

**Construcción y Pruebas de Software**

IV Ciclo

Laboratorio N° 12

**“Pruebas de Integración”**

##### Laboratorio: Pruebas de Integración

##### OBJETIVOS

1. Identificar los componentes de a probar
2. Definir las pruebas de integración a realizar
3. Construir, ejecutar y documentar las pruebas de integración

**EQUIPOS, MATERIALES, PROGRAMAS Y RECURSOS**

* PC Personal.
* Sistema operativo Windows XP o superior
* Material disponible desde Tecsup Virtual.
* Conexión a red.

##### SEGURIDAD

* Colocar las mochilas en el gabinete al final del salón para evitar caídas en caso de sismo.
* No ingresar con bebidas ni comidas.
* Apagar los equipos y los monitores al culminar la sesión.

**INTRODUCCIÓN**

En los proyectos de software es necesario realizar pruebas de integración para garantizar la correcta integración entre los componentes. En el presente laboratorio se procede a realizar pruebas de integración con una aplicación web basada en el framework de Spring

**PREPARACIÓN**

El Alumno debe revisar previamente el material del curso en Tecsup Virtual y revisar su texto.

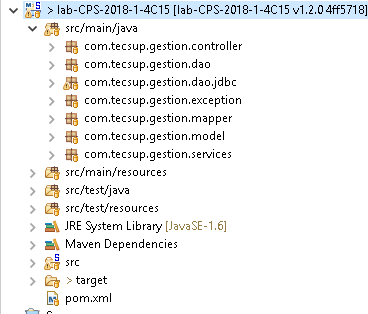
**PROCEDIMIENTO y RESULTADOS**

1. Bajar y desplegar correctamente la aplicación web en su IDE de STS, revisar el funcionamiento

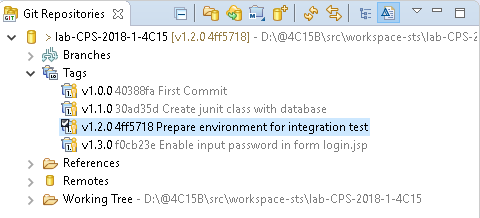
<https://github.com/jgomezm-tecsup/lab-CPS-2018-1-4C15>

versión : 1.2.0

**Proyecto clonado del GitHub:**

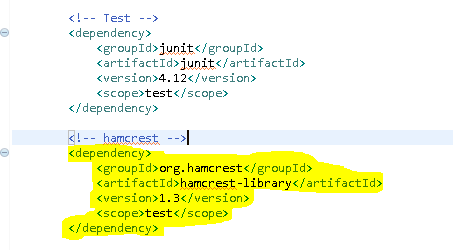


**Escogemos la version 1.2.0. en la sección de Git Repositories:**



1. La librería *Hamcrest* es un framework para testear software. Hamcrest permite realizar diferente tipo de condiciones, por ejemplo el método hasProperty, el cual permite validar si un elemento tiene una propiedad en concreto. Agregar la siguiente dependencia en maven (pom.xml) para el soporte de la librería *Hamcrest*

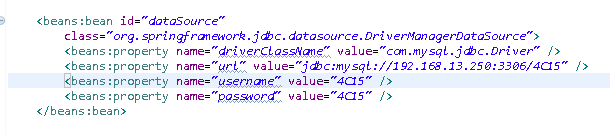
|  |
| --- |
| <!—Colocar después de las dependencias del JUNIT -->    <!-- hamcrest -->  <dependency>  <groupId>org.hamcrest</groupId>  <artifactId>hamcrest-library</artifactId>  <version>1.3</version>  <scope>test</scope>  </dependency> |

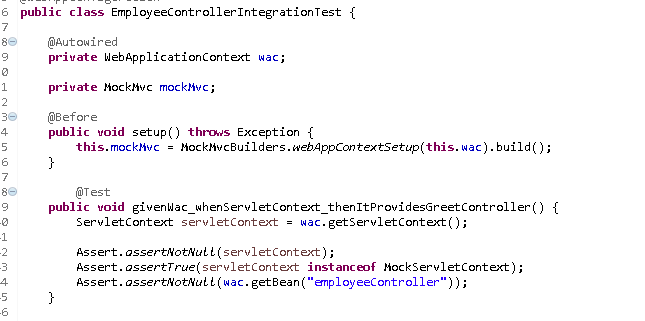


1. Crear una clase de pruebas de integración

Nombre de la clase: com.tecsup.gestion.controller.EmployeeControllerIntegrationTest

|  |
| --- |
| **package** com.tecsup.gestion.controller;  **import** **static** org.hamcrest.Matchers.\*;  **import** **static** org.springframework.test.web.servlet.request.MockMvcRequestBuilders.*get*;  **import** **static** org.springframework.test.web.servlet.result.MockMvcResultMatchers.\*;  **import** javax.servlet.ServletContext;  **import** org.junit.Assert;  **import** org.junit.Before;  **import** org.junit.Test;  **import** org.junit.runner.RunWith;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.mock.web.MockServletContext;  **import** org.springframework.test.context.ContextConfiguration;  **import** org.springframework.test.context.junit4.SpringJUnit4ClassRunner;  **import** org.springframework.test.context.web.WebAppConfiguration;  **import** org.springframework.test.web.servlet.MockMvc;  **import** org.springframework.test.web.servlet.setup.MockMvcBuilders;  **import** org.springframework.web.context.WebApplicationContext;  @RunWith(SpringJUnit4ClassRunner.**class**)  @ContextConfiguration("file:src/main/webapp/WEB-INF/spring/appServlet/servlet-context.xml")  @WebAppConfiguration  **public** **class** EmployeeControllerIntegrationTest {  @Autowired  **private** WebApplicationContext wac;  **private** MockMvc mockMvc;    @Before  **public** **void** setup() **throws** Exception {  **this**.mockMvc = MockMvcBuilders.*webAppContextSetup*(**this**.wac).build();  }    @Test  **public** **void** givenWac\_whenServletContext\_thenItProvidesGreetController() {  ServletContext servletContext = wac.getServletContext();    Assert.*assertNotNull*(servletContext);  Assert.*assertTrue*(servletContext **instanceof** MockServletContext);  Assert.*assertNotNull*(wac.getBean("employeeController"));  }  } |

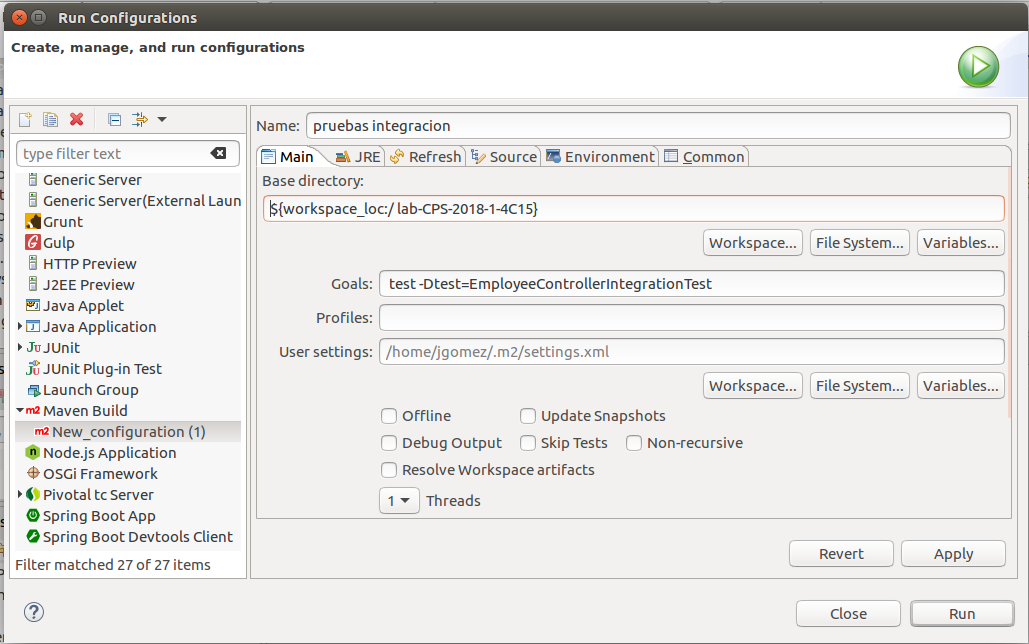


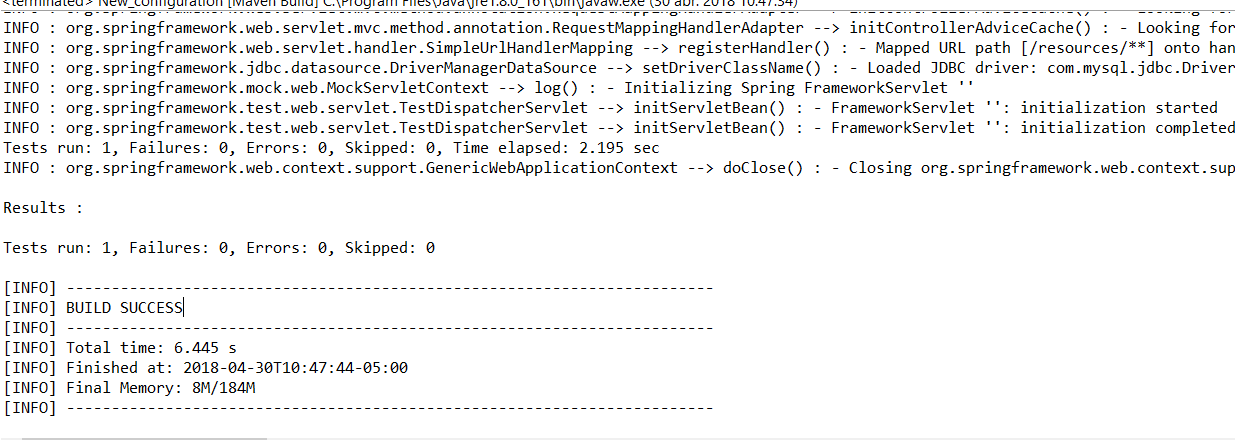


* 1. Ejecutar la prueba con maven

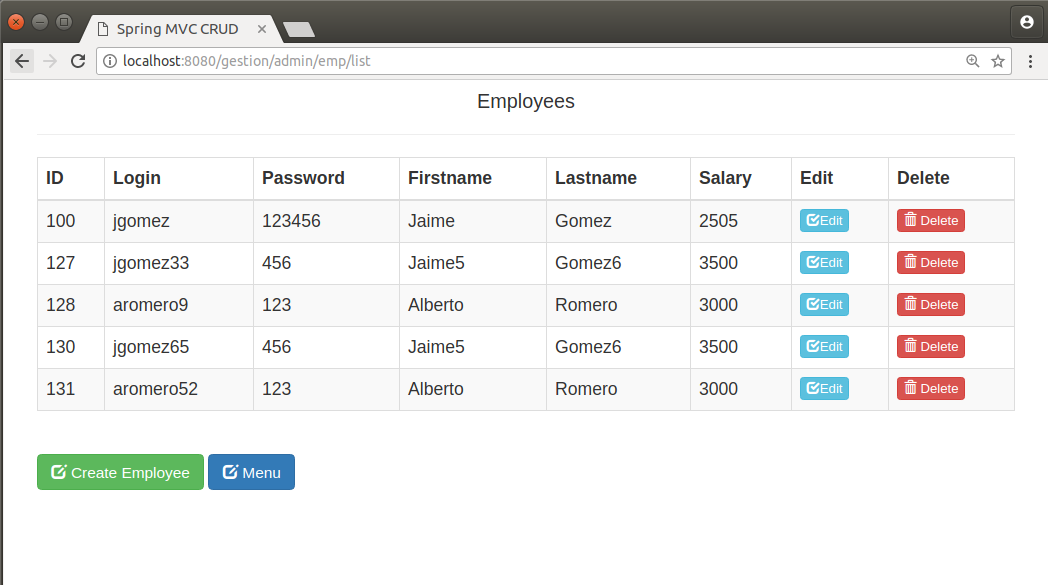
|  |
| --- |
| mvn test -Dtest=EmployeeControllerIntegrationTest |

Como alternativa puede usar la prueba directamente desde el IDE STS





1. Ejecutar la funcionalidad de mostrar todos los empleados por pantalla, previamente se deberá desplegar la aplicación en el contenedor web



* 1. Agregar el siguiente método

|  |
| --- |
| @Test  **public** **void** list() **throws** Exception {  mockMvc.perform(*get*("/admin/emp/list"))  .andExpect(*status*().isOk())  .andExpect(*view*().name("admin/emp/list"))  .andExpect(*forwardedUrl*("/WEB-INF/views/admin/emp/list.jsp"))  .andExpect(*model*().attribute("employees", *hasSize*(43)))  .andExpect(*model*().attribute("employees",  *hasItem*(*allOf*(  *hasProperty*("employeeId", *is*(100)),  *hasProperty*("login", *is*("jgomez")),  *hasProperty*("password", *is*("123456")),  *hasProperty*("firstname", *is*("Jaime")),  *hasProperty*("lastname", *is*("Gomez")),  *hasProperty*("salary", *is*(2505))  ))));  } |

* 1. Ejecutar la prueba con maven

|  |
| --- |
| mvn test -Dtest=EmployeeControllerIntegrationTest |

1. Agregar la funcionalidad de edición de registros

EmployeeController.java

|  |
| --- |
| **package** com.tecsup.gestion.controller;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Controller;  **import** org.springframework.ui.ModelMap;  **import** org.springframework.web.bind.annotation.GetMapping;  **import** org.springframework.web.bind.annotation.ModelAttribute;  **import** org.springframework.web.bind.annotation.PathVariable;  **import** org.springframework.web.bind.annotation.PostMapping;  **import** org.springframework.web.servlet.ModelAndView;  **import** com.tecsup.gestion.model.Employee;  **import** com.tecsup.gestion.services.EmployeeService;    /\*\*  \* Handles requests for the application home page.  \*/  @Controller  **public** **class** EmployeeController {    **private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(EmployeeController.**class**);    @Autowired  **private** EmployeeService employeeService;  ...............  ...............  ...............  @GetMapping("/admin/emp/{action}/{employee\_id}")  **public** ModelAndView form(@PathVariable String action, @PathVariable **int** employee\_id, ModelMap model) {  // action = {"editform","deleteform"}  ***logger***.info("action = " + action);  ModelAndView modelAndView = **null**;  **try** {  Employee emp = employeeService.find(employee\_id);  ***logger***.info(emp.toString());  modelAndView = **new** ModelAndView("admin/emp/" + action, "command", emp);  } **catch** (Exception e) {  model.addAttribute("message", e.getMessage());  modelAndView = **new** ModelAndView("admin/emp/" + action, "command", **new** Employee());  }  **return** modelAndView;  }    @PostMapping("/admin/emp/editsave")  **public** ModelAndView editsave(@ModelAttribute("SpringWeb") Employee emp, ModelMap model) {  ***logger***.info("emp = " + emp);    ModelAndView modelAndView = **null**;  **try** {  employeeService.update(emp.getLogin(), emp.getPassword(), emp.getFirstname(), emp.getLastname(),  emp.getSalary(), -1);  modelAndView = **new** ModelAndView("redirect:/admin/emp/list");  } **catch** (Exception e) {  model.addAttribute("message", e.getMessage());  modelAndView = **new** ModelAndView("redirect:/admin/emp/list");  }  **return** modelAndView;  }  } |

EmployeeService.java

|  |
| --- |
| **package** com.tecsup.gestion.services;  **import** java.util.List;  **import** com.tecsup.gestion.exception.DAOException;  **import** com.tecsup.gestion.exception.EmptyResultException;  **import** com.tecsup.gestion.model.Employee;  **public** **interface** EmployeeService {    ...............  ...............  ...............  **void** update(String login, String password, String lastname, String firstname, **int** salary, **int** dptId) **throws** DAOException;  } |

EmployeeServiceImpl.java

|  |
| --- |
| **package** com.tecsup.gestion.services;  **import** java.util.List;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  **import** com.tecsup.gestion.dao.EmployeeDAO;  **import** com.tecsup.gestion.exception.DAOException;  **import** com.tecsup.gestion.exception.EmptyResultException;  **import** com.tecsup.gestion.model.Employee;  @Service  **public** **class** EmployeeServiceImpl **implements** EmployeeService {  @Autowired  **private** EmployeeDAO employeeDAO;  ...............  ...............  ...............  @Override  **public** **void** update(String login, String password, String lastname, String firstname, **int** salary, **int** dptId)  **throws** DAOException {    employeeDAO.update(login, password, lastname, firstname, salary, dptId);  }      } |

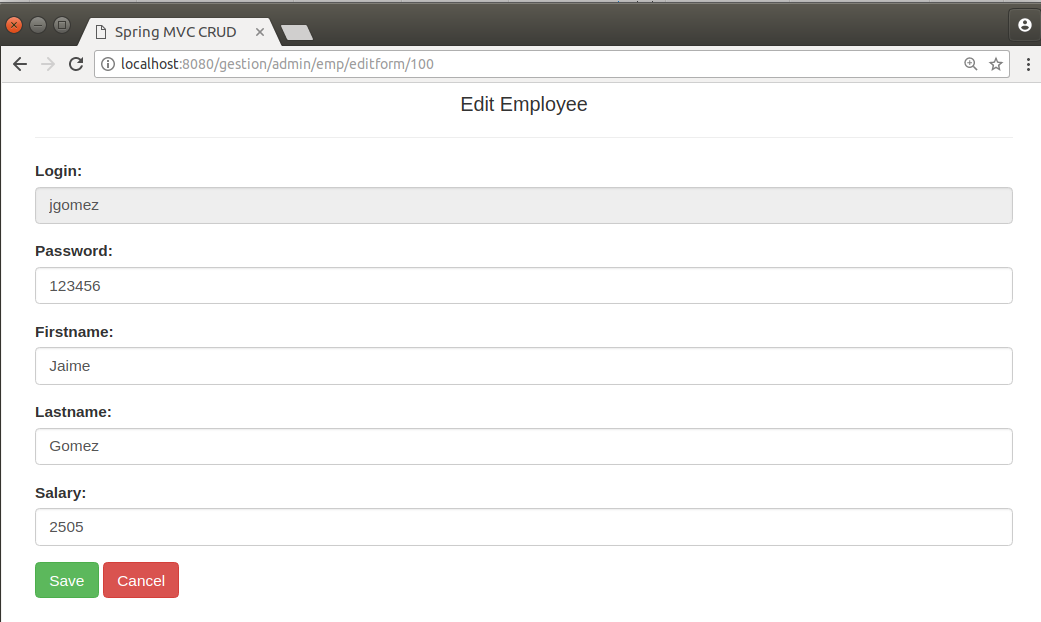
EmployeeDAOImpl.java

|  |
| --- |
| **package** com.tecsup.gestion.dao.jdbc;  **import** java.util.List;  **import** org.slf4j.Logger;  **import** org.slf4j.LoggerFactory;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.dao.EmptyResultDataAccessException;  **import** org.springframework.jdbc.core.JdbcTemplate;  **import** org.springframework.stereotype.Repository;  **import** com.tecsup.gestion.dao.EmployeeDAO;  **import** com.tecsup.gestion.exception.DAOException;  **import** com.tecsup.gestion.exception.EmptyResultException;  **import** com.tecsup.gestion.exception.LoginException;  **import** com.tecsup.gestion.mapper.EmployeeMapper;  **import** com.tecsup.gestion.model.Employee;  @Repository  **public** **class** EmployeeDAOImpl **implements** EmployeeDAO {  **private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(EmployeeDAOImpl.**class**);  @Autowired  **private** JdbcTemplate jdbcTemplate;  ...............  ...............  ...............  @Override  **public** **void** update(String login, String password, String lastname, String firstname, **int** salary, **int** dptId) **throws** DAOException {  String query = "UPDATE employees SET password = ?, first\_name =?, last\_name = ?, salary = ?, department\_id = ? WHERE login = ?";  Object[] params = **new** Object[] { password, lastname, firstname, salary, dptId, login };  **try** {  jdbcTemplate.update(query, params);  } **catch** (Exception e) {  ***logger***.info("Error: " + e.getMessage());  **throw** **new** DAOException(e.getMessage());  }  }  ...............  ...............  ...............  } |

/admin/emp/editform.jsp

|  |
| --- |
| <%@taglib uri=*"http://www.springframework.org/tags/form"* prefix=*"form"*%>  <%@taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>  <html>  <head>  <title>Spring MVC CRUD</title>  <meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>  <meta name=*"viewport"* content=*"width=device-width, initial-scale=1"*>  <link rel=*"stylesheet"*  href=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"*>  <script  src=*"https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"*></script>  <script  src=*"https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"*></script>  </head>  <body>  <div class=*"container-fluid"*>  <div class=*"row-fluid"*>  <div class=*"col-md-6"*>  <h4 class=*"text-center"*>Edit Employee</h4>  <hr>  <form:form method=*"post"* action=*"../editsave"*>  <form:hidden path=*"employeeId"* />  <div class=*"form-group"*>  <label for=*"login"*>Login: </label>  <form:input path=*"login"* class=*"form-control"* readonly=*"true"* />  </div>  <div class=*"form-group"*>  <label for=*"password"*>Password: </label>  <form:input path=*"password"* class=*"form-control"* />  </div>  <div class=*"form-group"*>  <label for=*"firstname"*>Firstname: </label>  <form:input path=*"firstname"* class=*"form-control"* />  </div>  <div class=*"form-group"*>  <label for=*"lastname"*>Lastname: </label>  <form:input path=*"lastname"* class=*"form-control"* />  </div>  <div class=*"form-group"*>  <label for=*"salary"*>Salary: </label>  <form:input path=*"salary"* class=*"form-control"* />  </div>  <%--  <div class="form-group">  <label for="department.departmentId">Department: </label>  <form:select path="department.departmentId" items="${departmentList}" />  </div>  --%>  <div class=*"form-group"*>  <input type=*"submit"* value=*"Save"* class=*"btn btn-success"* />  <a href=*"*<%=request.getContextPath()%>*/admin/emp/list"* class=*"btn btn-danger"*>Cancel</a>  </div>  </form:form>  </div>  </div>  </div>  </body>  </html> |

1. Ejecutar la funcionalidad de mostrar el detalle de un empleado a modificar



1. Preparar pruebas de integración de visualizar la edición de un registro de usuario
   1. Agregar el siguiente método

|  |
| --- |
| @Test  **public** **void** editForm() **throws** Exception {    mockMvc.perform(*get*("/admin/emp/editform/100"))  .andExpect(*status*().isOk())  .andExpect(*view*().name("admin/emp/editform"))  .andExpect(*forwardedUrl*("/WEB-INF/views/admin/emp/editform.jsp"))  .andExpect(*model*().attribute("command", *hasProperty*("employeeId", *is*(100))))  .andExpect(*model*().attribute("command", *hasProperty*("login", *is*("jgomez"))))  .andExpect(*model*().attribute("command", *hasProperty*("password", *is*("123456"))))  .andExpect(*model*().attribute("command", *hasProperty*("firstname", *is*("Jaime"))))  .andExpect(*model*().attribute("command", *hasProperty*("lastname", *is*("Gomez"))))  .andExpect(*model*().attribute("command", *hasProperty*("salary", *is*(2505))))  //.andExpect(model().attribute("command",  // hasProperty("department", hasProperty("departmentId",is(12)))))  ;  } |

* 1. Ejecutar la prueba con maven

|  |
| --- |
| mvn test -Dtest=EmployeeControllerIntegrationTest |

1. Ejecutar la funcionalidad de :
   1. El detalle de un empleado a eliminar
   2. La modificación de los datos de un empleado
   3. La eliminación de un empleado

**CONCLUSIONES**

Indicar las conclusiones que ha llegado después de desarrollar el laboratorio.

1.

2.

3.

***Criterios de Evaluación***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| logotipo | | ***Construcción y Pruebas de Software*** | | | | | | |
| **Rúbrica** | | | | | | |
| **Resultado** |  | | | | | | | |
| **Criterio de desempeño** |  | | | | | | | |
| **Curso** | ***Construcción y Pruebas de Software*** | | | | | **Periodo** | |  |
| **Actividad** | **Pruebas de Integración** | | | | | **Semestre** | | IV |
| **Nombre del Alumno** |  | | | | | **Semana** | | 12 |
| **Docente** | **Jaime Gómez** | **Fecha** |  | | | **Sección** | |  |
| **Criterios a Evaluar** | | **Excelente** | **Bueno** | **Requiere Mejora** | | **No Aceptable** | | **Puntaje Logrado** |
| * Identificar los componentes de a probar | | 7 | 5 | 3 | | | 2-0 |  |
| * Definir las pruebas de integración a realizar | | 6 | 4 | 3 | | | 1-0 |  |
| * Construir, ejecutar y documentar las pruebas de integración | | 7 | 6 | 4 | | | 2-0 |  |
| **Total** | | **20** | **15** | | **10** | **05** | |  |

|  |  |
| --- | --- |
| ***Acciones a cumplir*** | ***Menos*** |
| * Puntualidad y dedicación | 1 |
| * Cumplimiento de tiempos establecidos | 1 |
| * Conclusiones: ortografía y redacción. | 1 |
| ***Puntaje Total*** |  |

|  |  |
| --- | --- |
| Comentarios respecto del desempeño del alumno |  |

|  |  |
| --- | --- |
|  | ***Descripción*** |
| Excelente | Demuestra un completo entendimiento del problema o realiza la actividad cumpliendo todos los requerimientos especificados. |
| Bueno | Demuestra un considerable entendimiento del problema o realiza la actividad cumpliendo con la mayoría de los requerimientos especificados. |
| Requiere Mejora | Demuestra un bajo entendimiento del problema o realiza la actividad con pocos de los requerimientos especificados. |
| No aceptable | No demuestra entendimiento del problema o actividad. |