Servicio de Instrucción	Oficialización: Director Edwin Maraví
	Año:2020
CURSO **ARCHITECT**	Código : C05-2020

Sílabo
JAVA ARCHITEC

DATOS GENERALES

Año de vigencia	:	
Número de sesiones	:	20 (4 horas por sesión)
N° de Horas académicas	:	80 horas académicas
Requisitos	:	Desarrollo de aplicaciones (Front End y Back End)

I. FUNDAMENTACIÓN

El curso orienta a los participantes en el correcto uso de las librerías, frameworks Java y servidores de aplicaciones para poder estableces mecanismos de comunicaciones entre aplicaciones heterogéneas. El curso orienta al participante en uso de herramientas y conocimientos de tecnologías necesarias para poder establecer procedimientos de Integración de Aplicaciones.

II. UNIDAD DE COMPETENCIA

Integrar aplicaciones heterogéneas basadas en arquitectura orientada a servicios.

Sesión	Contenido
Sesión 01	**Introducción a la Arquitectura Empresarial** Definición de Arquitectura de Software. Necesidades de la Arquitectura de Software. Sistemas Altamente Distribuidos. Calidad de servicio. Arquitectura Empresarial. Diferencia entre Arquitectura y Diseño. Principios de Arquitectura.
Sesión 02	**Patrones Arquitecturales Empresariales** **Patrones de Frontera** API Gateway. Command Query Responsibility Segregation (CQRS). Offline-first database. Backend For Frontend. External Service Gateway. **Patrones de Control** Event collaboration. Saga. **Patrones de Despliegue** Decoupling deployment from reléase. Multi-level roadmaps. Task branch workflow. Modern deployment pipelines. Zero-downtime deployment.

Sesión 03	**Desarrollando una Arquitectura de Seguridad:** **Tipos de control de acceso: Basado** en Roles. Control de Acceso Mandatorio. SQL Injection. Cross Site scripting. Ingeniería social. Pretexting. **Restricciones regulatorias: HIPPA**. **Impactos de seguridad en computación distribuida** **Spring security / JSON Web Tokens**
Sesión 04	**Entendimiento de los requerimientos No Funcionales** Definición y ejemplos. Categorías de requerimientos no funcionales. Impacto dimensiones sobre requerimientos no funcionales**.** Introduciendo redundancia a la arquitectura de sistemas. Captura y examen de requerimientos no funcionales Balanceo de cargas. Conceptos de Failover. Clusters. Topología de Cluster en par Evaluación de estrategias de replicación. Diseño de redes. Consideraciones de costos Priorización de la Calidad de Servicio (QoS).
Sesión 05	
Sesión 06	**Definiendo problema comunes y soluciones : Flexibilidad y Factores de Riesgo** Identificación de Factores de riesgo en la Arquitectura. Flexibilidad en los sistemas Modelos transaccionales. Planeamiento y dimensionamiento de la arquitectura Aplicación de principios orientados a objetos. Principio de Open-Closed Principio de Inyección de Dependencias. Principio de Segregación de interfaces Principio de Composite Reuse. Aplicación de Patrones. Patrones de diseño Gan of Four (Gof). Patrones JavaEE. Patrones arquitectónicos de Buschman. Patrones de Capa Patrones Integración. Usando patrones a través de la red. Arquitectura basada en servicios. Patrones basados en objetos
Sesión 07	**Definiendo problema comunes y soluciones : Redes, Transacciones y Capacidad de Planeamiento** Guía para la Comunicación en red. Guía para la performance en red. Falacias de la computación distribuida. Creando un modelo de red. Estimación la latencia de red Construyendo modelos de datos eficientes. Incrementando la granularidad de operaciones. Justificación del uso de transacciones. Tipos de transacciones Impacto de las transacciones sobre la latencia de red. Conflictos de escritura Creación de modelos transaccionales. Planeamiento de la capacidad del sistema Estimación de la carga de transacciones. Estimando el ratio de transacciones Dimensionando el sistema. Planificando la escalabilidad. Cloud Computing Infraestructura de alta performance en AWS
Sesión 08	**Desarrollo de Arquitectura de la Capa Cliente y Front End** **Selección de la Tecnología de Interface de Usuario y Dispositivos** Limitación de recursos. Robustez de los dispositivos de ingreso de datos. Interacciones de Usuario Básicas y Complejas. **Arquitectura de Front para el diseño de Sistemas** HTML5 y CSS. SPA. JavaScript. Red Hat Code. Task Runners. Unit Testing. Performance Testing. Redux. Flux. **Desarrollo de Arquitectura de la Capa Cliente y Front End** **Tecnologías de Web Browser** HTML5. JavaScript. CSS. Compatibilidad en Browsers.
Sesión 09	**Desarrollo de Arquitectura de la Capa Cliente y Front End** **Tecnologías para el desarrollo de Móviles** Android. React Native. Ionic 4. Laboratorio con Android y React Native **Tecnologías de Interface de Usuario para desarrollo de Front End** Angular. ReactJs **Desarrollo de Arquitectura de la Capa Web**

	Gestión de sesiones de usuario y entidades con estado. Servlets, JSP, Java Server Faces. JNDI. Roles en el desarrollo de la capa web. Separación de Intereses. SoC. Lenguajes de Scripting. Internacionalización y Localización. Patrón MVC. Patrón MVVC. Frameworks de la Capa Web, Librerías. Spring Web Flow. Django. Ruby on Rails. PHP. GWT. Angular. Vue. React
Sesión 10	
Sesión 11	**<u>Desarrollo de una Arquitectura de la Capa de Negocio y Back End</u>** **Enterprise Java Beans.** **Web Services** Arquitectura de Web Services. Interoperabilidad de Web Services. Estándares para implementación de SOAP Web Services: SOAP, WSDL, UDDI y WS-Security. Estándares para implementación de RESTful Web Services: HTTP, WADL, URL/URI y Internet Media Types. SOAP Web Services vs REST Web Services. Especificación JAX-WS – Metro Reference Implementation. SOAP 1.1/1.2.
Sesión 12	**<u>Microservicios</u>** **Introducción a Microservicios**. Analizando Sistemas con arquitecturas monolíticas. Moviéndonos a una arquitectura de microservicios. Construyendo microservicios con Spring Boot. Descubrimiento de microservicios. Service Routing con Spring Cloud y Zuul. Pros y cons de una arquitectura dirigida por eventos. Gestión de eventos con RabbitMQ. Request de datos entre microservicios. Despliegue de microservicios a Docker. **Arquitectura orientada a mensajes** Message Oriented Middleware (MOM). Modelos de mensajería: Point to Point y Publish/Subscribe. Procesamiento síncrono vs asíncrono. Java Message Service (JMS) Especificación JSR 914 – Java Message Service API. Principales implementaciones del mercado. **API Drools.** Implementación de reglas con Drools Expert: reglas básicas, validaciones y transformaciones. Implementación de flujos con JBPM5. Editor BPMN. Events, actions, tasks y gateways.
Sesión 13	**<u>Desarrollo de una Arquitectura para integración y Capa de Recursos</u>** **Arquitectura REST**. JSR 311 – Java API for RESTful Web Services. JAX-RS Jersey Reference Implementation. JSR 339 – JAX-RS 2.0. **Arquitectura orientada a mensajes** Message Oriented Middleware (MOM). Modelos de mensajería: Point to Point y Publish/Subscribe. Procesamiento síncrono vs asíncrono. Java Message Service (JMS) Especificación JSR 914 – Java Message Service API. Principales implementaciones del mercado. **API Drools.** Implementación de reglas con Drools Expert: reglas básicas, validaciones y transformaciones. Implementación de flujos con JBPM5. Editor BPMN. Events, actions, tasks y gateways
Sesión 14	**Métricas de Rendimiento de Software** Normativa de medición del rendimiento de software. Herramientas de medición: JMeter o Selenium IDE. Selenium Driver. SOAP UI. Bases para implementación de una herramienta manual. Dimensionamiento y pruebas de rendimiento de software.
Sesión 15	
Sesión 16	**<u>Contenerizado de Aplicaciones</u>** **Dockers:** Instalación de Dockers. Definiendo Scripts Docker. Extensiones Docker en Visual Studio Code. Desplegando una app contenerizada. **Kubernetes**. Introducción a Kubernetes**.**

Sesión 17	**Modelado y diseño para desarrollo de Microservicios** Diseño guiado por dominio (DDD). Diseño guiado por comportamiento (BDD).
Sesión 18	**Diagramas UML para Arquitectura** Definición. Funciones y procedimientos. Paso de parámetros. Ejemplos y ejercicios con funciones y procedimientos.
Sesión 19	**Despliegue en Infraestructura de Cloud** Infraestructura de Alta Disponibilidad. Creación de cuentas seguras de AWS. Redimensionamiento correcto de Infraestructura en AWS. Despliegue a AWS ECS Fargate Creando balanceadores de carga.
Sesión 20	

III. METODOLOGÍA

- Curso teórico-práctico.
- Exposición del profesor, ayudas audiovisuales, diálogo abierto y debate.
- Realización de casos prácticos en laboratorio.

IV. BIBLIOGRAFÍA

- DIETEL & DIETEL - La Biblia de Java.
- OFALI, ROBERT y HARKEY, DAN - Cliente Servidor. McGraw-Hill, 2da Edición, México, 2001
- GALLAGHER, SIMON y HERBERT, SIMON - Cliente Servidor e Internet Addison Wesley, 2da Edición. México, 2002

V. PÁGINAS WEB DE APOYO:

- http://www.programacion.com/java
- http://www.javahispano.com
- http://docs.oracle.com/javaee/5/tutorial/doc/bnbpy.html
- http://docs.oracle.com/javaee/6/tutorial/doc/bnadr.html
- http://www.lawebdelprogramador.com
- http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
- http://www.elvex.ugr.es/decsai/java/pdf/