertenece en la base de datos. De esta manera cuando se ejecuten las pruebas de VPL++ podrá relacionar cada prueba a un proyecto y cada caso de prueba a un caso de prueba en específico en la base de datos.

### 5.4.3 Pruebas

Como hablamos anteriormente, una prueba de VPL ++ es una clase que implementa ciertos decoradores de VPL JLib Runner.

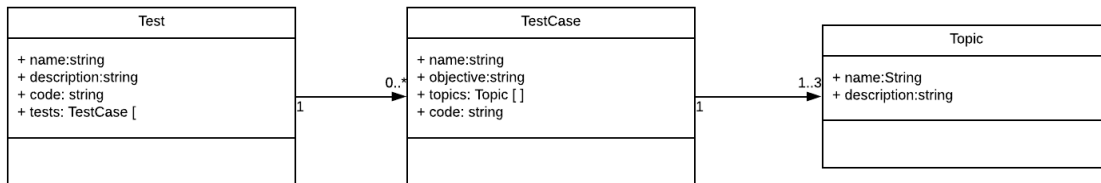


Figura 67 Diagrama de clase simple para explicar la relación entre caso de prueba y tópico+

### 5.4.4 Casos de prueba y tópicos

Los casos de pruebas son métodos de las pruebas de VPL ++. Como se dijo arriba, un caso de prueba implementa el decorador @VPLTestCase de VPL ++ JLib Runner.

Un tópico es un objetivo específico, el profesor usará un caso de prueba para ver si se cumple el objetivo específico. Si se cumple, el profesor dirá que el estudiante conoce herencia, si no se cumple el profesor dirá que no conoce herencia

Cada caso de prueba está vinculado a varios tópicos (máximo 3), esto permite a VPL ++ generar reportes especializados sobre objetivos específicos.

El siguiente test verifica que una instancia de la clase Carro sea heredada de Vehículo

```
@test
public void testHerenciaVehiculo(){
    Carro instance = new Carro();
    assertThat(instance, instanceOf(Vehiculo.class));
}
```

Con este caso de prueba el profesor probará si el estudiante conoce o no herencia. Si falla, el estudiante no conoce herencia, pero si se ejecuta, el profesor sabrá que el estudiante conoce herencia.

## 5.5 Ecuación matemática para medir el rendimiento de los estudiantes

El principal objetivo de VPL ++ es evaluar el rendimiento de los estudiantes, sin embargo, no se encontró una metodología de evaluación del rendimiento de un estudiante en programas de computer science o de ingeniería de sistemas.

Algunas estrategias encontradas fueron:

1. Evaluación ordinaria por evaluación automática de los programas de los estudiantes usando varias herramientas (Edwards, 2003).
2. Comparación de los resultados de pruebas automáticas de los programas de los estudiantes a través del tiempo usando diferentes lenguajes de programación (Parham, 2003).

### 3. Comparación de los resultados de los estudiantes según el día de entrega (Edwards, 2003)

Por otro lado, otro autor hizo un análisis probabilístico y asegura que la evaluación del desempeño del estudiante es una ciencia inexacta (Heines, 1999).

Las anteriores estrategias no se ajustaban a los requerimientos: objetividad y agnosticismo por la tecnología de aplicación. Se desarrolló una ecuación matemática para obtener el nivel de habilidad de uno o varios estudiantes, de acuerdo a las siguientes variables:

Tabla 8 ecuación matemática y sus variables para medir el nivel de habilidad de un estudiante

Nombre de la variable	Descripción
s	Entregas del estudiante por cada caso de prueba
a	Intentos para resolver cada caso de prueba. Por ejemplo, si el estudiante falla el caso de prueba 1, y en su siguiente entrega lo resuelve, la cantidad de intentos para resolver el caso de prueba 1 fue dos (2)
T	Total casos de prueba
R	Total casos de prueba que el estudiante logró resolver
N	Total, casos de prueba que el estudiante no logró resolver  Ecuación 1 casos no resueltos  $T - R$
C	Coeficiente negativo, este coeficiente castiga al estudiante según la cantidad de casos de prueba que NO logró resolver. Es directamente proporcional a los casos de prueba que no logró resolver

	<p>Ecuación 2 Coeficiente negativo</p> $\frac{T + 1}{R + 1}$
E	<p>Esfuerzo, es el total de intentos para resolver una prueba</p> <p>Ecuación 3 esfuerzo</p> $\sum_{i=n}^{nTestCases} a$
S	<p>Nivel de Habilidad del estudiante.</p> <p>Ecuación 4 Nivel de Habilidad</p> $\frac{R}{E * C}$

La variable S representa el nivel de habilidad del estudiante, tomando como input el número de casos de prueba, la cantidad de esfuerzo y sus casos resueltos.

Se decidió no incluir el tiempo de entrega entre los factores, para no castigar a los estudiantes que deseen tomar más tiempo para pensar y poder responder el problema.

Si el profesor lo desea, puede usar la fórmula para medir a los estudiantes manualmente. Se obtienen resultados interesantes.

Los rangos de la función son:



1. Valores válidos de T:  $T \geq R \ \& \ T \geq N \ \& \ T > 0$
2. Valores válidos de R:  $T \geq R \geq 0$
3. Valores válidos de N:  $T \geq N \Rightarrow 0$
4. Valores válidos de C:  $C \geq 1$
5. Valores válidos de E:  $E \geq R \Rightarrow 0$
6. Valores válidos de S:  $1 \geq S \geq 0$

## 5.6 Seguridad

La capa de datos de VPL ++ está dividida en dos partes:

1. La capa de datos de Moodle
2. La capa de datos del API de VPL ++

El API de VPL ++ se encarga solo de administrar los datos de su propia base de datos, el API jamás escribirá en la base de datos de Moodle. La razón de esto es delegar la seguridad de los datos del estudiante a Moodle y su sistema de roles, permisos y contextos.

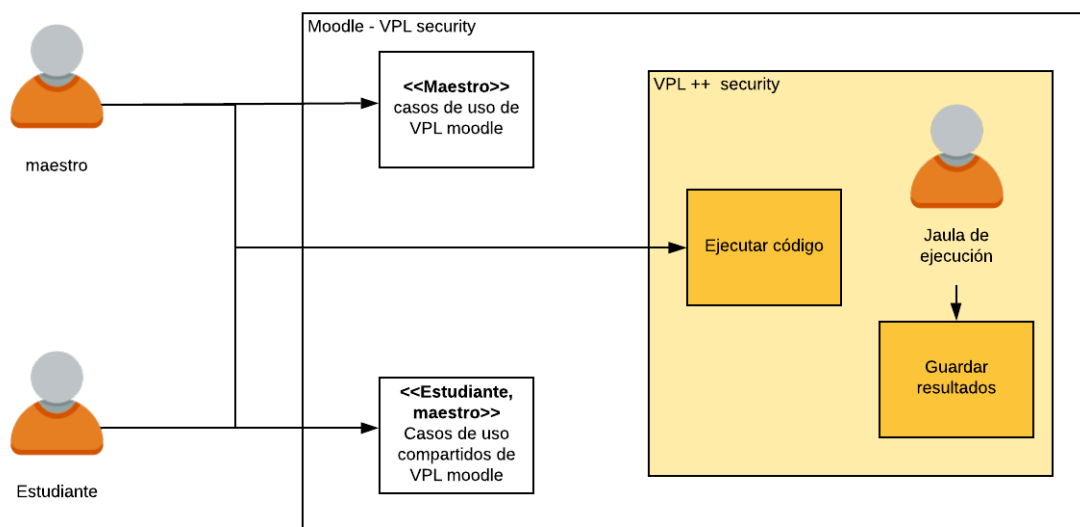


Figura 68 Seguridad de VPL ++

Como podrá ver en la imagen anterior, para ejecutar el código sobre VPL ++ deberá primero atravesar el sistema de permisos de Moodle y VPL.

Por otro lado, la capa de seguridad de VPL ++ API usa políticas y grupos de políticas para el acceso de control de recursos. Estas políticas refuerzan a los casos de uso, previniendo ejecuciones mal intencionadas de código.

### 5.6.1 Recursos

Un recurso es una porción de información o una funcionalidad a la que puede acceder un usuario. Los recursos pueden expresarse con una cadena de texto que se puede dividir en 4 partes por partes mediante un token separador.

Por ejemplo

*service:component:module:action*

Partes:

1. Service: hace referencia al servicio a la que el usuario puede acceder. Normalmente hace referencia a un microservicio, pero también podría ser el nombre del sistema operativo del API.
2. Component: hace referencia a un componente del servicio.
3. Module: hace referencia a un módulo del servicio.
4. Action: Hace referencia a una acción del servicio.

Cada una de las partes puede ser cualquier cadena, sin embargo, toma significado cuando se procesa, por ejemplo, la cadena *client:web:system:webclient.logout.show* hace referencia al microservicio client y el componente web, el action a logout esto nos dice que se trata de una política para el frontend de VPL ++ sobre la acción de cerrar sesión.

## 5.6.2 Políticas

Una política dice como se debe acceder a un recurso, una política está asociado solo a un recurso.

Una política luce, por ejemplo:

```
{
  "extends": [],
  "resource": "service:api:system:token.list",
  "name": "token.list",
```

```

"slug": "List tokens",
"type": "default",
"description": "Policy for list an application token",
"actions": [
  {
    "scopes": [
      "listToken"
    ],
    "path": "GET/api/v1/token/:id?"
  }
],
}

```

1. Extends: política de la cual quisiera extender, si una política extiende a otra, ésta toma los mismos permisos sobre el recurso
2. Resource: string que hace referencia al recurso
3. Name : nombre de la política
4. Slug: Nombre amigable de la política, ideal si deseamos tener un sistema de administración de políticas y recursos
5. Type: tipo de política
6. Description: al igual que slug, es para ser más amigable con el usuario si deseamos tener un sistema de administración de políticas y recursos
7. Actions: es un array de acciones

### 5.6.3 Acciones

Una acción es la nomenclatura formal del ámbito de un caso de uso. Sus partes son:

1. Scope: ámbito de la acción

## 2. Path: ruta o endpoint para acceder a la acción

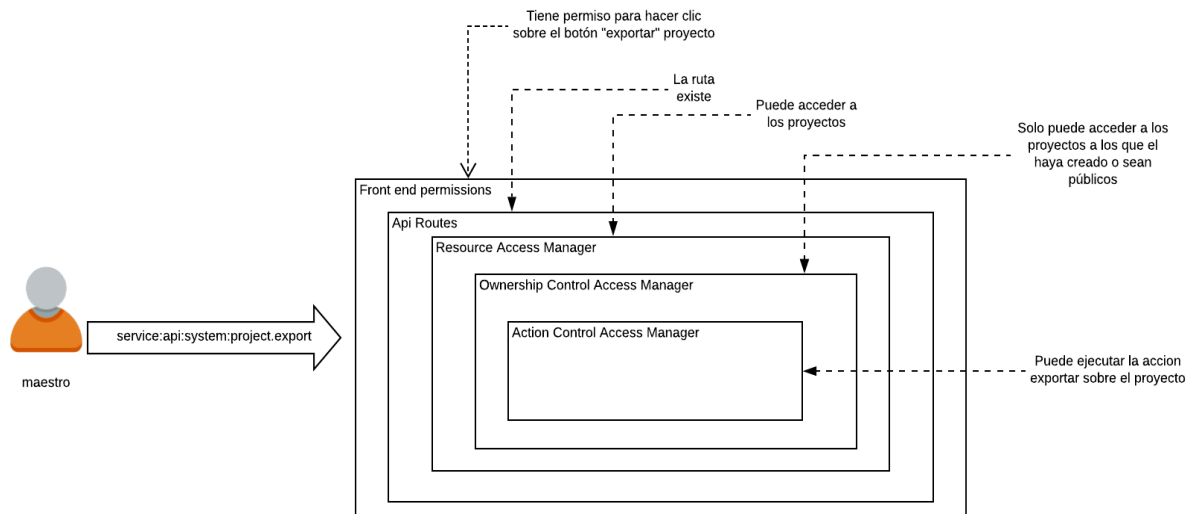


Figura 69 Relación entre políticas de acciones sobre recursos.

### 5.6.4 Roles y grupos

VPL ++ No tiene un sistema de roles sino de grupos. Un grupo asocia múltiples políticas a un usuario. El API obtiene los roles de un usuario desde la base de datos de Moodle, y a cada rol se le asocia un grupo de políticas, cada grupo de políticas depende de un caso de uso para cada actor de VPL ++.

## 5.7 Arquitectura

### 5.7.1 Orientada a microservicios

La arquitectura orientada a microservicios, a diferencia de las arquitectura monolítica, sus componentes no se despliegan de un mismo aplicación, servidor, repositorio o contexto (Ren et al., 2018). Algunas características de esta arquitectura son:

1. Cada microservicio se despliega independientemente
2. Tienen un punto único de entrada (Gateway)
3. Respetan el principio de una sola responsabilidad.
4. Cada microservicio puede operar bajo su propio lenguaje de programación.
5. Usan un protocolo de comunicación común.
6. La falla de uno no involucra la falla de todo el sistema.
7. Existen en contextos auto contenidos, como máquinas virtuales, o imágenes de instancias virtualizadas.
8. A nivel de proyecto, cada microservicio es independiente.
9. Se enfocan en el proceso de desarrollo individual.

Cada uno de los microservicios del ecosistema puede ser extendido, ser desplegado y funcionar de manera independiente (Cerny et al., 2018), el microservicio Gateway es el punto de entrada, cada microservicio tiene una única responsabilidad, están construidos en diferentes lenguajes, y usan http como protocolo de comunicación interno.

### 5.7.2 S.O.A (Service Oriented Architecture)

La arquitectura orientada a microservicios es similar S.O.A, ésta se diferencia de los microservicios por las siguientes razones (Wolff, n.d.) :

1. Se centran en la descomposición de por medio de servicios simples, enfatizando sobre el mecanismo de administración de los mismos. Los servicios en S.O.A usan el mismo contexto. Es decir, cada microservicio no está auto contenido.
2. S.O.A se centra en la estructura de T.I empresarial.

3. Usan lenguajes de programación empresariales entre sus servicios, como Java EE
4. En cuanto a administración del proyecto no es independiente.
5. Los cambios involucran a más de un servicio.

	SOA	Microservices
Scope	Enterprise-wide architecture	Architecture for one project
Flexibility	Flexibility by orchestration	Flexibility by fast deployment and rapid, independent development of Microservices
Organization	Services are implemented by different organizational units	Services are implemented by different organizational by teams in the same project
Deployment	Monolithic deployment of several services	Each microservice can be deployed individually
UI	Portal as universal UI for all services	Service contains UI

Figura 70 Diferencias entre SOA y Microservicios, tomado de *Microservices: Flexible Software Architecture* p.92

Según lo anterior, la arquitectura escogida para desarrollar nuestro producto fue la orientada a microservicios y sus ventajas.

Esta arquitectura además se enmarca en los diferentes proyectos cloud que el programa ha ejecutado desde hace un tiempo, por ejemplo, la uvirtual in cloud.

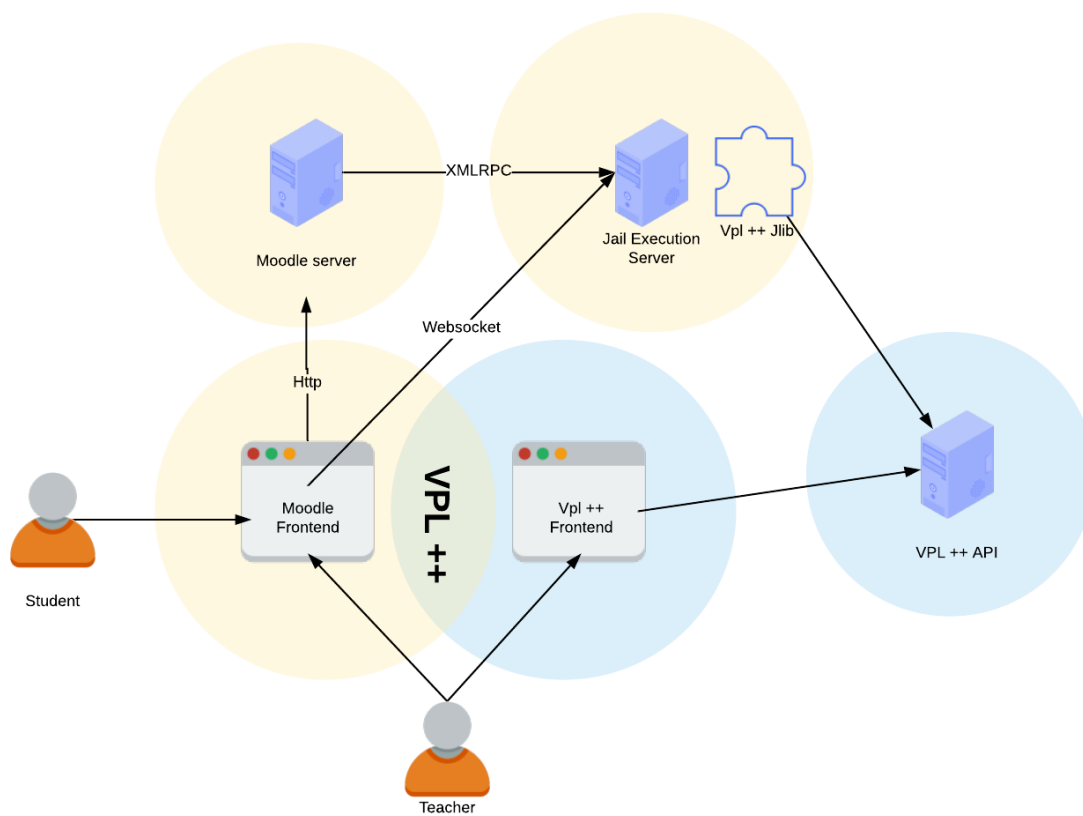


Figura 71 Distribución de los microservicios de VPL ++ por cada usuario

### 5.7.3 Diagrama de componentes del ecosistema

El diagrama de componentes muestra a grandes rasgos los microservicios y sus relaciones.



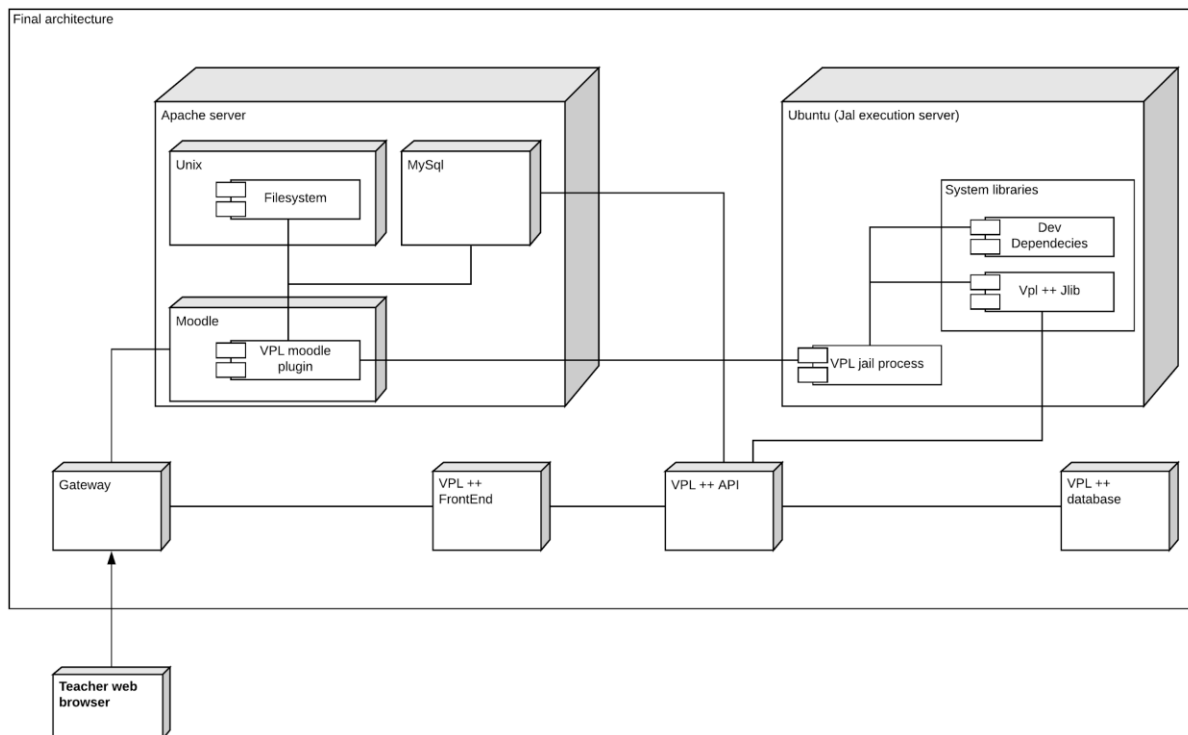


Figura 72 Diagrama de componentes de VPL ++

### 5.7.4 Diagrama de secuencia del ecosistema

El diagrama de secuencia muestra el proceso de ejecución de un programa del profesor / estudiante cuando éste intenta ejecutar una actividad.

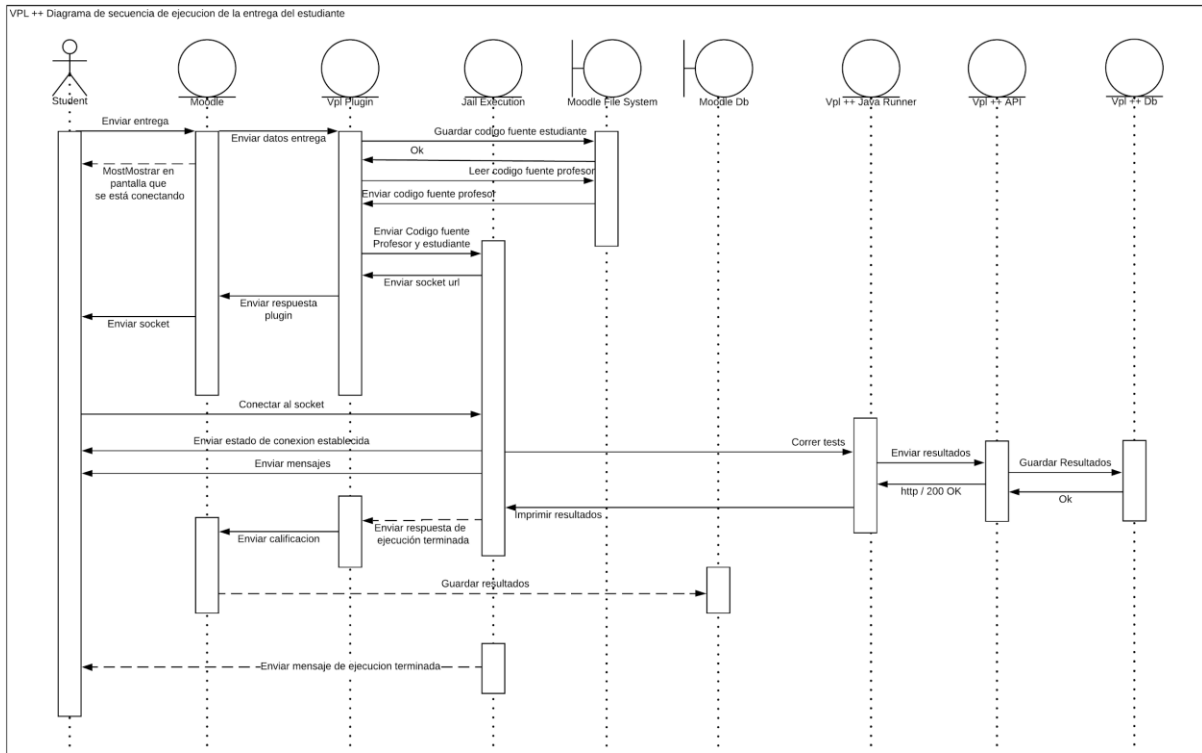


Figura 73 Diagrama de secuencia de VPL ++

### 5.7.5 Jaula de ejecución y VPL ++ JLib Runner

Recordando que VPL++ JLib runner es un pequeño software usado para comunicar la jaula de ejecución con la API de VPL ++. Está escrito en java, se ejecuta por consola y se encarga de ejecutar los tests de VPL ++, obtener los resultados, compararlos, enviarlos a la API para almacenar los resultados y finalmente enviar a Moodle los resultados con la calificación

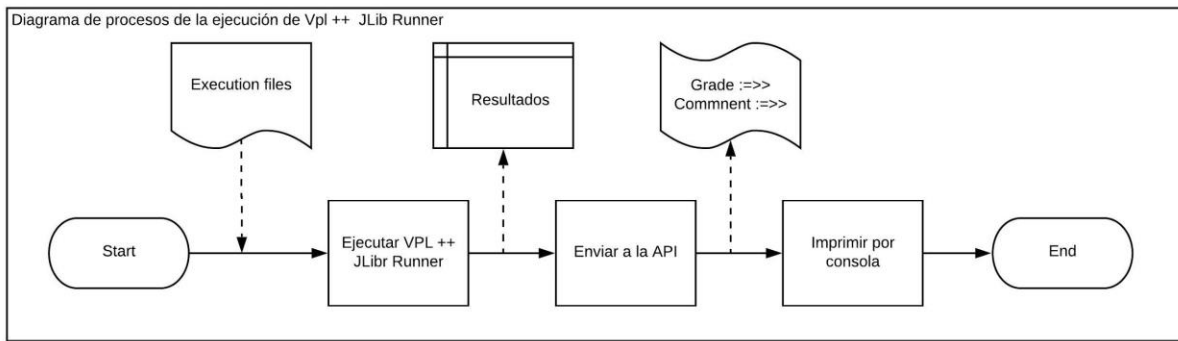


Figura 74 Diagrama de proceso de la calificación automática de VPL ++

## Casos de uso

La siguiente grafica describe los casos de uso para el Runner de JLib de VPL ++. Como podrá notar, solo el runner podrá almacenar respuestas de los estudiantes

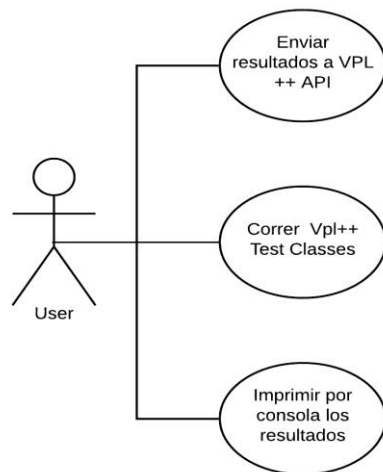


Figura 75 Casos de uso de VPL ++ JLib

## Diagrama de secuencia

En el siguiente diagrama de secuencia se muestra cada una de las acciones de las clases principales de JLib Runner

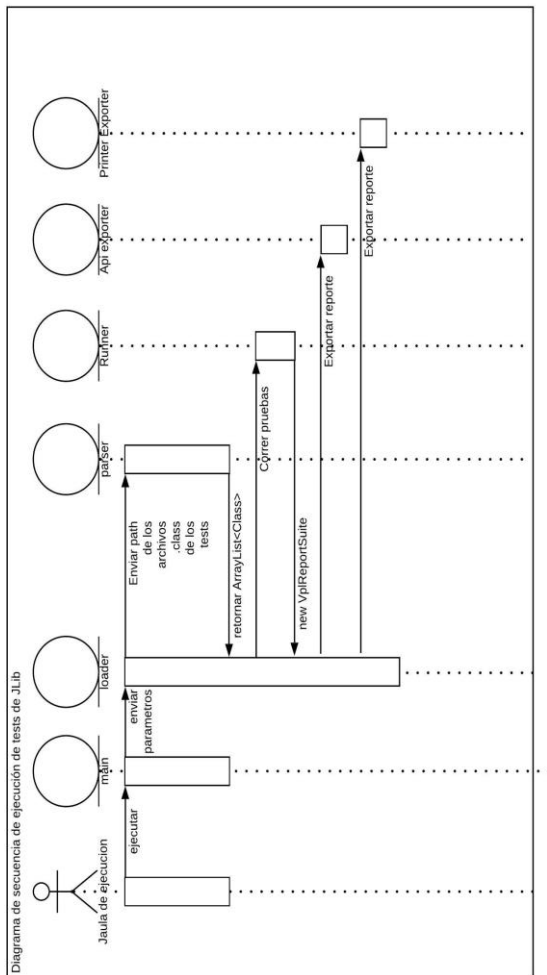


Figura 76 Diagrama de secuencia del proceso de ejecución de pruebas unitarias de VPL ++ por JLib



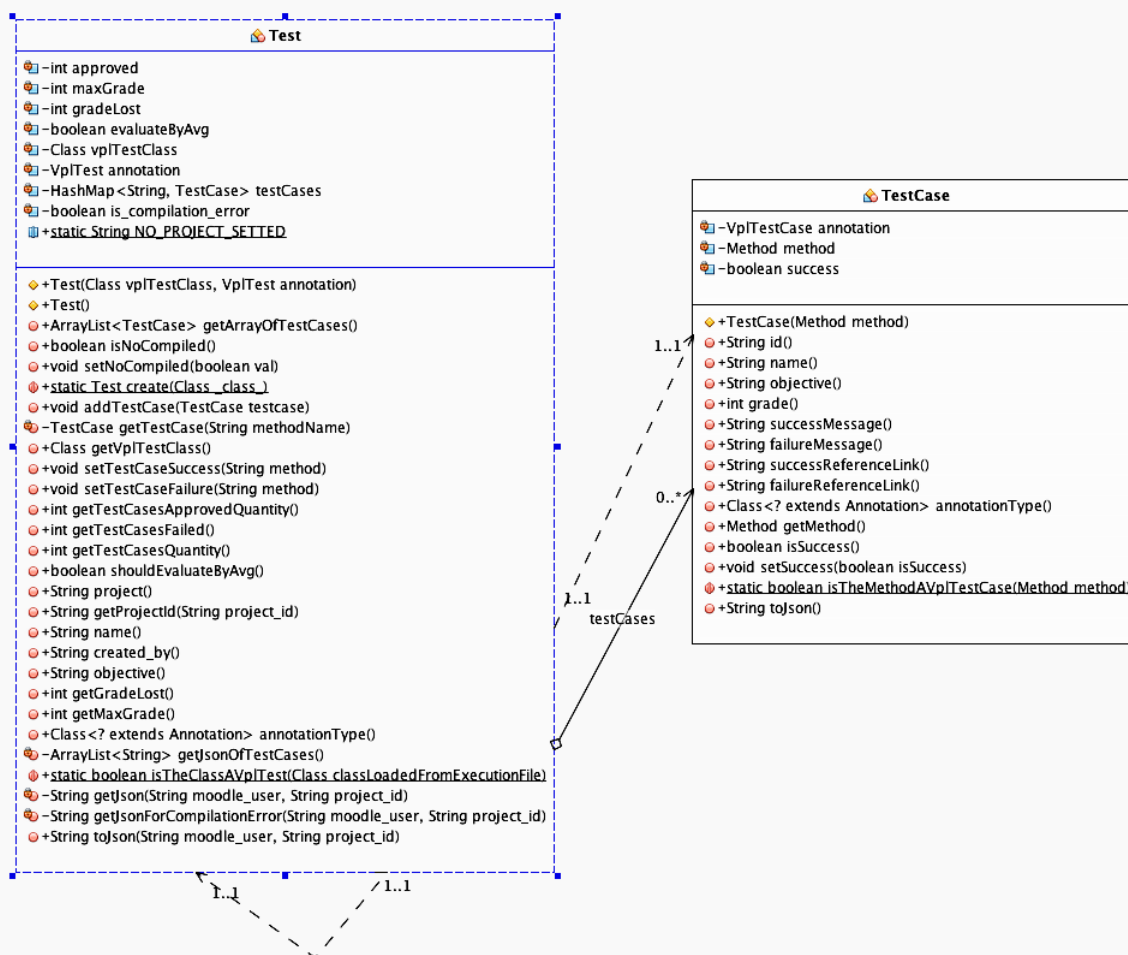


Figura 78 Diagrama de clases de Test y TestCase

### *VPL Parser Class*

Un parser es una clase cuya responsabilidad es tomar un ArrayList de Objetos Class de las pruebas del profesor, filtrar las clases que tengan el decorador @VPLPlusPlus, y devolver un ArrayList de la clase Test.

Además revisa si entre sus métodos tiene la etiqueta @test y la etiqueta @TestCase para así poder mapear los casos de prueba, finalmente cada caso de prueba se añade a la clase Test

orrespondiente

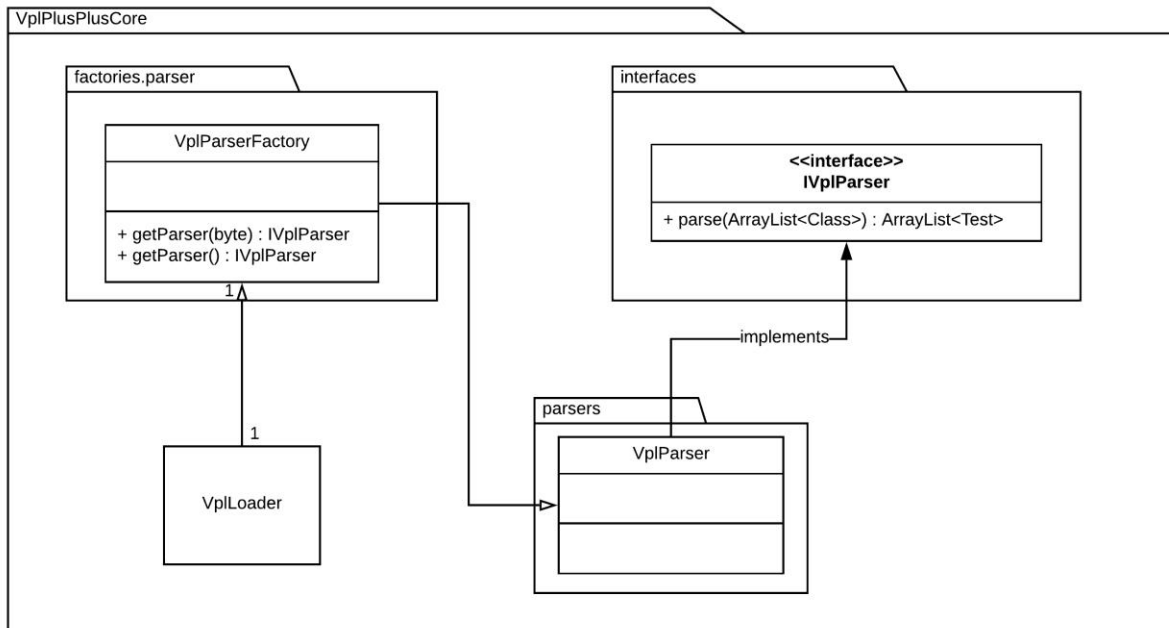


Figura 79 Diagrama de clases de una clase Parser

### *VPLSuiteReport class y VPLReport*

VPLReport es una clase que representa un reporte, este reporte incluye qué Test pasó correctamente, cuáles casos de pruebas fueron ejecutados satisfactoriamente, cual es la calificación obtenida, etc... Por otro lado, VPLSuiteReport es una clase que representa múltiples reportes.

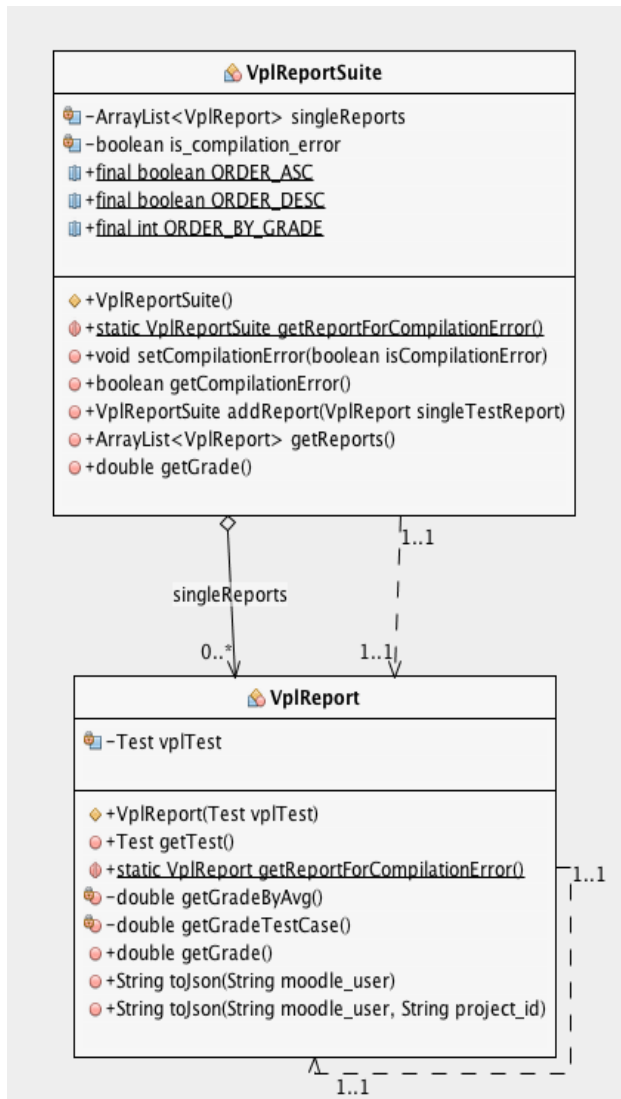


Figura 80 Diagrama de clase de VplReportSuite y VplReport



## VPL Runner Classes

Un Runner es una clase que usa la clase JUnitCore para ejecutar los tests de las pruebas del profesor. Al finalizar, hace un match entre los Test y sus respectivos TestCase, de esa manera logra identificar que caso de prueba pasó y cuál no.

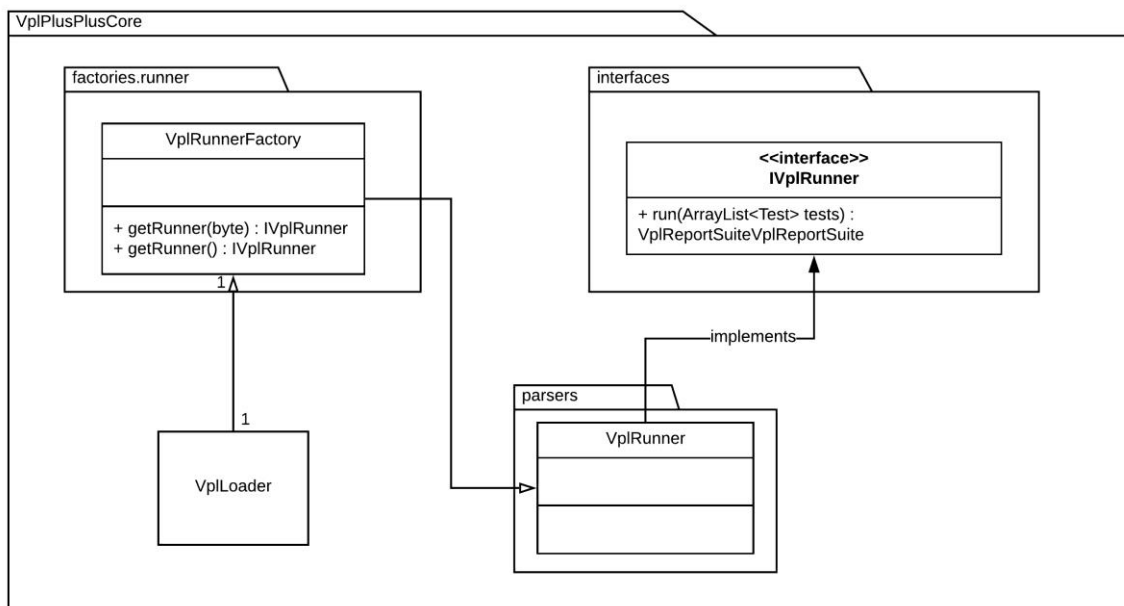


Figura 81 Diagrama de clases de VplRunner

## VPL Exporters

Un exporter es una clase que toma las clases VplReportSuite y la exporta a diferentes formatos. En la aplicación hay un exporter para enviar los datos a la API, esta clase se llama ApiExporter. También hay otra clase que se encarga de exportar los datos imprimiendo por consola, esta clase se llama Printer. Ambas implementan la interface IExporter.

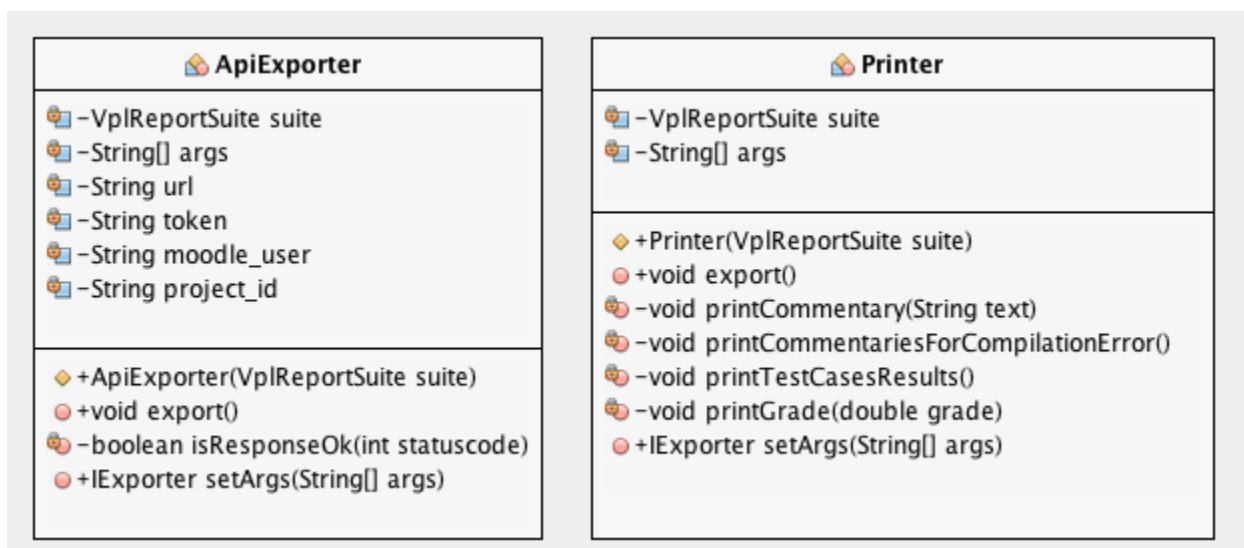


Figura 82 Diagrama de clase de la clase ApiExporter y Printer

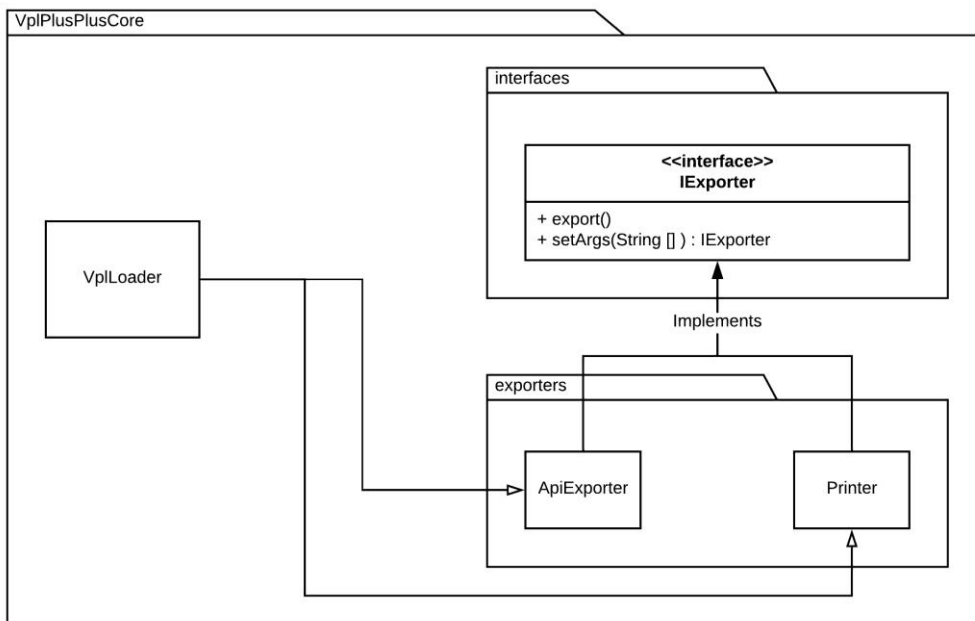


Figura 83 Diagrama de clases de un exporter de VPL ++

## 5.7.6 VPL ++ API

### *Casos de uso*

#### *VPL ++ JLib Runner*

Esta aplicación solo tiene un caso de uso, enviar los resultados de las ejecuciones de las pruebas del profesor al microservicio VPL ++ API

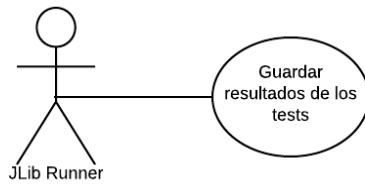


Figura 84 Caso de uso de JLib en el API

### *Administrador de Moodle*

El administrador de Moodle puede autenticarse en VPL ++ API usando las credenciales y el correo de Moodle. Él solo tiene tareas específicas de administración. El manejo de tópicos le fue delegado para evitar que los profesores modifiquen tópicos.

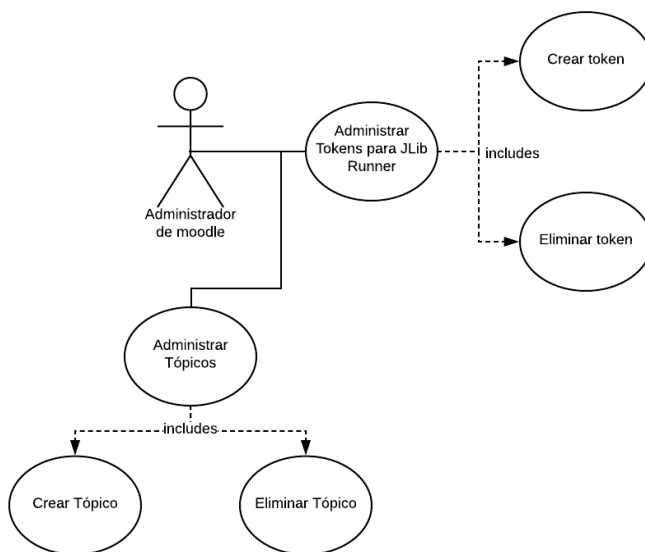


Figura 85 Diagrama de casos de uso del administrador de Moodle en el API

## *Profesor de Moodle*

Los profesores de Moodle también pueden iniciar sesión usando su email y contraseña de Moodle.

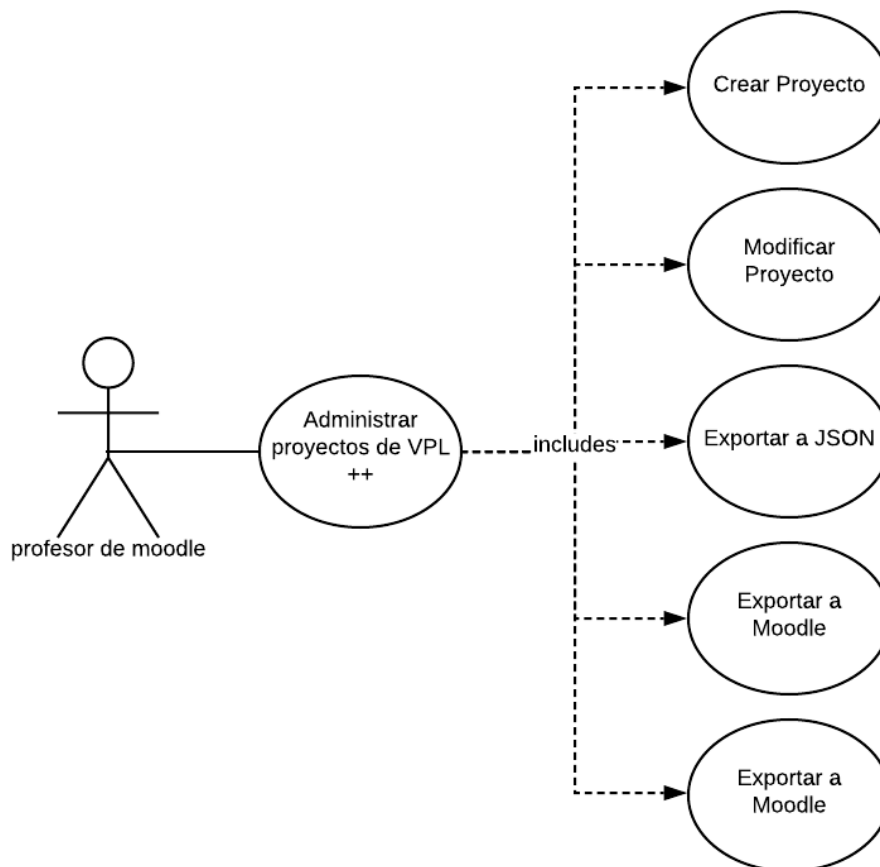


Figura 86 Diagrama de casos de uso del maestro en el API de VPL ++

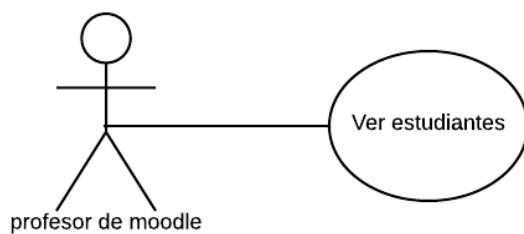


Figura 87 Diagrama de uso del profesor de Moodle para ver sus estudiantes

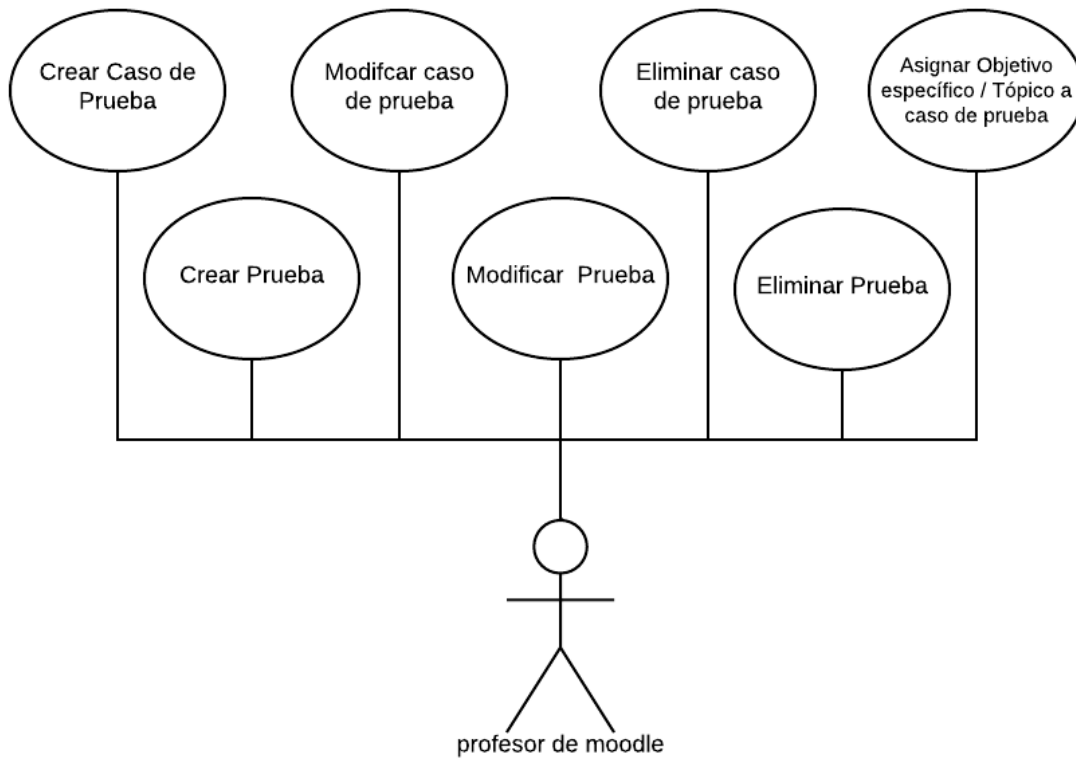


Figura 88 Diagrama de casos de uso del profesor de Moodle para administrar sus proyectos

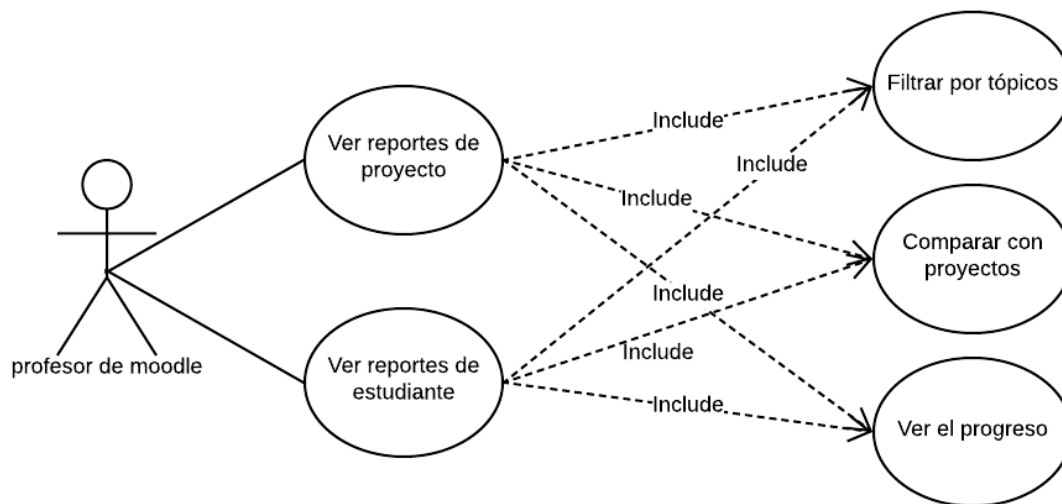


Figura 89 Diagrama de casos de uso del maestro sobre reportes

## *Arquitectura lógica*

Se usó una arquitectura basada en el dominio, es decir, separados por dominios en común.

1. Solo los modelos tendrán comunicación con la base de datos, ellos son los encargados de manejar la información allí.
2. Los servicios son clases fachada, hacen uso de los modelos, las funciones utilitarias y los webservices. Abstraen lógica de negocio específica, por ejemplo “crear proyecto”.
3. Los webservices son los que se comunican con los sistemas externos. Sus objetos actúan como modelos de Moodle. No tienen lógica de negocio, sino que hacen solo lectura de sistemas externos como Moodle.
4. Los controladores se encargan de recibir los requests http. Estos, limpian los headers, el body y el query de la consulta http. Luego, llaman al servicio al cual debe ser dirigida el request
5. Routes es la carpeta donde se almacenan los objetos que tienen la definición de cada una de las rutas que la API expone.

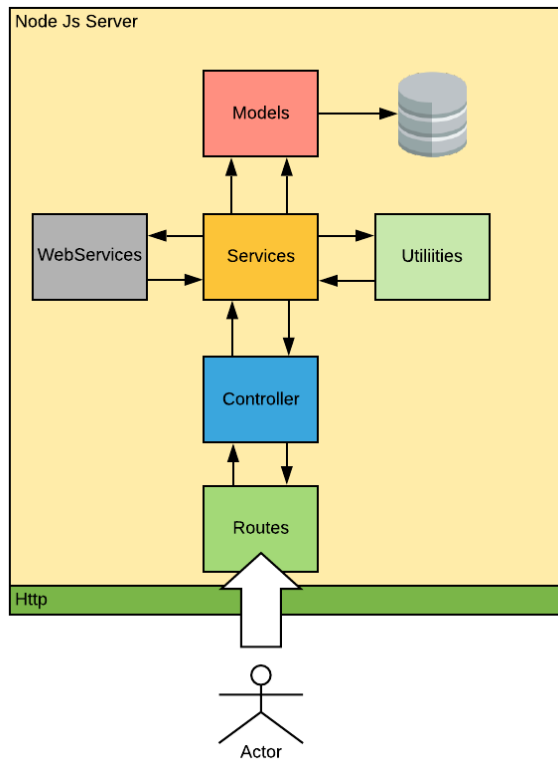


Figura 90 Arquitectura lógica de VPL ++ API

La estructura de archivos luce de la siguiente manera

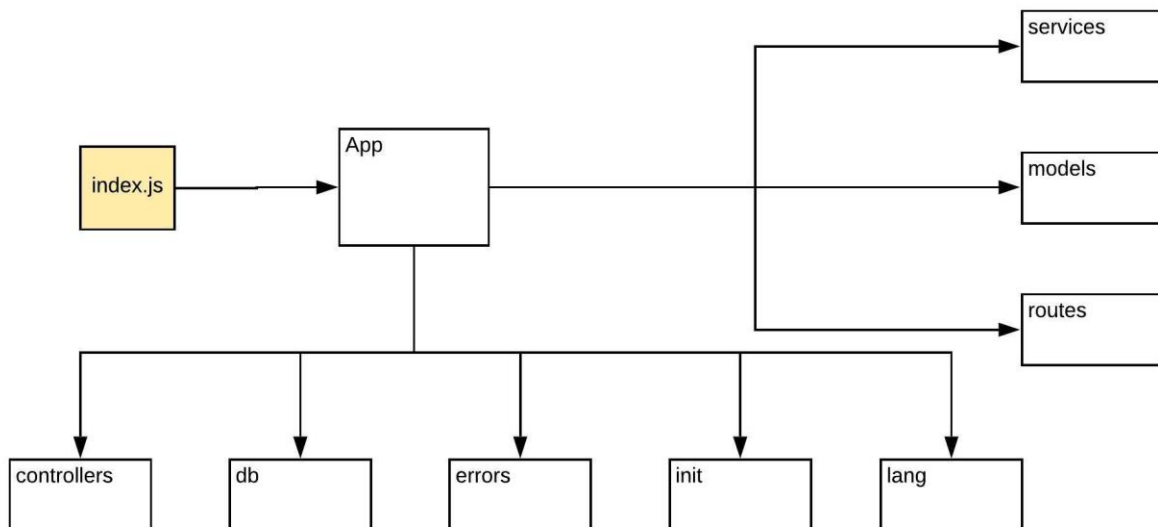


Figura 91 Distribución de las carpetas de VPL ++ API



En conjunto, los microservicios de VPL ++ lograron expandir el sistema (no un software en específico) satisfactoriamente. VPL ++ es una herramienta potente de análisis del progreso de los estudiantes, y así el profesor podrá desarrollar su metodología de aprendizaje más asertivamente.

En cuanto a los detalles técnicos, se usaron diferentes arquitecturas dependiendo de la tecnología. En el caso del API se usó una arquitectura basada en dominios, por otro lado, JLib Runner usó el paradigma orientado a objetos. Como tal, cada microservicio fue tratado de manera diferente, respetando su autonomía.

Al cada ser independiente cada microservicio, se tuvo la libertad de desarrollar sin restricciones de software, o dependencias entre uno y otros. Fue un enfoque interesante que vale la pena seguir replicando y estudiando.

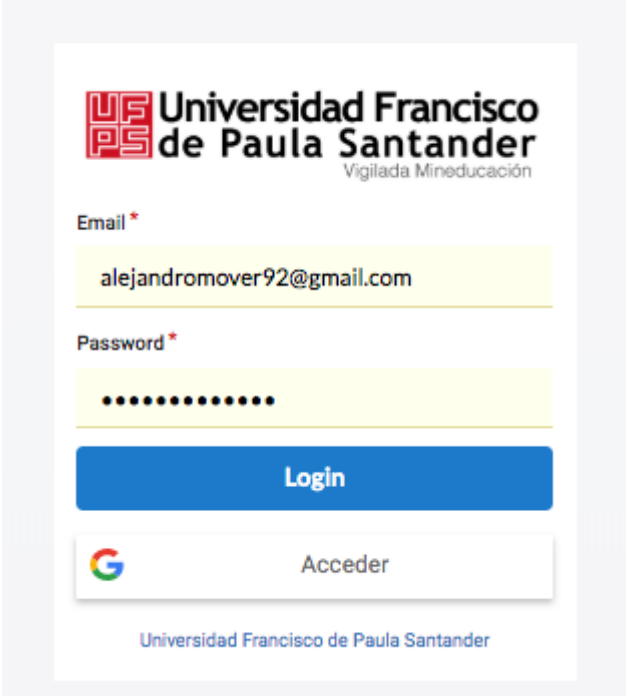
## 5.8 Manual de usuario

### 5.8.1 Instalación y configuración

Las instrucciones de instalación y configuración se pueden encontrar en los repositorios correspondientes de cada microservicios, ver anexos.

### 5.8.2 Iniciar sesión

Si no ha iniciado sesión verá la siguiente pantalla



Logo de la Universidad Francisco de Paula Santander, con el texto "Vigilada Mineducación".

Formulario de inicio de sesión con los siguientes campos:

- Etiqueta "Email \*": campo de texto con el valor "alejandromover92@gmail.com".
- Etiqueta "Password \*": campo de contraseña con caracteres ocultos por puntos.
- Botón "Login" de color azul.
- Botón "Acceder" con el logo de Google.

En la parte inferior del formulario, se muestra el texto "Universidad Francisco de Paula Santander".

Figura 92 Formulario inicio de sesión

Para iniciar sesión use el correo y la contraseña de su usuario en Moodle. También podrá iniciar sesión con Google si tiene un usuario asociado al correo de la cuenta de Google.

### 5.8.3 Profesores

La interfaz gráfica de los profesores permite administrar los proyectos de VPL ++, crear, modificar, exportar, restaurar y eliminar proyectos. Además, podrá generar reportes especializados de sus estudiantes y actividades.

## *Dashboard y administrar proyectos*

Al iniciar sesión, el profesor podrá ver la siguiente vista.

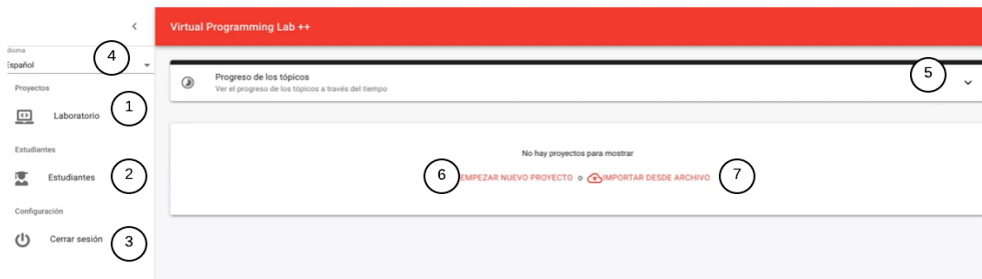


Figura 93 Dashboard sin proyectos

1. Menú laboratorio: abre la ventana para administrar los proyectos y sus reportes
2. Menú estudiantes: abre la ventana para administrar los reportes de los estudiantes
3. Menú cerrar sesión: cierra la sesión de usuario
4. Selector de idioma, selecciona el idioma de la interfaz gráfica
5. Abre la pestaña de progreso de los tópicos
6. Botón para crear un nuevo proyecto de VPL ++
7. Abre una ventana para cargar un proyecto de VPL ++ en formato JSON y editarlo.

De lo contrario, si el profesor ya ha creado proyectos de VPL ++, la ventana de laboratorio mostrará una tabla con los proyectos presentes

Virtual Programming Lab ++

Progreso de los tópicos  
Ver el progreso de los tópicos a través del tiempo

Projects

1	2	3	4	5	6	7
Nombre ↑	Descripción	Modificable	Actividad Id	Tests	Casos de prueba	Número de entregas
<input type="checkbox"/>	CasaDeCambioTest	CasaDeCambioTest	No	2	1	25
<input type="checkbox"/>	FracionarioTest	FracionarioTest	No	6	1	5
<input type="checkbox"/>	MaquinaCafeteraTest	MaquinaCafeteraTest	No	5	1	20
<input type="checkbox"/>	Otro Reto para estudiar Segundo Previo	Otro Reto para estudiar Segundo Previo	No	12	3	10
<input type="checkbox"/>	Quiz Trabajo en Clase	Quiz Trabajo en Clase	No	9	5	25

Items per page: 5 1-6 of 9

Figura 94 Dashboard con proyectos

1. Columna nombre del proyecto
2. Columna descripción del proyecto
3. Columna que indica si el proyecto es modificable o no, un proyecto no se puede modificar si existe una entrega para sus casos de prueba
4. Columna actividad id, hace referencia a la actividad de Moodle VPL en Moodle
5. Columna tests, muestra cuantos tests tiene ese proyecto
6. Columna casos de prueba, muestra cuántos casos de prueba tiene ese proyecto
7. Columna número de entregas, muestra cuántas entregas tiene ese proyecto, es decir, cuántas veces los estudiantes han enviado respuestas para ese proyecto
8. Botón crear proyecto
9. Botón restaurar proyecto de VPL ++ desde JSON

## *Modificar proyecto*

Un proyecto se puede modificar solo si este tiene entregas de estudiantes, es decir, que se ejecutaron las pruebas unitarias de VPL ++ y se guardaron los resultados. Al seleccionar un proyecto en la tabla, el encabezado cambiará y mostrará los siguientes botones, según si es modificable o no:

Botones si el proyecto NO es modificable



Figura 95 Menú al seleccionar proyecto que no se puede modificar

1. Ver proyecto
2. Eliminar Proyecto
3. Ver reportes del proyecto
4. Descargar proyecto en JSON, útil para compartir proyectos entre profesores
5. Descargar proyecto de Moodle
6. Quitar selección del proyecto

Si intenta eliminar un proyecto que no es modificable obtendrá el siguiente mensaje de alerta

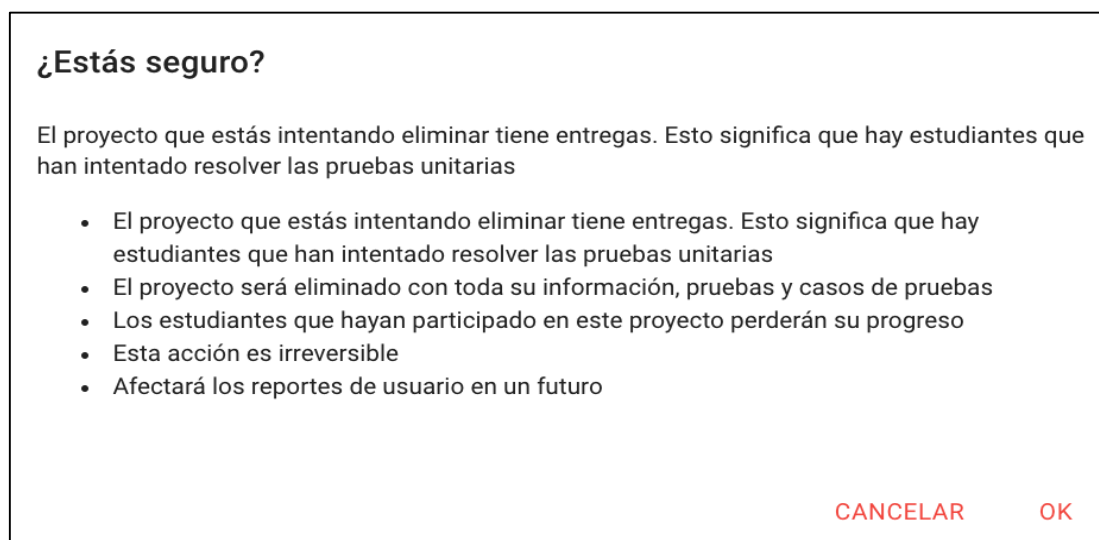


Figura 96 Advertencia al eliminar proyecto con respuestas de los estudiantes

Botones si el proyecto es modificable



Figura 97 Menú al seleccionar proyecto que si se puede modificar

A diferencia de cuando un proyecto no es editable, el botón de ver reportes no estará disponible, pero si estará disponible el botón editar.

## Crear proyecto

Para crear un proyecto deberá hacer clic en el botón crear proyecto que está sobre la tabla de proyectos, o si aún no ha creado un proyecto, en el botón “Empezar crear proyecto”. Vea el capítulo ver proyectos para más información.

Al hacer clic en cualquiera de los dos botones se le mostrará la siguiente vista:

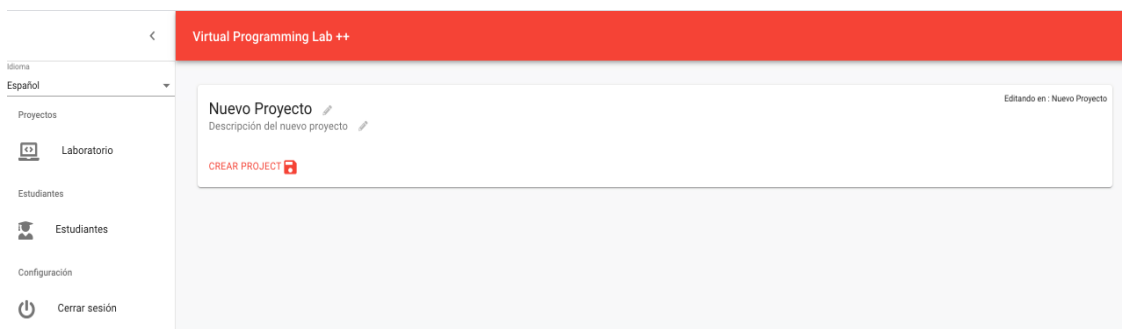



Figura 98 Crear proyecto vista

El ícono de lápiz  indica que es un campo editable. El primer campo es el nombre del proyecto, y el segundo la descripción. Edítelos y haga clic en el botón “Crear proyecto”, ésto creará un nuevo proyecto y podrá editarlo.

## Crear test

Al crear un nuevo proyecto, o al editar uno existente, podrá ver el botón para crear un test debajo de la pestaña de nombre y descripción.

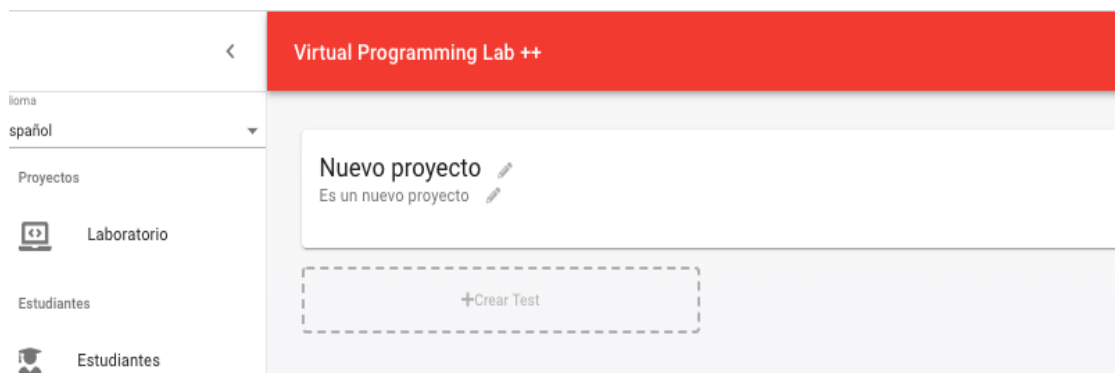


Figura 99 Vista crear test

Al hacer clic, en “Crear Test” verá que se añade una nueva tarjeta con un test que podrá editar

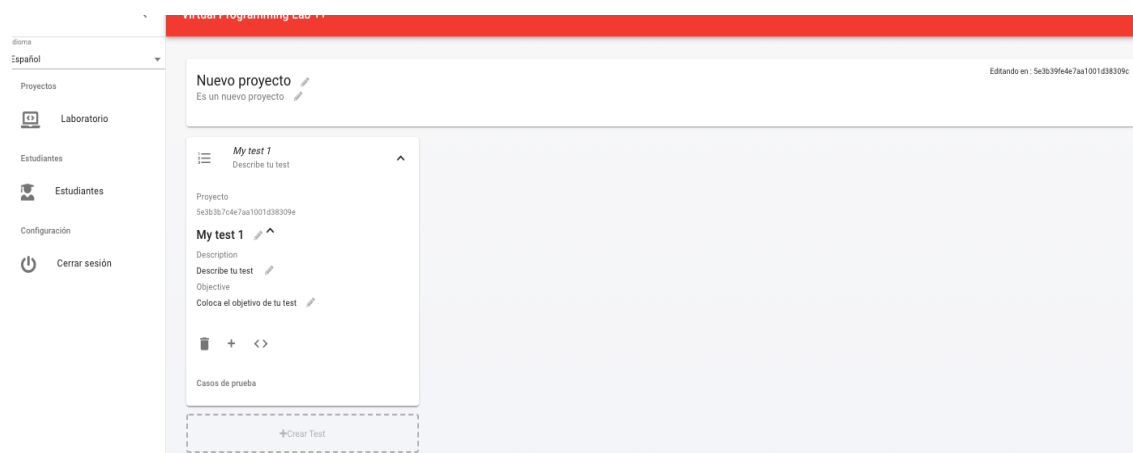


Figura 100 Vista test recién creado

Puede agregar cuantos tests desee.

## *Configurar test*

Un test en la interfaz gráfica luce como una tarjeta con las siguientes características:





Figura 101 Tarjeta de test

1. Vista del Nombre del test
2. Vista de la descripción del test
3. Abrir/cerrar pestaña
4. Id del proyecto al cual pertenece el test
5. Editar nombre del test
6. Editar descripción del test
7. Editar objetivo del test
8. Eliminar test
9. Agregar un caso de prueba

### *Editar código del test*

Hacer clic en el botón pre visualizar para ver cómo quedará el código del test.

## *Configurar test para que califique por promedio*

Si desea que los tests sean calificados por promedio, coloque en cero la calificación de al menos un caso de prueba.

## *Configurar test para que califique por ponderación*

Si desea que los tests sean calificados por ponderación, seleccione en cada caso de prueba del test un valor en su calificación, entre más sea la diferencia entre las calificaciones de los casos de pruebas, más peso tendrá el caso de prueba.

## *Casos de prueba*

Para crear un caso de prueba haga clic en el botón de crear caso de prueba (ver fig. 101). El caso de prueba lucirá como una tarjeta dentro del test en la parte inferior de la tarjeta de test.



Figura 102 Test con su caso de prueba recién creado

1. Ver ventana de configurar caso de prueba
2. Eliminar caso de prueba

## *Configurar caso de prueba*

Para configurar un caso de prueba haga clic sobre él mismo.



The screenshot shows a vertical list of five configuration items, each with a small icon on the left and a dropdown arrow on the right. Below the list is a red button labeled 'GUARDAR'.

- Editar información básica**  
Configura tu caso de prueba, por favor se descriptivo
- Escribe el código del caso de prueba**  
Escribe el cuerpo del método de la prueba unitaria.
- Configurar la calificación y tópicos**  
Los tópicos permiten hacer un seguimiento a un estudiante a lo largo del tiempo. La calificación será usada para calificar en moodle.
- Establecer la retroalimentación positiva**  
Escriba un mensaje para el estudiante en caso de éxito
- Establecer la retroalimentación negativa**  
Escriba un mensaje para el estudiante en caso de falla

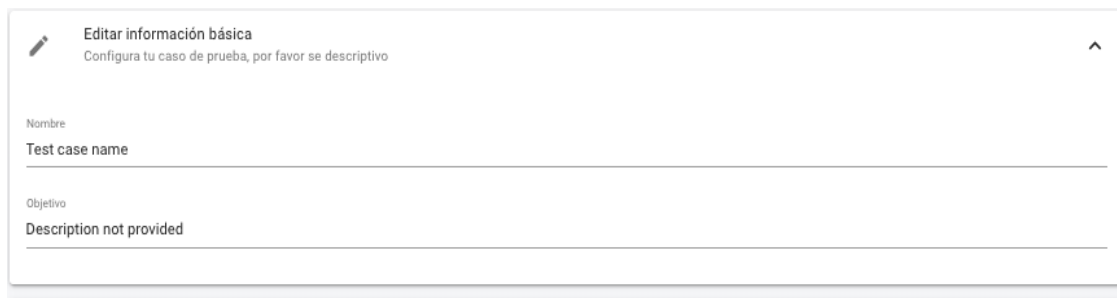
GUARDAR

Figura 103 Ventana para configurar caso de prueba

Para configurar cada ítem, haga clic en la flecha a la derecha la pestaña que desee editar.

### *Editar información básica*

En esta pestaña podrá editar la información básica del caso de prueba, es decir, nombre y descripción.



**Editar información básica**  
Configura tu caso de prueba, por favor se descriptivo

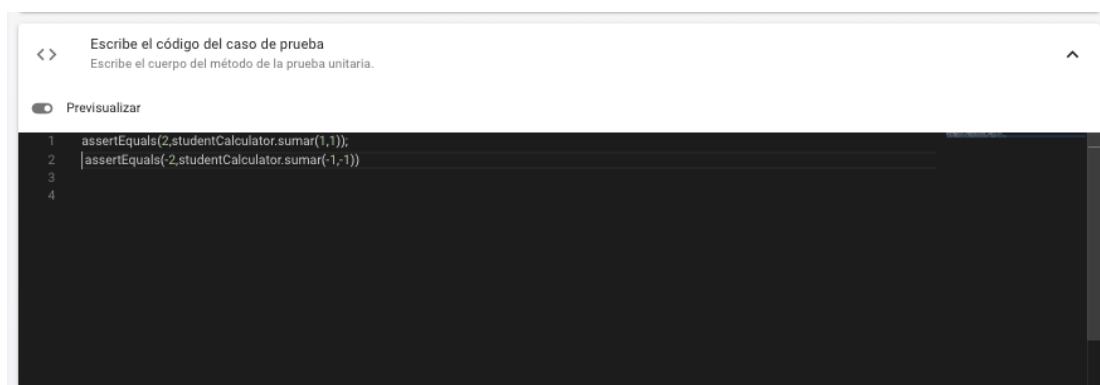
Nombre  
Test case name

Objetivo  
Description not provided

Figura 104 Caso de prueba - Información general

### *Editar código del caso de prueba*

En esta pestaña puede configurar el cuerpo del caso de prueba



**Escribe el código del caso de prueba**  
Escribe el cuerpo del método de la prueba unitaria.

Previsualizar

```
1 assertEquals(2,studentCalculator.sumar(1,1)).
2 |assertEquals(-2,studentCalculator.sumar(-1,-1))
3
4
```

Figura 105 Editar código fuente del caso de prueba

Si hace clic en el botón pre visualizar podrá observar cómo quedaría su caso de prueba:

```

1
2  @Test()
3  public void Testcasename() {
4      assertEquals(2,studentCalculator.sumar(1,1));
5      assertEquals(-2,studentCalculator.sumar(-1,-1))
6
7
8  }
9

```

Figura 106 Caso de prueba compilado

### *Configurar la calificación y los tópicos*

En esta pestaña podrás asignar los tópicos al caso de prueba y su calificación. No necesitará más de 3 tópicos por caso de prueba, si necesita agregar más, considere agregar más casos de prueba para los tópicos adicionales.

Figura 107 Pestaña para configurar la calificación y los tópicos del caso de prueba

Escriba en la casilla de tópicos para buscar alguno, o haga clic en la flecha a la derecha de la casilla.

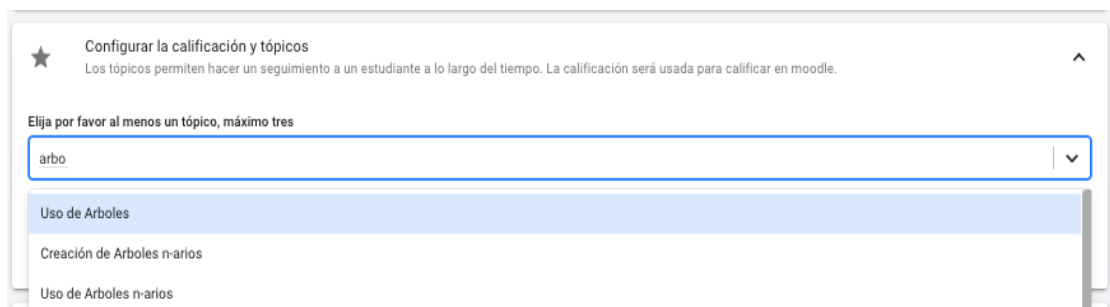


Figura 108 Seleccionar tópico

### *Establecer retroalimentación positiva*

Introduzca un mensaje de éxito y un link de referencia para el estudiante pase el caso de prueba



Figura 109 Configurar retroalimentación positiva

### *Establecer retroalimentación negativa*

Introduzca un mensaje de éxito y un link de referencia para el estudiante no pase el caso de prueba

Figura 110 Configurar retroalimentación negativa

Después haga clic en el botón guardar para guardar el caso de prueba

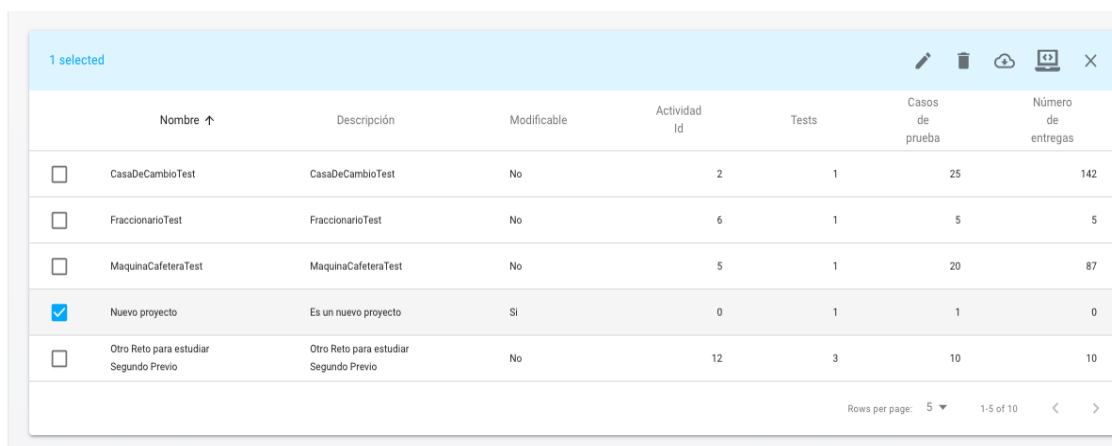
Figura 111 Vista final de configurar un caso de prueba

Cree tantos tests o casos de prueba usted necesite. Los cambios son automáticamente guardados.

## Exportar proyecto a Moodle


Luego de crear un proyecto se necesita exportarse para integrarlo con Moodle. Luego, los estudiantes podrán resolver los ejercicios de programación. Al exportar un proyecto en Moodle, VPL ++ creará un archivo de restauración de Moodle. Este archivo, contendrá la información de la actividad de VPL ++, además añadirá los archivos de ejecución necesarios para evaluar las pruebas de los estudiantes usando las pruebas unitarias de VPL++, éstas últimas también son generadas automáticamente.

Vaya a la ventana de laboratorio, y seleccione un proyecto.



Nombre ↑	Descripción	Modificable	Actividad Id	Tests	Casos de prueba	Número de entregas	
<input type="checkbox"/>	CasaDeCambioTest	CasaDeCambioTest	No	2	1	25	142
<input type="checkbox"/>	FraccionarioTest	FraccionarioTest	No	6	1	5	5
<input type="checkbox"/>	MaquinaCafeteraTest	MaquinaCafeteraTest	No	5	1	20	87
<input checked="" type="checkbox"/>	Nuevo proyecto	Es un nuevo proyecto	Si	0	1	1	0
<input type="checkbox"/>	Otro Reto para estudiar Segundo Previo	Otro Reto para estudiar Segundo Previo	No	12	3	10	10

Figura 112 Seleccionar proyecto para exportar

Haga clic en el botón exportar a Moodle , inmediatamente se abrirá una ventana para descargar un archivo. Guárdelo con el nombre que desee.



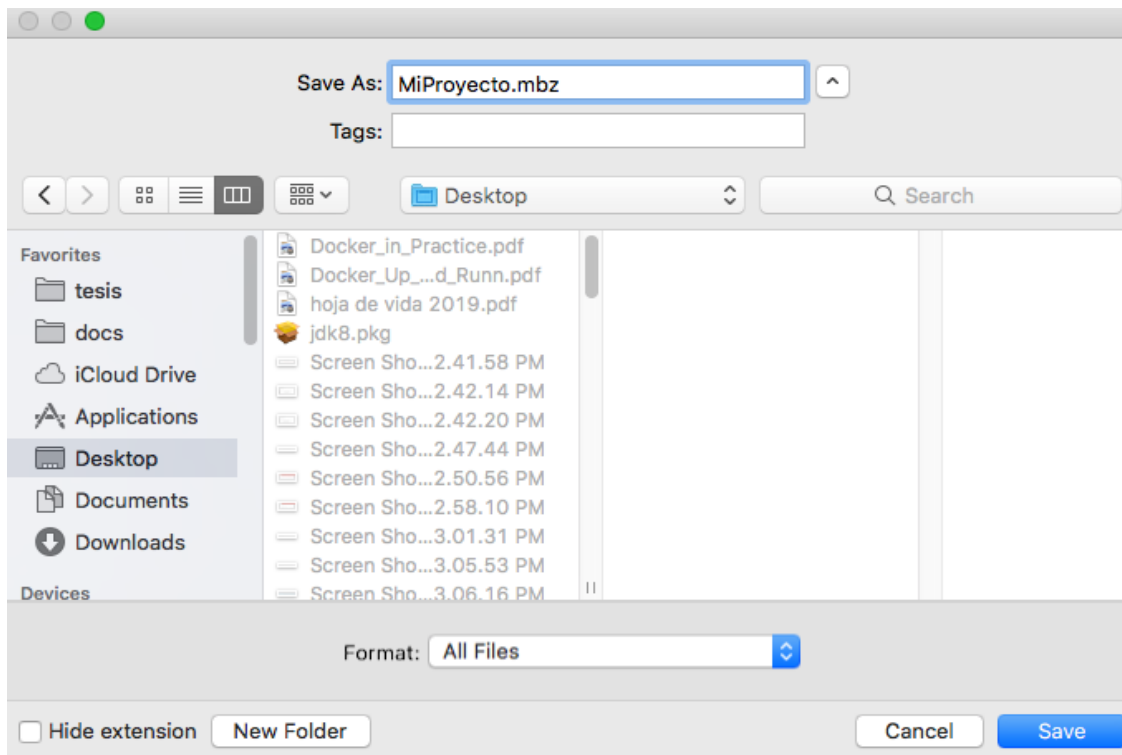


Figura 113 Guardar archivo de restauración de actividad de Moodle generado por VPL++

Este es un archivo de Moodle y podrá usarse para importarlo vía restauración. En Moodle vaya a administrar el curso y active el modo de edición.

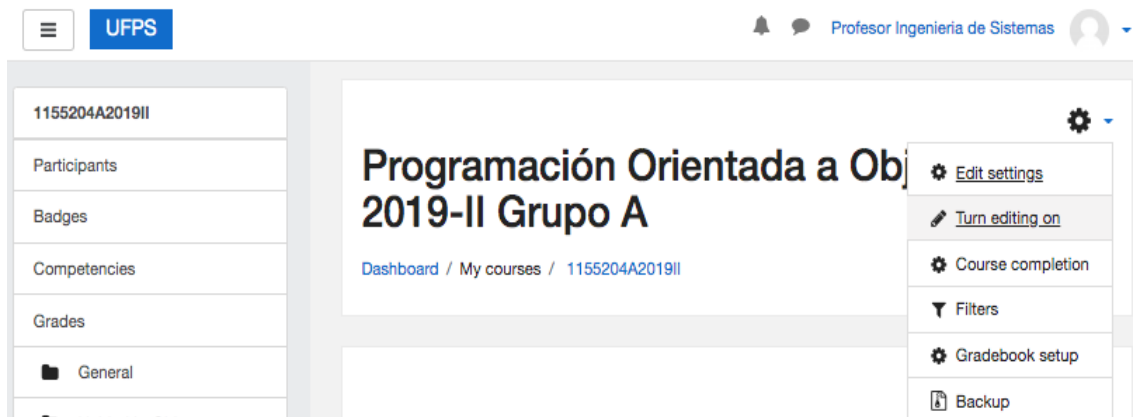


Figura 114 Activar modo de edición de Moodle

Ahora seleccione la opción restaurar del menú anterior.

The screenshot shows the Moodle course configuration interface for '1155204A2019II'. The left sidebar contains a navigation menu with options like 'Participants', 'Badges', 'Competencies', 'Grades', and 'General'. The main content area displays the course title 'Programación Orientada a Objetos 2019-II Grupo A' and a settings menu on the right. The 'Restore' option is highlighted in the settings menu, which also includes options like 'Edit settings', 'Turn editing off', 'Course completion', 'Filters', 'Gradebook setup', 'Backup', 'Import', and 'Reset'.

Figura 115 Menú restaurar en configuración de curso

Suba el archivo que descargó en VPL ++ a Moodle, y haga clic en restaurar.

The screenshot shows the 'Restore course' page in Moodle. The page title is 'Restore course' and the breadcrumb trail is 'Dashboard / My courses / 1155204A2019II / Restore'. The main content area is titled 'Import a backup file' and features a file upload section. A red warning icon is present next to the 'Files' label. The upload area includes a 'Choose a file...' button, a file name 'MiProyecto.mbz', and a note 'Maximum size for new files: 2MB'. A blue 'Restore' button is located at the bottom of the section.

Figura 116 Vista de archivo de restauración de actividad de Moodle cargado

Haga clic en siguiente cada vez que se le solicite. Le pedirá el curso donde usted lo desea guardar.

1. Confirm ▶ 2. Destination ▶ 3. Settings ▶ 4. Schema ▶ 5. Review ▶ 6. Process ▶ 7. Complete

## Restore into an existing course

Select a course

Course short name	Course full name
<input checked="" type="checkbox"/> 1155204A2019II	Programación Orientada a Objetos I 2019-II Grupo A

Search

Continue

Figura 117 Vista Seleccionar curso para restaurar la actividad de VPL ++

De clic en siguiente hasta que lo restaure.

1. Confirm ▶ 2. Destination ▶ 3. Settings ▶ 4. Schema ▶ 5. Review ▶ 6. Process ▶ 7. Complete

The course was restored successfully, clicking the continue button below will take you to view the course you restored. ✕

Continue

Figura 118 Vista que confirma que Moodle restauró la actividad normalmente

De clic en continuar. Verá que en la primera sección fue restaurado

Figura 119 Actividad de VPL ++ restaurada satisfactoriamente

Arrastrarlo hasta la sección indicada.

### *Vincular actividad de Moodle de VPL++ al proyecto de VPL++*

Antes que el estudiante resuelva la actividad deberá vincular el proyecto de VPL ++ a la actividad de Moodle recién creada. Este paso es fundamental, si no lo hace el estudiante no podrá evaluar sus entregas.

Seleccione el proyecto recién restaurado en el frontend de VPL ++, y haga clic en modificar, tal cual se ha explicado anteriormente.

Podrá ver que se ha activado una nueva pestaña llamada “Actividad del proyecto”, debajo de la tarjeta que contiene el nombre y la descripción del proyecto.

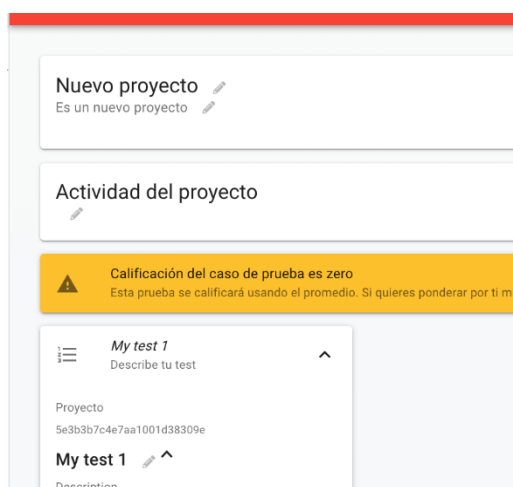


Figura 120 Pestaña actividad del proyecto

Haga clic en el icono del lápiz para vincular el proyecto de VPL++ con la actividad de Moodle

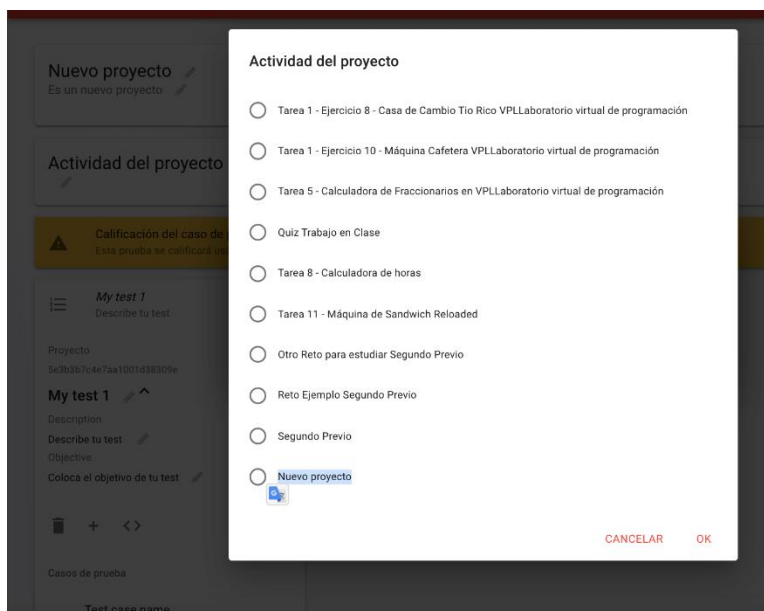


Figura 121 Dialogo de actividades de Moodle de VPL

## *Configurar actividad de VPL de VPL ++*

Después de subir la actividad de VPL ++ deberá configurar la actividad de Moodle como una actividad más de VPL.

## *Editar prueba de VPL ++ para usar ArrayList y otras librerías de Java en un test de VPL +++*

En algunos casos usted necesitará usar ArrayList y otras Apis de Java en sus pruebas de VPL ++, entre a la actividad, y seleccione en el menú de configuración “Archivos de Ejecución”.

Nota: podrá modificar las pruebas cuando desee, si ese es el caso, recuerda cambiar la configuración del proyecto en VPL ++ para mantener consistencia. Lo más recomendable es no cambiar las pruebas.

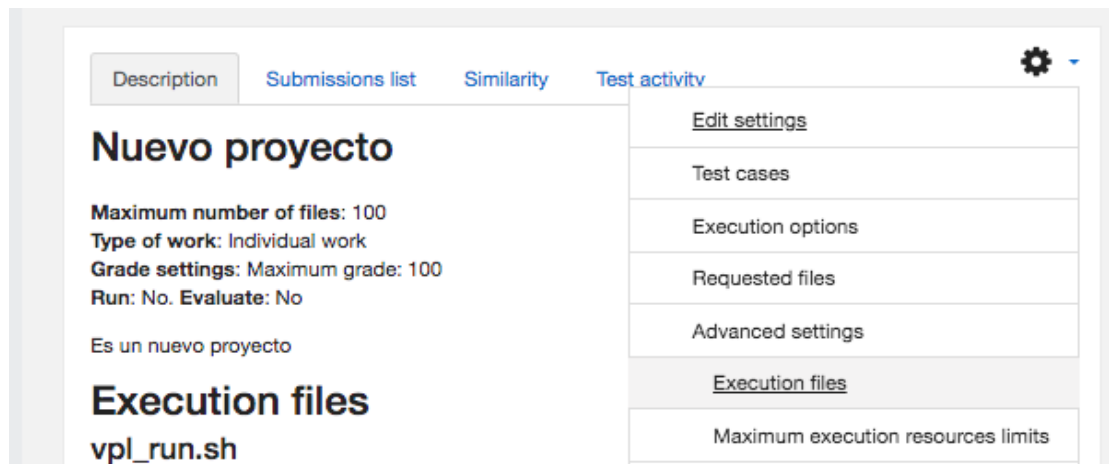


Figura 122 Menú archivos de ejecución

En el IDE de VPL seleccione la prueba, y agregue el import de la librería que desea.

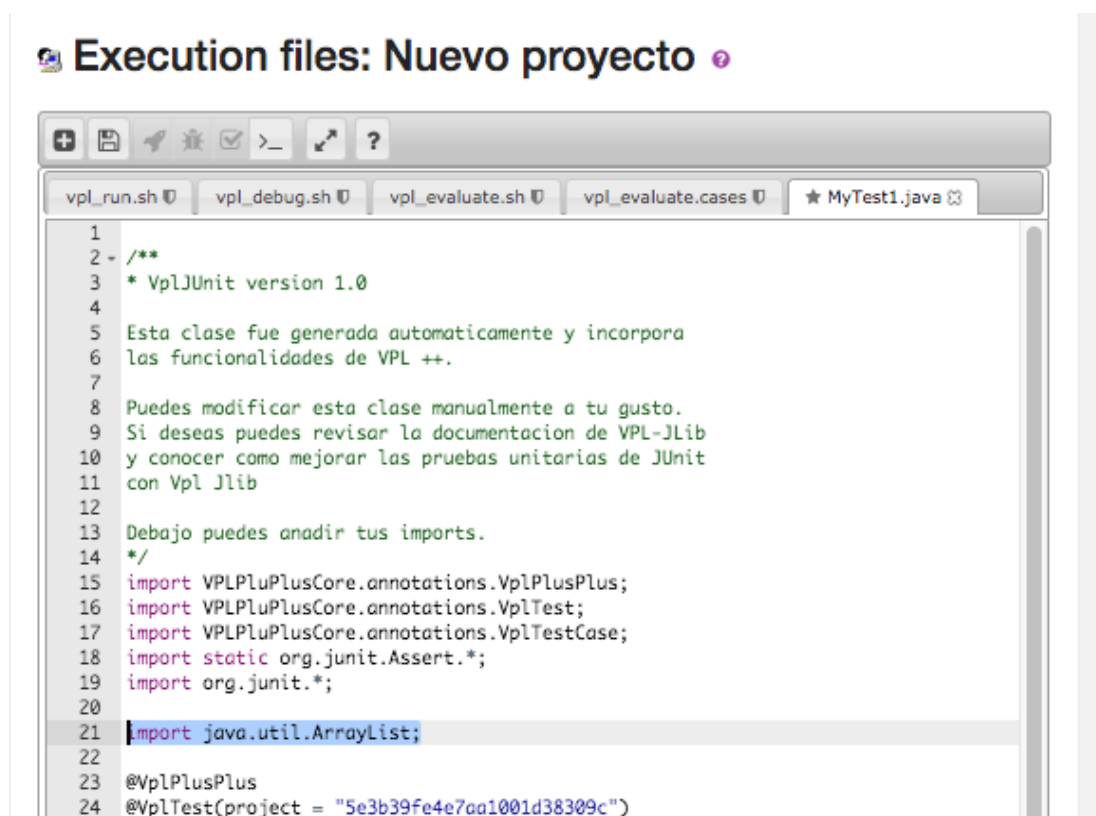


Figura 123 Añadir ArrayList a la prueba unitaria de VPL ++

Cuando termine haga clic en el botón guardar .

Al finalizar puede probar el ejercicio subiendo el código para probar la actividad, como una actividad de VPL normal.

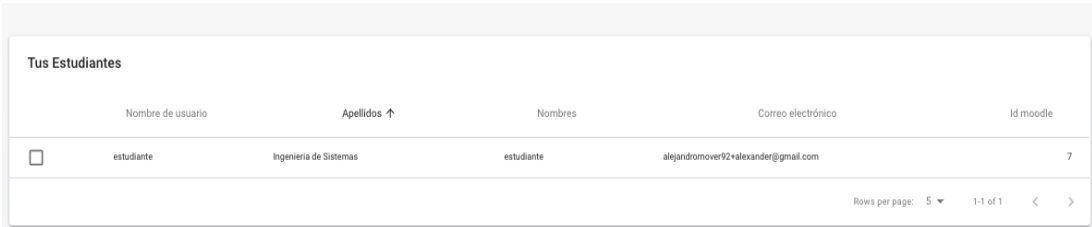
### *Administrar estudiantes*

Podrá solo ver los estudiantes a los que el profesor tiene acceso, desde Moodle podrá:

1. Eliminar estudiantes
2. Modificar estudiante
3. Agregar estudiante

Si desea hacerlo deberá hacerlo directamente en Moodle según los permisos a lo que su rol le permitirá.

En el menú, haga clic en el botón administrar estudiantes. Verá la tabla siguiente



Tus Estudiantes					
	Nombre de usuario	Apellidos ↑	Nombres	Correo electrónico	Id moodle
<input type="checkbox"/>	estudiante	Ingeniería de Sistemas	estudiante	alejandromover92+alexander@gmail.com	7

Rows per page: 5 ▾ 1-1 of 1 < >

Figura 124 Tabla de estudiantes



Si desea ver reportes del estudiante haga clic en la fila de la tabla del estudiante que desea generar el reporte.

El encabezado de la tabla mostrará lo siguiente

Nombre de usuario	Apellidos ↑	Nombres	Correo electrónico	Id moodle	
<input checked="" type="checkbox"/>	estudiante	Ingeniería de Sistemas	estudiante	alejandromover92+alexander@gmail.com	7

Figura 125 Seleccionar estudiante

De izquierda a derecha, el primer botón, mostrará la vista de ver reportes del estudiante seleccionado, el segundo botón (con forma de equis) quitará la selección del estudiante.

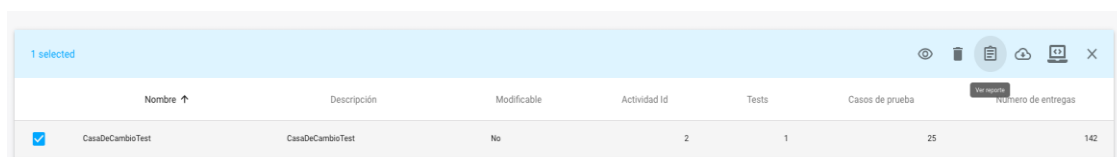
## *Reportes*

Los reportes son el principal componente de VPL ++, pues el profesor podrá ver el desempeño de sus estudiantes y sus proyectos. Esto le ayudará a ajustar su metodología de enseñanza adecuadamente.

La vista de reportes luce idéntica para reportes de estudiantes y proyectos. La diferencia es que en el reporte de proyectos traerá un reporte por estudiante según los parámetros solicitados, mientras que, en reportes de estudiantes solo traerá el reporte del estudiante y según los parámetros solicitado.

Un reporte de estudiantes solo mostrará un reporte, en el reporte de proyectos mostrará más los reportes de los estudiantes que participaron en el.

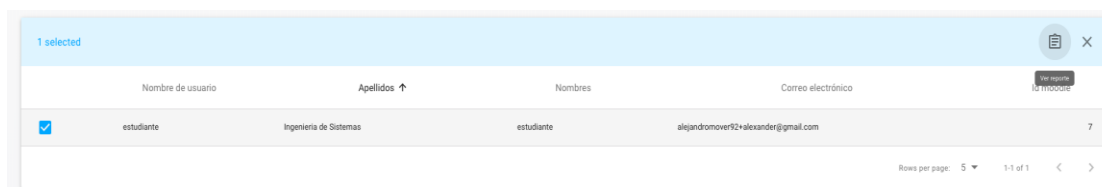
Para ver un reporte de proyectos, vaya a Laboratorio, seleccione un proyecto que NO se pueda modificar, y haga clic en ver reporte.



Nombre ↑	Descripción	Modificable	Actividad Id	Tests	Casos de prueba	Ver reporte	Número de entregas
<input checked="" type="checkbox"/> CasaDeCambioTest	CasaDeCambioTest	No	2	1	25		142

Figura 126 Seleccionar proyecto para ver reporte

Si desea ver un reporte de estudiante seleccione un estudiante de la tabla de estudiantes. Y haga clic en el botón ver reporte.



Nombre de usuario	Apellidos ↑	Nombres	Correo electrónico	Ver reporte
<input checked="" type="checkbox"/> estudiante	Ingenieria de Sistemas	estudiante	alejandromover92+alexander@gmail.com	

Rows per page: 5 1-1 of 1

Figura 127 Seleccionar estudiante para ver reporte

Si no hay reportes para la fecha seleccionada, verá el siguiente mensaje

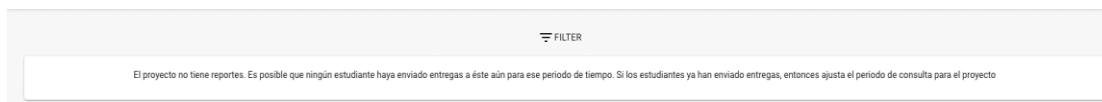


Figura 128 Vista de reportes si no hay reportes para mostrar

Por defecto, al entrar a la vista de proyectos, tomará los rangos de fecha según el semestre del sistema. Si desea cambiar el rango de fechas del proyecto haga clic en el botón filter en la parte superior de la ventana de proyectos, como se observa en la siguiente imagen



Figura 129 Botón de filtrar reportes

Al hacer clic se abrirá la ventana de filtro de reportes.

### *Filtrar por semestre*

Figura 130 Cuadro de dialogo filtrar reportes

1. Se seleccionó filtrar por reporte, desactívelo para filtrar por rango de fechas
2. Seleccionar primer o segundo semestre
3. Seleccionar el año del reporte
4. Seleccionar reporte por tópicos del proyecto / estudiante

### *Filtrar por rango de fechas*

**Create project report**

Custom range dates **1**

From **2**  
02/05/2020

From **3**  
02/05/2020

Select topic **4**

CANCELAR OK

Figura 131 cuadro de dialogo filtrar reporte por fechas

Se seleccionó generar reportes por rango de fecha

1. Fecha de inicio del reporte
2. Fecha de fin del reporte
3. Seleccionar por tópicos

Si encuentra reportes para las fechas indicadas podrá ver las siguientes pestañas

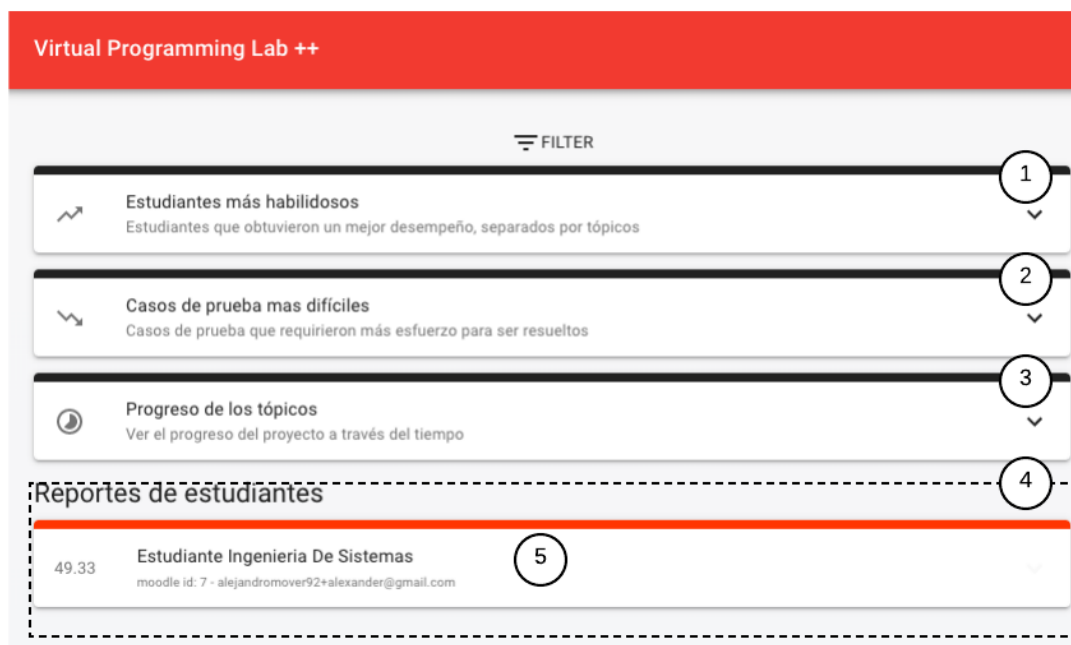


Figura 132 Vista reportes

1. Pestaña de estudiantes más habilidosos, es decir, quienes tuvieron un mejor desempeño por tópico
2. Pestaña de casos de pruebas más difíciles, es decir, los que requirió más esfuerzo de los estudiantes del reporte para pasarlo.
3. Progreso de los tópicos en esta pestaña se puede generar una línea de tiempo del progreso de los tópicos, proyectos o desempeño general.
4. Sección de los reportes de los estudiantes
5. Reporte de un estudiante

### *Reporte de estudiante*

Al hacer clic sobre la pestaña de reporte de un estudiante, está despliega una tabla con información relevante

49.33 Estudiante Ingeniería De Sistemas moodle id: 7 - alejandromover92+alexander@gmail.com							
Tópico	El estudiante conoce	(%) Habilidad	Esfuerzo	Casos aprobados	Casos No Aprobados	Total	Penalización
t006	Manejo de precedencia de operadores	100%	5	5	0	5	1.00
t001	Manejo de estructuras de control	60.99%	56	36	2	38	1.05
t0028	Uso de Listas	49.1%	101	58	10	68	1.17
t0044	Uso de Herencia	43.22%	59	33	10	43	1.29
t009	Temas básicos de Java	35.63%	363	142	14	156	1.10
t005	Manejo de operadores aritméticos	29.16%	254	76	2	78	1.03
t000	Construir clases	27.2%	243	68	2	70	1.03

Figura 133 Pestaña Reporte de usuario

Cada una de las filas está relacionada con un tópico. Si hace clic en alguna de las filas podrá ver más información sobre las ejecuciones de los casos de prueba del tópico

49.33 Estudiante Ingeniería De Sistemas moodle id: 7 - alejandromover92+alexander@gmail.com							
Tópico	El estudiante conoce	(%) Habilidad	Esfuerzo	Casos aprobados	Casos No Aprobados	Total	Penalización
t006	Manejo de precedencia de operadores	100%	5	5	0	5	1.00
t001	Manejo de estructuras de control	60.99%	56	36	2	38	1.05

Test Cases							
	<b>testIngredientesInsuficientesSimpleBlanco</b>	Description not provided					
	Submission not passed	2019-09-22T00:00:00.000Z					
	Submission not passed	2019-09-22T00:00:00.000Z					
	Submission not passed	2019-09-22T00:00:00.000Z					
	Submission not passed	2019-09-22T00:00:00.000Z					
	Submission passed	2019-09-22T00:00:00.000Z					
	Submission not passed	2019-09-22T00:00:00.000Z					
	<b>testRegistrarVentaPanBlanco</b>	Description not provided					
	<b>testRegistrarVentaJamon</b>	Description not provided					

Figura 134 Información adicional del reporte de usuario

## *Progreso de los tópicos a través del tiempo*

Esta opción le permitirá ver el desempeño del proyecto y/o de los estudiantes durante un periodo de tiempo.

En la ventana de reportes haga clic sobre la pestaña “Progreso de los tópicos”



Figura 135 Pestaña progreso de los tópicos

Verá el siguiente formulario, este formulario le permitirá generar un gráfico para ver el desempeño del proyecto o del estudiante en una serie de tiempo, según proyectos, o tópicos.

Figura 136 Pestaña progreso de los tópicos expandida

1. Generar línea de tiempo por tópico.
2. Seleccionar todos los tópicos

3. Seleccionar proyectos para compararlos
4. Seleccionar todos los proyectos
5. Fecha inicial de la línea de tiempo
6. Seleccionar la frecuencia de tiempo de la toma de la muestra durante la línea de tiempo, puede ser diario, semanal, mensual o anual, por ejemplo
7. Seleccionar cada cuanto se tomará la muestra, por ejemplo, cada dos meses, cada 1 mes, cada 6 días, etc.
8. Seleccionará la frecuencia de toma de la muestra, es decir si selecciona 4, tomará cuatro veces la muestra. En el filtro de la imagen, se puede apreciar que se tomará 4 muestras, cada 6 meses, desde el primero de junio del 2019
9. Activar para borrar los filtros al finalizar
10. Activar para no borrar la línea de tiempo anterior, sino que superponerla
11. Borrar, elimina la línea de tiempo actual
12. Visualizar, crea el gráfico de la línea de tiempo

Al hacer clic en visualizar podrá ver la gráfica generada





Figura 137 Gráfico progreso del estudiante

1. Si está activo, no muestra información vacía
2. Leyendas, son las leyendas de la gráfica
3. Gráfico

Como podrá observar, en el gráfico anterior, el estudiante mejoró su desempeño durante 3 semestres.

## 5.8.4 Administradores de Moodle

Después de iniciar sesión, el administrador verá su dashboard en su pestaña de aplicaciones

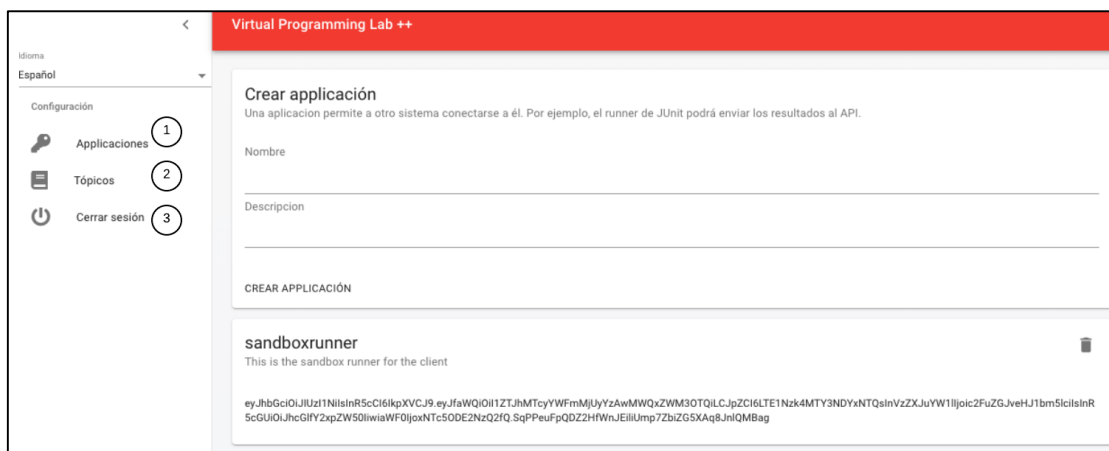


Figura 138 Administrador ventana inicial

1. Administrar aplicaciones
2. Administrar tópicos
3. Cerrar sesión

## *Administrar Aplicaciones*

En la vista de administrar aplicaciones, podrá configurar el acceso a VPL ++ desde aplicaciones externas, por defecto solo podrá crear accesos al JLib Runner.

Nota: Por defecto los token generados por VPL ++ nunca expiran, ese el administrador quien debe cambiarlos constantemente. Esos token son privados.

## Crear aplicación

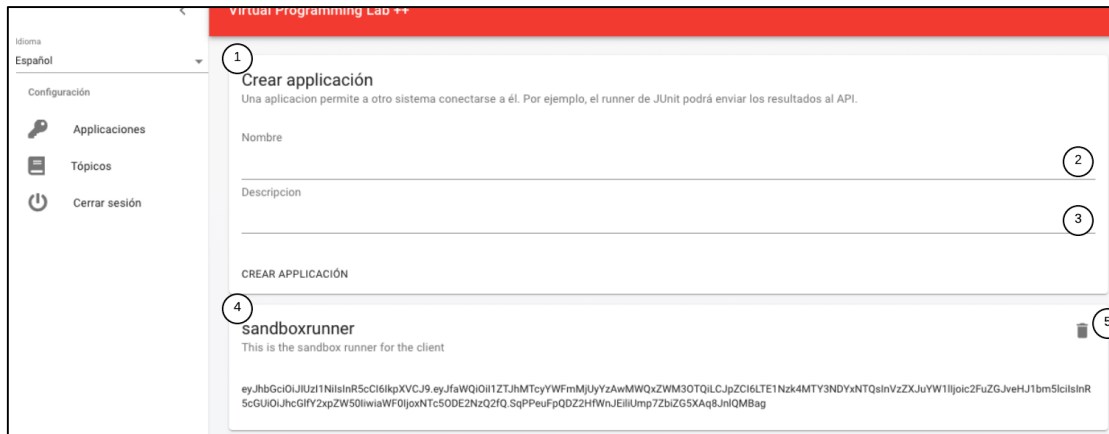


Figura 139 Vista de administrar aplicaciones

1. Formulario para crear aplicación
2. Nombre de la aplicación
3. Descripción de la aplicación
4. Vista de token de acceso a aplicación externa
5. Revocar/eliminar acceso a aplicación externa
6. Eliminar Aplicación

Solo haga clic en el botón eliminar aplicación. Si usted elimina el token, la aplicación externa perderá el acceso a VPL ++.

## Administrar tópicos

Los tópicos son objetivos específicos y cada caso de prueba está asociado a uno. Esto sirve para medir el desempeño de los estudiantes y proyectos.

Nota: no podrá modificar el nombre o la descripción de los tópicos. Deberá eliminar y crear uno nuevo.

## Crear Tópico

**1** Crear nuevo tópico

Un tópico se relaciona a un caso de prueba. Esto es usado para trazar y cuantificar el progreso de un estudiante respecto a una temática o algo que se desee evaluar. Por ejemplo: 'operadores aritméticos'

Nombre

Descripción

CREAR TÓPICO

**4** Tópicos registrados

	Id	Nombre ↑	Descripción	Tiene entregas
<input type="checkbox"/>	5e24dd6a6170a20029b73991	t000	Construir clases	true
<input type="checkbox"/>	5e24dd6a6170a20029b73989	t001	Manejo de estructuras de control	true
<input type="checkbox"/>	5e24dd6a6170a20029b73993	t0010	Temas intermedios de java	false
<input type="checkbox"/>	5e24dd6a6170a20029b73985	t0011	Temas avanzados de java	false
<input type="checkbox"/>	5e24dd6a6170a20029b7398b	t0012	Encapsulamiento	false

Rows per page: 5 1-5 of 46

Figura 140 Vista Administrar tópicos

1. Formulario para crear un nuevo tópico
2. Nombre del tópico
3. Descripción del tópico
4. Tabla de tópicos

El nombre del tópico deberá ser corto, use el campo descripción para describir el objetivo del tópico, este campo es el que será mostrado al profesor.

## *Eliminar tópico*

Solo podrá eliminar los tópicos que nunca han sido usados o cuyos casos de prueba NO tienen entregas.

Si desea eliminar un tópico pida a los profesores que eliminen cada uno de los proyectos que han sido resueltos y que usan el tópico que desea eliminar.

Para eliminar el tópico selecciónelo



1 selected				
id	Nombre ↑	Descripción	Tiene entregas	
<input checked="" type="checkbox"/>	5e24dd6a6170a20029b73991	1000	Construir clases	true

Figura 141 Vista seleccionar tópico

Haga clic en el primer botón (de izquierda a derecha). Si el tópico tiene entregas **NO** podrá eliminarlo y mostrará el siguiente mensaje si lo intenta:

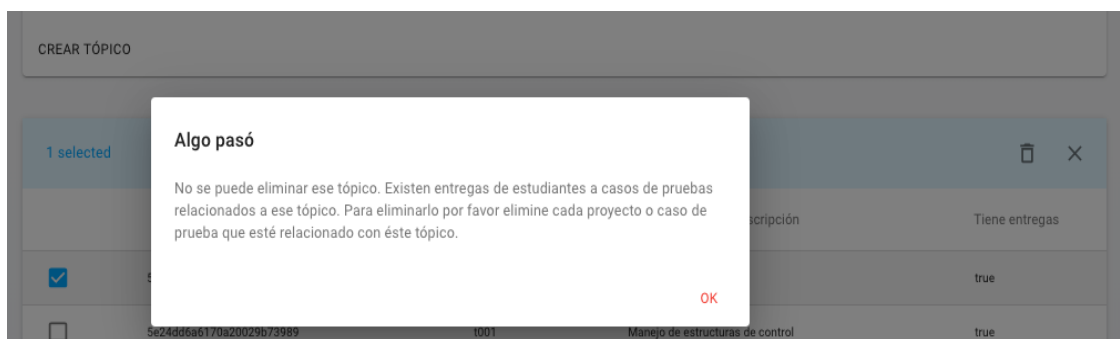


Figura 142 Vista error al eliminar tópico

Si el t3pico no tiene entregas, podr3 eliminarlo satisfactoriamente.

## 5.8.5 Recursos adicionales

### *Profesor*

#### *Administraci3n de Proyectos*

1. Crear un Proyecto en VPL ++

<https://www.youtube.com/watch?v=0Oa3lQuSGhI>

2. Crear test

[https://www.youtube.com/watch?v=I\\_qUObPjN3g](https://www.youtube.com/watch?v=I_qUObPjN3g)

3. Configurar test VPL ++

<https://www.youtube.com/watch?v=pKDLLzcAiaw>

4. Crear caso de prueba a test de VPL ++

<https://www.youtube.com/watch?v=6eFDGPOhkBE>

5. Exportar proyecto de VPL ++ a Moodle

<https://www.youtube.com/watch?v=tvJLJZLkm6Q>

6. Cargar proyecto de VPL ++ a Moodle

<https://www.youtube.com/watch?v=IV5kvPnBrmc>

7. Vincular proyecto de VPL ++ a la actividad

<https://www.youtube.com/watch?v=LQ3lSjm1-I0>

8. Descargar proyecto de VPL ++

<https://www.youtube.com/watch?v=UqruLYsoyYs>

9. Cargar archivo de proyecto de VPL ++ a VPL ++

<https://www.youtube.com/watch?v=44R-yhiObdc>

10. Mensaje de error cuando la actividad de VPL no est3 vinculada a proyecto de VPL ++ <https://www.youtube.com/watch?v=rT1StjzJCQs>

11. Como agregar ArrayList y otras Apis de java a los tests de VPL ++ generados por VPL ++

<https://www.youtube.com/watch?v=iQid7gZHLM4>

12. El profesor puede probar VPL ++ sin crear información en la base de datos

<https://www.youtube.com/watch?v=YiF3PtWKVRk>

## *Reportes*

El video siguiente muestra cómo generar reportes por proyectos y estudiantes:

<https://www.youtube.com/watch?v=PMFGhQACVbM>

## *Administrador*

### *Administrar tokens para usar el runner de VPL ++ JLib*

1. Crear token <https://youtu.be/tT0wkENBPok>
2. Eliminar token [https://youtu.be/Of\\_fhOHtAhg](https://youtu.be/Of_fhOHtAhg)

### *Administrar tokens para usar el runner de VPL ++ JLib*

1. Crear Tópico <https://www.youtube.com/watch?v=0u5jJQgBU5E>
2. Eliminar Tópico

<https://www.youtube.com/watch?v=0u5jJQgBU5E>

El administrador solo podrá eliminar un tópico solo si nadie ha resuelto un caso de prueba asociado a este. Éste error se puede observar en el siguiente video :

<https://www.youtube.com/watch?v=8zNPrvUGVko>

## 5.8.6 F.A.Q

El siguiente link contiene las respuestas a preguntas o errores comunes:

[https://github.com/alphonse92/VPLplusplus\\_composer/blob/master/faq.md](https://github.com/alphonse92/VPLplusplus_composer/blob/master/faq.md)



## 6. Prueba piloto

Desde el año 2017, Virtual Programming Lab se ha estado usando en las materias de Fundamentos de Programación y en Programación Orientada a Objetos I, y existe una gran cantidad de datos que no fueron procesados y que son difíciles de procesar, ya que VPL no genera reportes que permitan al profesor observar el desempeño de los estudiantes y las actividades a través del curso.

Usando las actividades que se tomaron para el caso de estudio, se procesaron las entregas enviadas por los estudiantes antes de implementar VPL.

Se tomaron en cuenta 3 variables:

1. Total, de estudiantes: 45
2. Total, de entregas de las actividades del caso de estudio: 664
3. Total, de actividades, es decir nueve, las que se tomaron para el caso de estudio
4. Calificación promedio de las actividades

### 6.1 Selección de las actividades a comparar para el caso de estudio

Las actividades se seleccionaron de manera semi-aleatoria, por criterio propio y según la dificultad del ejercicio o la cantidad de tests en cada una de ellas.

Se seleccionaron las actividades por los siguientes criterios.

1. Presentaba un nivel de dificultad medio, según la unidad.
2. Eran actividades en clase, es decir, cuyas pruebas no estaban ofuscadas.
3. Fueron desarrolladas en un ambiente controlado por el profesor, es decir, fueron resueltas en clase durante la supervisión del profesor.

De la unidad 1 se seleccionaron:

1. Tarea 1 - Ejercicio 8 - Casa de Cambio Tío Rico VPLLaboratorio virtual de programación
2. Tarea 1 - Ejercicio 10 - Máquina Cafetera VPLLaboratorio virtual de programación
3. Tarea 5 - Calculadora de Fraccionarios en VPLLaboratorio virtual de programación

De la unidad 2 se seleccionaron:

1. Quizá Trabajo en ClaseLaboratorio virtual de programación
2. Tarea 8 - Calculadora de Horas
3. Tarea 11 - Máquina de Sandwich Reloaded

De la unidad 3 se seleccionaron:

1. Otro reto para estudiar segundo previo

2. Reto ejemplo segundo previo
3. Segundo previo

En promedio, por actividad se enviaron 73.77 entregas, de las cuales la información que se obtiene es si el estudiante pasó la prueba o no.

Primero se analizó las calificaciones promedio a través del tiempo del curso.

Tabla 9 Calificaciones promedio de las actividades

Promedio	84.22
Punto máximo	96.22
Punto Mínimo	71.67
Desviación estándar	8.73
Límite máximo	92.95
Límite mínimo	75.49

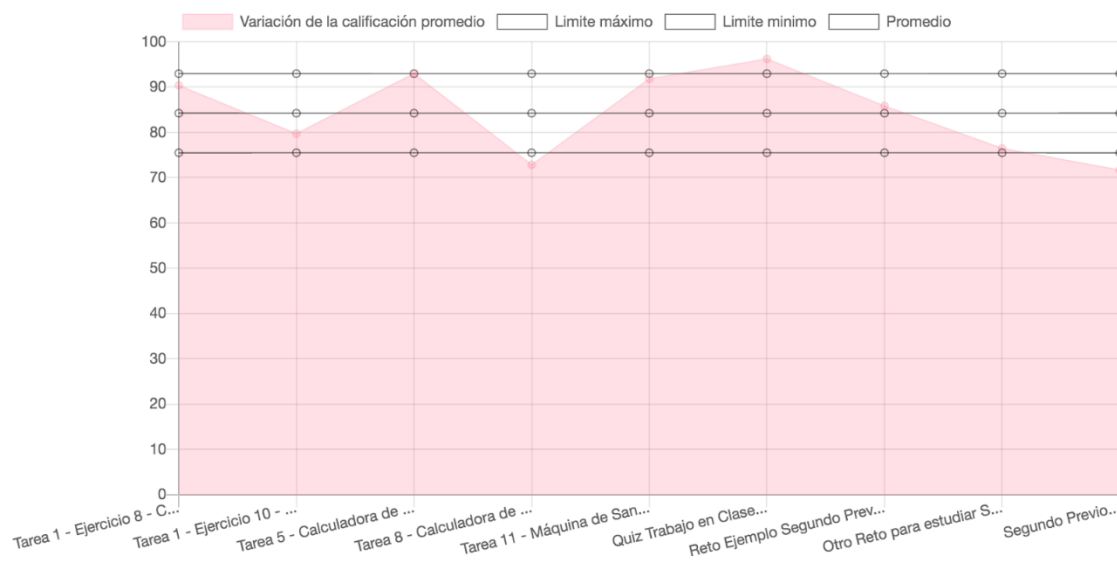


Figura 143 Gráfico de calificaciones de Moodle promedio

Se puede observar que las calificaciones tienden a subir y bajar abruptamente hasta la tercera actividad de la Unidad 2, desde ahí empieza a descender de manera constante, hasta su punto más bajo, se evidencia que los ejercicios aumentaron su dificultad en la última unidad si se compara con las pruebas unitarias en VPL de la Uvirtual.

Sin embargo, no es concluyente, considere los siguientes datos sobre el esfuerzo del estudiante.

Tabla 10 Esfuerzo promedio

Promedio	3.77
Punto máximo	8.00
Punto Mínimo	2.13
Desviación estándar	1.79
Límite máximo	5.56

Límite mínimo	1.98
---------------	------

Donde el esfuerzo es la cantidad de intentos promedio (entregas promedio) de los estudiantes para alcanzar la calificación promedio

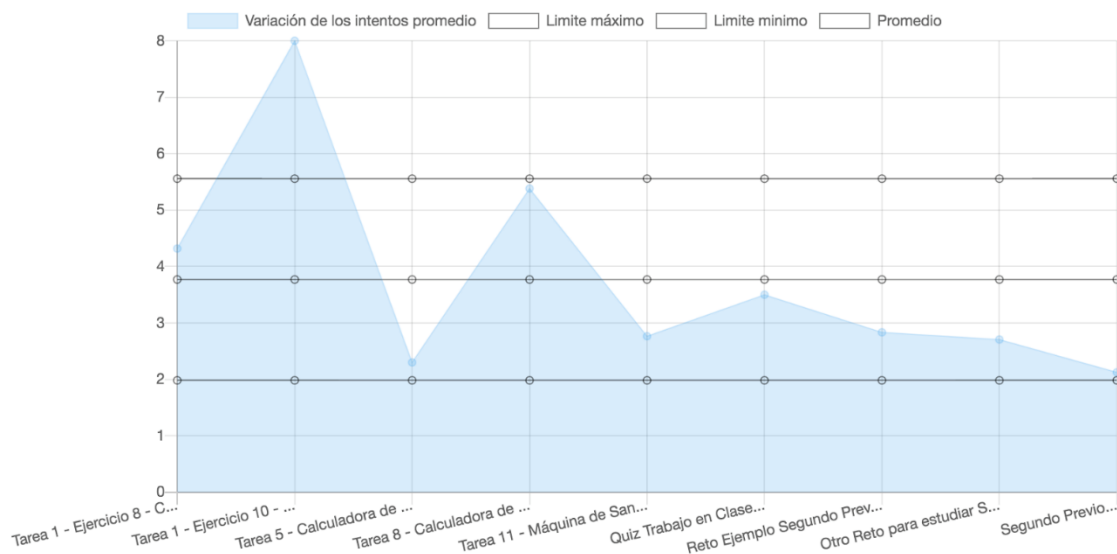


Figura 144 Grafica de esfuerzo promedio

Se puede apreciar que existen variaciones muy similares entre el esfuerzo promedio y la calificación promedio en cada una de las actividades.

Finalmente se analizó la cantidad de estudiantes que participaron (es decir, quienes enviaron al menos una entrega) en las actividades.

Tabla 11 Participación promedio

Promedio	19.56
Punto máximo	24.00
Punto Mínimo	17.00
Desviación estándar	2.27
Límite máximo	21.82
Límite mínimo	17.29

La gráfica obtenida fue la siguiente:

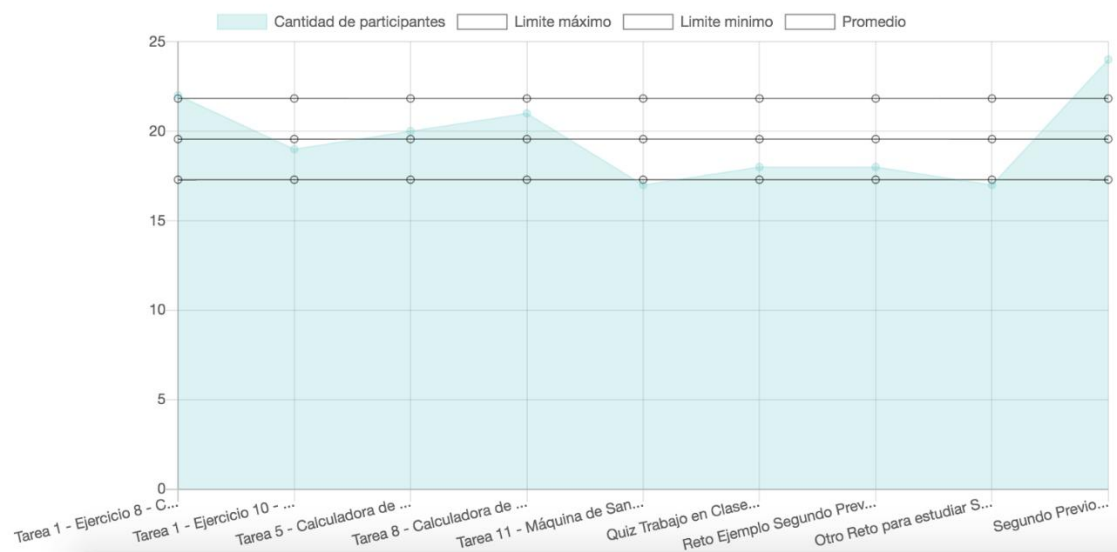


Figura 145 Gráfico de participación promedio

Lo anterior nos permite suponer que:

1. A menor esfuerzo, menor calificación.

2. Los estudiantes al principio del semestre muestran más dificultad en algunos temas, lo que explicaría las abruptas subidas y bajadas
3. Se obtienen calificaciones más estables cuando se mantiene el nivel de dificultad de las pruebas, esto se deja apreciar en las calificaciones de la unidad III en comparación con sus pruebas unitarias.
4. La participación se dispara en los puntos de corte

Sin embargo, estas estadísticas no permiten saber si el estudiante está avanzando o retrocediendo. Considere las siguientes interrogantes:

1. ¿Los estudiantes tienen un buen desempeño usando herencia?
2. ¿Los estudiantes pueden identificar los casos en los cuales pueden usar herencia?
3. ¿Los estudiantes entienden más listas que vectores?

La información actual no es lo suficientemente específica para responder las preguntas anteriores. Además, es difícil de obtener, ya que VPL no genera estos reportes, el profesor deberá extraer esta información a mano, y ésta solo nos permite especular. El objetivo de VPL ++ es generar reportes adicionales que ayuden a resolver las interrogantes anteriores bajo un criterio objetivo, cuantitativo y medible.

A continuación, describiremos los resultados obtenidos al seleccionar un estudiante, ejecutar las pruebas unitarias usando VPL ++ y generar los reportes.

## 6.2 Selección del estudiante para aplicar las pruebas con VPL ++ en el caso de estudio

Se seleccionó a un estudiante, la razón de selección de este estudiante fue que tuvo una gran cantidad de entregas la primera tarea (esfuerzo) pero el número de entregas para pasar las actividades fue disminuyendo sustancialmente a medida que resolvió actividades, esto es muy importante porque me permitió ver el rendimiento del estudiante a lo largo del semestre 2 del 2019 en el curso de Programación Orientada a Objetos I.

### 6.2.1 Consideraciones

#### *Consideraciones generales*

1. Moodle VPL califica la actividad en el sistema Moodle, VPL ++ NO califica, solo evalúa el desempeño del estudiante en una serie de temas o tópicos.
2. La calificación del estudiante depende del profesor y del estudiante.
3. VPL ++ No tomará en cuenta si el estudiante envió o no respuesta a una actividad, sólo leerá los resultados de las ejecuciones de las respuestas de las actividades de los estudiantes, por este motivo el “score” de un estudiante no bajará si no envía ninguna actividad.
4. Es responsabilidad del profesor leer e interpretar los resultados.
5. Las clases de pruebas originales (no VPL ++ ) no fueron desarrolladas para medir ciertos aspectos del conocimiento del estudiante, no tienen un objetivo específico a evaluar.



6. Debido al punto anterior, los tópicos seleccionados para cada caso de prueba fueron seleccionados según el criterio personal del desarrollador de VPL++ basado en el código entregado como muestra al estudiante.
7. Se tomaron las actividades comprendidas en el semestre 2 del año 2019.
8. Se mantuvo la fecha exacta de las entregas del estudiante, pero no las horas.
9. La ejecución de las actividades de VPL ++ fue hecha en una instancia de Moodle diferente a la de la Uvirtual

### *Consideraciones al profesor*

1. Continuar calificando al estudiante con las notas generadas con Moodle.
2. Basar su enfoque metodológico usando los resultados de VPL ++.

VPL++ puede indicarle al profesor si la metodología empleada es (o no) la adecuada, podrá ver con ésta herramienta el progreso de sus estudiantes bajo unos parámetros.

## 6.3 Ejecución de la prueba piloto

El análisis comparativo duró 44 Horas durante una semana. Se siguieron los siguientes pasos después de seleccionar la muestra de actividades y estudiantes:

1. Se descargaron las pruebas unitarias de JUnit de los archivos de ejecución de cada actividad de la muestra.

2. Por cada actividad de VPL de la muestra, se transformó en un proyecto de VPL ++
3. Se creó el proyecto como se puede apreciar en este video <https://youtu.be/0Oa3lQuSGhI>
4. Se crearon los tests por cada clase de prueba, como se puede apreciar en este video: [https://youtu.be/I\\_qUObPjN3g](https://youtu.be/I_qUObPjN3g) (se omitió las clases “IntegralTest” pues será el runner de VPL ++ quien maneje la ejecución de los tests)
5. Por cada método de prueba de JUnit (estos tienen el decorador @test ), se creó y configuró un caso de prueba para la prueba, como se puede apreciar en este video: <https://youtu.be/6eFDGPOhkBE>
6. Después de creado el proyecto se descargó como un archivo de restauración de una actividad de Moodle. Tal cual se puede apreciar en este video: <https://youtu.be/tvJLJZLkM6Q>
7. Este archivo fue usado para subir la actividad de VPL ++ a Moodle, como se puede apreciar en este video <https://youtu.be/IV5kvPnBrmc>
8. Después de haberse importado a Moodle, se vinculó la actividad de VPL ++ a la actividad de Moodle: <https://youtu.be/LQ3lSjm1-I0>
9. Algunas clases de VPL++ fueron modificadas para importar librerías como ArrayList <https://youtu.be/iQid7gZHLm4>
10. Se descargaron las entregas del estudiante que se tomó como muestra, para cada actividad de VPL que se tomaron en la muestra de actividades.
11. Se inició sesión como el estudiante en Moodle VPL ++, y se subieron una a una las entregas a cada una de las actividades correspondientes.

## 6.4 Resultados de la prueba piloto

### 6.4.1 Nivel de habilidad

Al finalizar el segundo semestre del año 2019 en el curso Programación Orientada a Objetos I se obtuvo que el estudiante alcanzó un Nivel de habilidad: 49.33 de 100 posible.

Reportes de estudiantes

49.33 Estudiante Ingenieria De Sistemas moodle id: 7 - alejandromover92+alexander@gmail.com							
Tópico	El estudiante conoce	(%) Habilidad	Esfuerzo	Casos aprobados	Casos No aprobados	Total	Penalización
t006	Manejo de precedencia de operadores	100%	5	5	0	5	1.00
t001	Manejo de estructuras de control	60.99%	56	36	2	38	1.05
t0028	Uso de Listas	49.1%	101	58	10	68	1.17
t0044	Uso de Herencia	43.22%	59	33	10	43	1.29
t009	Temas basicos de java	35.63%	363	142	14	156	1.10
t005	Manejo de operadores aritméticos	29.16%	254	76	2	78	1.03
t000	Construir clases	27.2%	243	68	2	70	1.03

Figura 146 Reporte de habilidad del estudiante tomado en la prueba piloto

### 6.4.2 Progreso del estudiante a través del tiempo

Para el estudiante, se tomó una muestra del nivel de habilidad del estudiante desde 2019-07-01 cada mes durante 6 meses.

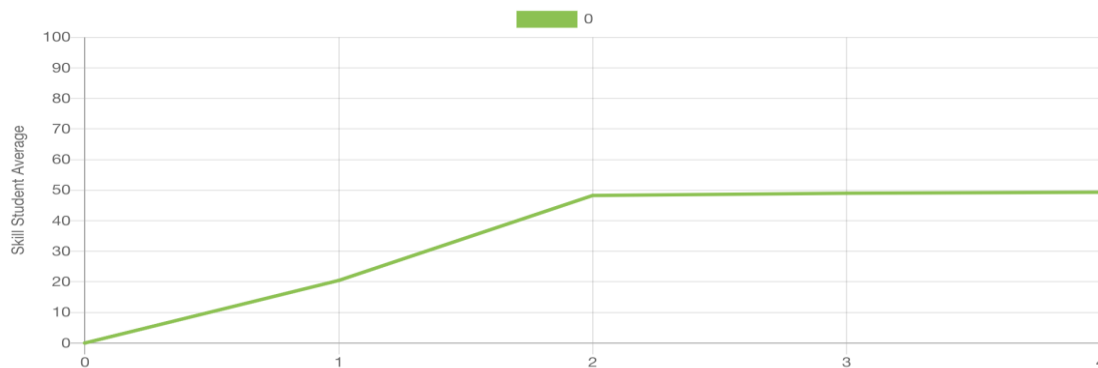


Figura 147 Progreso del estudiante a través del tiempo

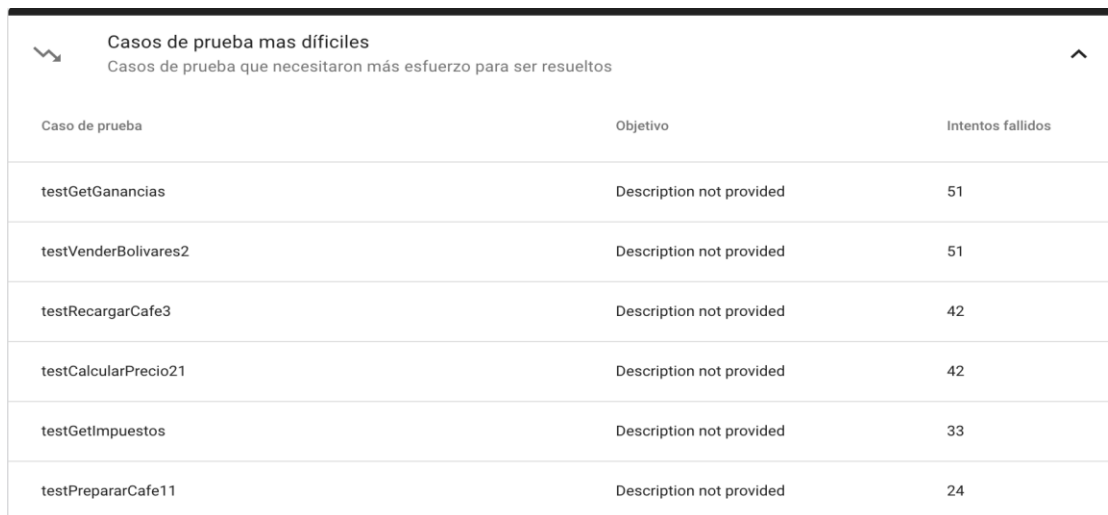
Tabla 12 Nivel de habilidad por mes del estudiante de la prueba piloto

Mes	Nivel de habilidad
Agosto	20.46
Septiembre	48.25
Octubre	49.00
Noviembre	49.33

Cabe resaltar que el estudiante mejoró significativamente su desempeño a medida que fue resolviendo actividades, a pesar de que el promedio de calificaciones de las actividades descritos en la figura 143 decreció con el tiempo.

### 6.4.3 Dificultad

Para el estudiante, los casos de prueba que fueron más difíciles fueron:



Caso de prueba	Objetivo	Intentos fallidos
testGetGanancias	Description not provided	51
testVenderBolivares2	Description not provided	51
testRecargarCafe3	Description not provided	42
testCalcularPrecio21	Description not provided	42
testGetImpuestos	Description not provided	33
testPrepararCafe11	Description not provided	24

Figura 148 casos de prueba que resultaron más difíciles de resolver para el estudiante de la prueba piloto

Pues requirieron hasta 51 intentos por superarlos.

1. El máximo de fallas requeridos para superar un caso de prueba fue 51
2. El mínimo de fallas requeridas para superar un caso de prueba fue 3

### 6.4.4 Tópicos

Se puede observar que el estudiante tiene gran control sobre el manejo de precedencia de operadores, mientras que le cuesta construir clases, según el reporte siguiente:



Tópico	Estudiante	Habilidad
Manejo de precedencia de operadores	estudiante Ingeniería de Sistemas	100.00
Manejo de estructuras de control	estudiante Ingeniería de Sistemas	60.99
Uso de Listas	estudiante Ingeniería de Sistemas	49.10
Uso de Herencia	estudiante Ingeniería de Sistemas	43.22
Temas básicos de java	estudiante Ingeniería de Sistemas	35.63
Manejo de operadores aritméticos	estudiante Ingeniería de Sistemas	29.16
Construir clases	estudiante Ingeniería de Sistemas	27.20

Figura 149 Nivel de habilidad del estudiante de la prueba piloto por tópico

Al finalizar el semestre, el estudiante obtuvo un nivel de 100 en el tópico de manejo de precedencia de operadores. Pero en construir clases obtuvo un nivel de 27.

Los tópicos variaron de la siguiente manera durante los meses del semestre

### Legendas

Numero	Proyectp	Tópicos
0	Todos los proyectos	Construir clases
1	Todos los proyectos	Manejo de precedencia de operadores
2	Todos los proyectos	Manejo de estructuras de control
3	Todos los proyectos	Topicos basicos de java
4	Todos los proyectos	Manejo de operadores aritméticos
5	Todos los proyectos	Uso de Herencia
6	Todos los proyectos	Uso de Listas

Figura 150 Legendas del progreso del nivel de habilidad de los tópicos del estudiante de la prueba piloto

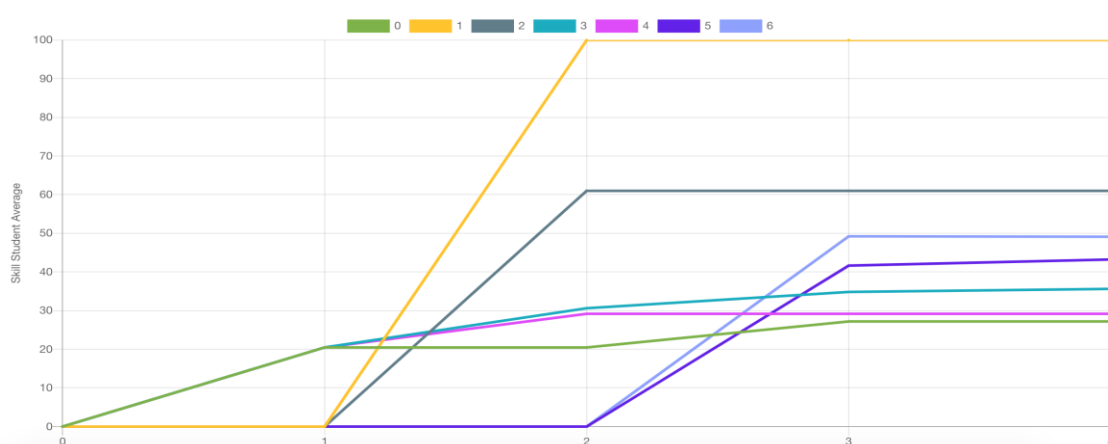


Figura 151 Gráfica del progreso del nivel de habilidad de los tópicos del estudiante de la prueba piloto

Finalmente, se logró evidenciar que el estudiante mejoró su desempeño sobre ciertos temas.

Se pudo obtener esta información gracias a que VPL++ hizo un seguimiento de las ejecuciones del estudiante y sus resultados a lo largo del curso. VPL ++ permite dar una idea del progreso del estudiante respecto a unos tópicos que evalúan los objetivos específicos de los casos de prueba.

VPL++ es una herramienta potente que permite hacer un seguimiento de las actividades y sus estudiantes, para poder medir su desempeño. Sus reportes especializados son fundamentales para poder entender el progreso de los estudiantes, a su vez, el profesor podrá ajustar la metodología de enseñanza.

Con VPL++ el programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, podrá generar una gran cantidad de datos sobre el desarrollo de las actividades de programación, y poder procesarlo incluso con herramientas más potentes de análisis de datos. Por otro lado, VPL ++ puede ser usado en cualquier materia de programación de computadores, de manera que se puede obtener estadísticas de un estudiante no solo a nivel de semestre sino a lo largo de la carrera, y usando técnicas de análisis de datos como Machine Learning, Deep Learning, entre otras, generar predicciones que permitan prever el comportamiento de los estudiantes que ayude a prevenir la deserción de los estudiantes.



# Conclusiones y recomendaciones

1. Virtual Programming Lab es un software que junto a Moodle es una herramienta que le ha permitido al profesor Milton Jesús Vera Contreras apoyar sus cursos de programación.

2. Evosuite es un software que permite generar pruebas unitarias automáticamente y puede ser usado de apoyo para crear pruebas unitarias para los programas de los estudiantes de los cursos de programación de computadores.

3. Virtual Programming Lab ++ puede generar reportes especializados de las ejecuciones de las pruebas unitarias en VPL, éstas estadísticas pueden ser usadas por los profesores de materias de programación de computadores.

4. Aunque el estudiante tuvo un score bajo, logró resolver la mayoría de ejercicios por lo que no hay relación alguna entre un score alto y la capacidad de ejercicios que el estudiante resuelve.

5. El estudiante tuvo que hacer un mayor esfuerzo durante los dos primeros meses del semestre.

6. Las arquitecturas orientadas a microservicios son altamente escalables y desacopladas.

7. La calidad de los resultados entregados por VPL ++ dependerá de la calidad de las pruebas unitarias que el profesor plantee. El profesor deberá plantear pruebas unitarias con mayor especificidad para cada tópico que quiera evaluar.

8. VPL ++ genera reportes que el profesor deberá interpretar y asumir por cuenta propia.

9. Existen otras variables que se deben tener en cuenta para generar una mejor aproximación al nivel de habilidad de un estudiante, como pruebas de complejidad ciclométrica y otras variables

## Recomendaciones

1. Desarrollar aplicaciones para sistemas de administración de aprendizaje (LMS) tiene una aplicación e impacto favorable. En éste proyecto se logró conocer un poco al respecto y es recomendable que el Programa de Ingeniería de Sistemas agregue a su currículo el desarrollo de plugins, add-ons, e integraciones con Moodle y otras plataformas de aprendizaje actuales, como EDX, Coursera, Udemy, Platzi entre otros.
2. La metodología de desarrollo (resolución) de los ejercicios de VPL se basan en los principios del desarrollo dirigido por tests. Se sugiere que el programa de Ingeniería de Sistemas agregue en sus cursos de programación después de POO II temáticas que expliquen la importancia de las pruebas unitarias de software, y orientar a los estudiantes para que conozcan las ventajas de este tipo de estrategia de desarrollo (Edwards, 2003).
3. El desarrollo dirigido con tests ha ganado gran importancia ya que mejora la calidad de código (Jones, 2004). Se sugiere que sean en cursos después de POO II porque los estudios sugieren que el aprendizaje temprano de TDD sobrecarga la capacidad cognitiva del estudiante, tal cual lo concluye: “Test-Driven Development in Education: Experiences with Critical Viewpoints” (Kollanus & Isomöttönen, 2008).
4. En un futuro, extender las funcionalidades de VPL ++ como por ejemplo implementar la librería checkstyle, para poder calificar según otras métricas de código (Ohtsuki et al., 2016).
5. Se recomienda aplicar técnicas de predictivas usando Inteligencia Artificial que mejoren la retroalimentación.

# Anexos

La url para acceder a VPL++ se encuentra en el siguiente link:

[https://github.com/alphonse92/VPLplusplus\\_composer](https://github.com/alphonse92/VPLplusplus_composer)

## Repositorios

En los siguientes repositorios se puede encontrar el código de cada uno de los elementos de software creados, en ellos se encuentran las instrucciones (en inglés) de la instalación y configuración de cada uno de ellos.

Tabla 13 Repositorios

Nombre del repositorio	Descripción	Link
VPLplusplus_composer	Este repositorio es el compositor de los microservicios. En él se encuentran las instrucciones para la instalación y configuración del ecosistema de microservicios	<a href="https://github.com/alphonse92/VPLplusplus_composer">https://github.com/alphonse92/VPLplusplus_composer</a>
VPLplusplus_api	Repositorio de VPL ++ API	<a href="https://github.com/alphonse92/VPLplusplus_">https://github.com/alphonse92/VPLplusplus_</a>
VPLplusplus_client	Repositorio del frontend de VPL ++	<a href="https://github.com/alphonse92/VPLplusplus_client">https://github.com/alphonse92/VPLplusplus_client</a>
VPLplusplus_gateway	Repositorio del microservicio de Gateway.	<a href="https://github.com/alphonse92/VPLplusplus_gateway">https://github.com/alphonse92/VPLplusplus_gateway</a>
VPLplusplus_jail	Repositorio del código para crear la imagen de Docker de la jaula de ejecución con solo java	<a href="https://github.com/alphonse92/VPLplusplus_jail">https://github.com/alphonse92/VPLplusplus_jail</a>
VPLplusplus_jail_jlib	Repositorio del código para crear la imagen de Docker de la jaula de ejecución con solo java el programa	<a href="https://github.com/alphonse92/VPLplusplus_jail_jlib">https://github.com/alphonse92/VPLplusplus_jail_jlib</a>

	VPL ++ JLib Runner	
VPLplusplus_jlib	Repositorio del código de VPL ++ JLib Runner	<a href="https://github.com/alphonse92/VPLplusplus_jlib">https://github.com/alphonse92/VPLplusplus_jlib</a>
Docker-Moodle	Repositorio del código para crear la imagen de Moodle	<a href="https://github.com/alphonse92/Docker-Moodle">https://github.com/alphonse92/Docker-Moodle</a>

# Bibliografía

- Alhammad, S., Atkinson, S., & Stuart, L. (2016). The role of Visualisation in the study of Computer Programming. *27th Annual Workshop of the Psychology of Programming Interest Group - PPIG 2016*, 5–16.
- Andrés, J., Pacheco, C., Iván, C., Álvarez, O., Iván, J. C., Ibarra, G., Jesús, M., Contreras, V., Francisco, U., & Santander, D. P. (n.d.). *DOCKERIZANDO UN LABORATORIO VIRTUAL DE PROGRAMACIÓN ( VPL ) Y MOODLE EN GOOGLE*.
- Ariza, A. (2014). *EVALUACIÓN DE LOS ESTUDIANTES DE PRIMER A TERCER SEMESTRE DEL PROGRAMA INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER CÚCUTA EN RELACIÓN CON EL APRENDIZAJE DE LAS ÁREAS DE PROGRAMACIÓN*.
- Bashari Rad, B., John Bhatti, H., & Ahmadi, M. (2017). An Introduction to Docker and Analysis of its Performance. *IJCSNS International Journal of Computer Science and Network Security*, 17(3), 228–235.
- Bosse, Y., & Gerosa, M. A. (2016). Why is programming so difficult to learn? Patterns of difficulties related to programming language. *ACM SIGSOFT Software Engineering Notes*, 41(6), 1–6. <https://doi.org/10.1145/3011286.3011>
- Carlos, J., & Royo, E. R. (n.d.). *VPL : Laboratorio Virtual de Programación para Moodle*.
- Cerny, T., Donahoo, M. J., & Trnka, M. (2018). Contextual understanding of microservice architecture. *ACM SIGAPP Applied Computing Review*, 17(4), 29–45. <https://doi.org/10.1145/3183628.3183631>
- Chu, R. B. S. T. S. X. (2009). *Software Development and Object-Oriented Programming Paradigms*. New Delhi ; Singapore : Tata McGraw-Hill.
- Cosmina, I. (2017). Spring Microservices in Action. In *Manning*. <https://doi.org/10.1007/978-1-4842-0811-3>
- Douce, C., Livingstone, D., & Orwell, J. (2005). Automatic Test-Based Assessment of Programming: A Review. *ACM Journal on Educational Resources in Computing*, 5(3), 4. <https://doi.org/10.1145/1163405.1163409>

- Edith Lovos, A. G., & Inés. (n.d.). *Combinando ABP y Herramientas Colaborativas para la Enseñanza de Programación en el Primer Año de la Lic. en Sistemas de la UNRN*.
- Edwards, S. H. (2003). Improving Student Performance by Evaluating How Well Students Test Their Own Programs. *ACM Journal on Educational Resources in Computing*, 3(3), 1. <https://doi.org/10.1145/1029994.1029995>
- Esearch, S. Y. R., Hevner, B. A. R., March, S. T., Park, J., & Ram, S. (2004a). *DESIGN SCIENCE IN INFORMATION*. 28(1), 75–105.
- Esearch, S. Y. R., Hevner, B. A. R., March, S. T., Park, J., & Ram, S. (2004b). Design Science in Information System Research. *MIS Quarterly*, 28(1), 75–105.
- Fraser, G., & Arcuri, A. (2011). EvoSuite: Automatic test suite generation for object-oriented software. *SIGSOFT/FSE 2011 - Proceedings of the 19th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 416–419. <https://doi.org/10.1145/2025113.2025179>
- Heines, J. M. (1999). Evaluating course web sites and student performance. *SIGCSE Bulletin (Association for Computing Machinery, Special Interest Group on Computer Science Education)*, 31(3), 143–146. <https://doi.org/10.1145/384267.305898>
- Jones, C. (2004). Test-driven development goes to school. *Journal of Computing Sciences in Colleges*, 20(1), 220–231.
- Kollanus, S., & Isomöttönen, V. (2008). Test-driven development in education: Experiences with critical viewpoints. *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, 124–127. <https://doi.org/10.1145/1384271.1384306>
- Kölling, M. (1999). The problem of teaching object-oriented programming, part I: Languages. *JOOP - Journal of Object-Oriented Programming*, 11(8), 8–15.
- Milton-Jesús Vera-Contreras, & Herrera-Cáceres, M. (2017a). *Working Paper: Estado del Arte sobre educación en programación de computadores*. UFPS.
- Milton-Jesús Vera-Contreras, & Herrera-Cáceres, M. (2017b). *Working Paper: Uso de VPL (Laboratorio Virtual de Programación) en el curso de Fundamentos de Programación de la UFPS*. UFPS.
- Ohtsuki, M., Ohta, K., & Kakeshita, T. (2016). Software engineer education support system ALECSS utilizing devOps tools. *ACM International Conference Proceeding Series*, 209–213. <https://doi.org/10.1145/3011141.3011200>

- Papaspyrou, N. S., & Zachos, S. (2013). Teaching programming through problem solving: The role of the programming language. *2013 Federated Conference on Computer Science and Information Systems, FedCSIS 2013*, 1545–1548.
- Parham, J. R. (2003). An assessment and evaluation of Computer Science education. *Journal of Computing Sciences in Colleges*, 19(12), 115–127.  
[http://www.cs.clemson.edu/~jparham/parham\\_2003\\_publication.pdf](http://www.cs.clemson.edu/~jparham/parham_2003_publication.pdf)
- Patterson, A., Kölling, M., & Rosenberg, J. (2003). Introducing Unit Testing With BlueJ. *Proceedings of the Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiSCE)*, 8, 11–15. <https://doi.org/10.1145/961290.961518>
- Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, 2(2), 137–168. [https://doi.org/10.1016/0732-118X\(84\)90018-7](https://doi.org/10.1016/0732-118X(84)90018-7)
- Queirós, R., & Leal, J. P. (2012). Programming exercises evaluation systems: An interoperability survey. *CSEdu 2012 - Proceedings of the 4th International Conference on Computer Supported Education*, 1, 83–90. <https://doi.org/10.5220/0003924900830090>
- Ren, Z., Wang, W., Wu, G., Gao, C., Chen, W., Wei, J., & Huang, T. (2018). Migrating web applications from monolithic structure to microservices architecture. *ACM International Conference Proceeding Series*. <https://doi.org/10.1145/3275219.3275230>
- Rodríguez-del-Pino, J. C., Rubio-Royo, E., & Hernández-Figueroa, Z. (2012). A Virtual Programming Lab for Moodle with automatic assessment and anti-plagiarism features. *Conference on E-Learning, e-Business, Enterprise Information Systems, & e-Government*.
- Rosales Sales, Euline esperanza; Suarez García, German Darío; Velazco Sanchez, D. C. (n.d.). *estudio sobre desercion - retencion de los estudiantes del plan de estudios de ingenieria de sistemas de la universidad francisco de paula santander y perfil ocupacional del egresado*.
- Scherer, R. (2016). Learning from the past—the need for empirical evidence on the transfer effects of computer programming skills. *Frontiers in Psychology*, 7(SEP), 7–10.  
<https://doi.org/10.3389/fpsyg.2016.01390>
- Toward Unified Theory of Teaching and Learning Computer Programming : A Systematic Review of the Literature*. (2017).



- Vera-Contreras, M.-J., & Herrera-Cáceres, M. (2017). *Working Papers: Análisis del nivel de reprobación y promedio de calificación en los cursos de programación de computadores del programa de ingeniería de sistemas de la UFPS*.
- Vera, M. J. C., Cáceres, M. H., & Pérez, O. A. G. (2018). *Pruebas Automáticas para Evaluar Cursos de programación de Computadores*. TdeA.
- Villalobos, J. A., & Calderón, N. A. (2009). Proyecto Cupi2: un enfoque multidimensional frente al problema de enseñar y aprender a programar. *Revista de Investigaciones UNAD*, 8(2), 45. <https://doi.org/10.22490/25391887.635>
- Villalobos, J., Casallas, R., & Marcos, K. (2005). El reto de diseñar un primer curso de programación de computadores. *XIII Congreso Iberoamericano de Educación Superior En Computación, Cali, Colombia, 1*, 1–12. <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:El+Reto+de+Disenar+un+Primer+Curso+de+Programacion+de+Computadores#0>
- Villalobos, J., Casallas, R., Ph, D., Cupi, E., Cupi, E., Cupi, E., & Cupi, E. (n.d.). *El Reto de Enseñar a Programar Seguimiento Diagnóstico Diagnóstico primer curso Algunos números Marco conceptual*.
- Villamizar, M., Garces, O., Ochoa, L., Castro, H., Salamanca, L., Verano, M., Casallas, R., Gil, S., Valencia, C., Zambrano, A., & Lang, M. (2016). Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures. *Proceedings - 2016 16th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2016*, 179–182. <https://doi.org/10.1109/CCGrid.2016.37>
- Weinberg, G. M. (1985). *The Psychology of Computer Programming*. John Wiley & Sons, Inc.
- Wolff, E. (n.d.). *Microservices: Flexible Software Architectures*.