

1. **INFORMACIÓN GENERAL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Apellidos y Nombres: |  | ID: | 13 | |
| Dirección Zonal/CFP: | Huánuco | | | |
| Carrera: | Ingeniería de software con inteligencia artificial | Semestre: | | III |
| Curso/ Mód. Formativo | Lenguaje de programación II | | | |
| Tema del Trabajo: | Creación de E commerce con servlets y SpringBoot | | | |

1. **PLANIFICACIÓN DEL TRABAJO**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ACTIVIDAD** | **INICIO DEL PLAN** | **DURACIÓN DEL PLAN** | **INICIO REAL** | **DURACIÓN REAL** | **PORCENTAJE COMPLETADO** | **PERIODOS** | | |  |  |  |  |  |  |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
| **Diseño front-end** | 1 | 3 | 1 | 3 | **100%** |  |  |  |  |  |  |  |  |  |
| **Diseño back-end** | 1 | 2 | 1 | 2 | **70%** |  |  |  |  |  |  |  |  |  |
| **Controllers(Piero)** | 2 | 2 | 2 | 2 | **100%** |  |  |  |  |  |  |  |  |  |
| **Model(Antony)** | 2 | 3 | 2 | 3 | **50%** |  |  |  |  |  |  |  |  |  |
| **Services(Antony)** | 4 | 3 | 4 | 3 | **85%** |  |  |  |  |  |  |  |  |  |
| **Repository(Piero)** | 4 | 3 | 4 | 3 | **35%** |  |  |  |  |  |  |  |  |  |
| **Front-end (thymeleaf) prototipo (Piero)** | 7 | 2 | 7 | 2 | **50%** |  |  |  |  |  |  |  |  |  |
| **CSS (Antony) (Materialize)** | 7 | 2 | 7 | 2 | **60%** |  |  |  |  |  |  |  |  |  |
| **Pool de conexiones** | 4 | 1 | 4 | 1 | **100%** |  |  |  |  |  |  |  |  |  |

1. **PREGUNTAS GUIA**

**Durante la investigación de estudio, debes obtener las respuestas a las siguientes interrogantes:**

|  |  |
| --- | --- |
| **Nº** | **PREGUNTAS** |
| **1** | **¿Cuáles son las ventajas de programar interfaces web con JAVA EE?** |
| **2** | **¿Por qué se deben diseñar capas de la aplicación?** |
| **3** | **¿En qué consiste la creación de un Servlet?** |
| **4** | **¿Por qué implementar JDBC / Servlet API para la conexión de base de datos?** |
| **5** | **¿Qué ventaja tiene implementar POOL de conexiones y transacciones JDBC?** |

|  |  |
| --- | --- |
| 1. |  |
| Primero debemos entender que arquitectura de software más empleada par desarrollo de aplicaciones web es el patrón MVC (Modelo, Vista, Controlador), Es el más conocido y empleado porque separa tu código en capas, para evitar generar código espagueti, de esta manera puedes ‘mantener’ tu proyecto entendible y las tener una base sólida para que hacer crecer tu proyecto. Las capas permiten separar la vista, el modelo, y el controlador para que en la etapa de desarrollo podamos realizar pruebas si ocurre algún error.  Asimismo, diseñamos capas para que la aplicación web sea más segura, y ninguna persona externa pueda filtrar tus datos. | |
| 2. |  |
| Un servlet es un script java con la implementación necesaria para establecer una conexión con el servidor, Tiene dos métodos principales, doGet y doPost para obtener las peticiones del usuario y darle una respuesta inmediata (request, response). Adicional a ello podemos mapear la estructura lógica del programa, conectar a un modelo, y brindarle respuestas a una vista. En pocas palabras un servlet es un controlador. | |
| 3. |  |
| Un servlet es un script java con la implementación necesaria para establecer una conexión con el servidor, Tiene dos métodos principales, doGet y doPost para obtener las peticiones del usuario y darle una respuesta inmediata (request, response). Adicional a ello podemos mapear la estructura lógica del programa, conectar a un modelo, y brindarle respuestas a una vista. En pocas palabras un servlet es un controlador. | |
| 4. |  |
| JDBC permite la integración de llamadas SQL dentro del código de nuestro Servlet Java proporcionando clases que nos permiten interactuar de forma fácil, cómoda y homogénea con bases de datos externas. | |
| 5. |  |
| Un pool de conexiones mantiene y gestiona un número de conexiones físicas, que se irán reutilizando automáticamente para aumentar la eficiencia.  A su vez el pool es capaz de ofrecer múltiples conexiones lógicas utilizando un reducido número de conexiones reales. El manejo de un pool favorece la escalabilidad y el rendimiento de una aplicación. | |

**PROCESO DE EJECUCIÓN**

|  |  |
| --- | --- |
| **OPERACIONES / PASOS /SUBPASOS** | **SEGURIDAD / MEDIO AMBIENTE / NORMAS -ESTANDARES** |
| Creación de branchs en el sistema de gestión de versiones | ISO 9001 |
| Puesta a punto del entorno |  |
| Creación de diagramas de flujo y diagrama modelo entidad-relación | ISO 9001 |
| Creación de la base de datos y las tablas | Utf-8 sin bom |
| **Proceso de desarrollo para el front-end:** | |
| **productos.html**: en esta interfaz se mostrar todos los productos disponibles de la empresa, ademas de ello el usuario podra filtrarlos por categorias, | Hypertext Markup Language |
| **detalleProducto.html**: Es la vista detallada del producto, se actualizara la información dependiendo del producto que el usuario haya solicitado ver. | Hypertext Markup Language |
| **carrito.html:** Una ves el consumidor disponga realizar la compra el producto se añadirá a lista de espera que se mostrar en esta página. | Hypertext Markup Language |
| Para facilitarnos el trabajo emplearemos el framework CSS Materialize, no será necesario descargar la librería, tan solo con importar su CDN el sitio web tendrá un estilo amigable. | <https://materializecss.com/getting-started.html> |
| Conectamos la vista y el controlador con la dependencia Thymeleaf. que nos permitirá convertir html en un lenguaje de programación. | <https://www.thymeleaf.org/> |
| **Proceso de desarrollo para el back end** | |
| **Controller.java:** El controlador con anotaciones GetMapping y PostMapping que redirigen acorde a la url solicitada | SpringBoot |
| **Product.java:** El modelo en forma de Entity que creará la tabla con las columnas necesarias y también del cual se creará un servicio para añadir como atributo a la sesión | SpringBoot |
| **ProductRepository.java:** Interfaz heredada de CRUDRepository con los métodos necesarios para interactuar con JPA/Hibernate para la persistencia de datos | SpringBoot |
| **ProductService.java:** Servicio de springboot que implementa los métodos de la interfaz ProductRepository | SpringBoot |
| **CarritoCompras.java**: Clase que abstrae un carrito de compras como un iterable para poder solicitarlo en la sesión como un atributo para no depender de la persistencia | SpringBoot |

|  |  |  |
| --- | --- | --- |
| **USE CASE DIAGRAM** | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | USE CASE DIAGRAM | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

|  |  |  |
| --- | --- | --- |
| **DIAGRAMA DE CLASE (RAW UML 1)** | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

|  |  |  |
| --- | --- | --- |
|  | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

|  |  |  |
| --- | --- | --- |
| Mostrar todos los productos en el catalogo  Filtrar por categoría (protección facial) | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

|  |  |  |
| --- | --- | --- |
| Mostrar en detalle el producto Orejeras NoiceCancel  Mostrar en detalle el producto Mascara de Soldar | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

|  |  |  |
| --- | --- | --- |
| Listado de los productos agregados al carrito de compras | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |
| Se demuestra que la pagina web esta adaptado para dispositivos mobiles | | |
| C:\Unidad_D\Nuevo Logo\SENATI_FF-01 Modificado.jpg | [NOMBRE DEL TRABAJO] | |
| [APELLIDOS Y NOMBRES] | [ESCALA] |

**INSTRUCCIONES: completa la lista de recursos necesarios para la ejecución del trabajo.**

|  |  |  |
| --- | --- | --- |
| **1. MÁQUINAS Y EQUIPOS** | | |
|  | PC de escritorio: Procesador: Ryzen 7, GPU: NVIDIA gtx1050, Disco Duro: WD SSD. RAM: ddr3 8gb |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **3. HERRAMIENTAS E INSTRUMENTOS** | | |
|  | Sistema de control de versiones: GIT |  |
|  | Entorno: JetBrains |  |
|  | IDE: Inteliji IDEA |  |
|  | Lenguaje: JAVA |  |
|  | BBDD: MYSQL |  |
|  | GUI BDD: CleverCluud |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **5. MATERIALES E INSUMOS** | | |
|  | Energía eléctrica |  |
|  | GitHub como repositorio cloud |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |