|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| escudojpg | **POLITÉCNICO COLOMBIANO JAIME ISAZA CADAVID**  **FORMATO PARA REALIZACIÓN DE ACTIVIDADES EVALUATIVAS**  **FACULTAD DE INGENIERÍAS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **PROGRAMA: Tecnología en Sistematización de Datos Propedéuticos Con la Técnica Profesional \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | | | | | **Código:**  **FD-GC195** |
| **Versión: 02** |
| **ASIGNATURA** | **CÓDIGO:ING01411** |  | **NOMBRE: Construcción de Elementos de Software 3** | | | |
| **PROFESOR:** | **OSCAR MESA** | | | **FECHA: 08-04-2025** |  | |
| **TIPO DE EVALUACIÓN** | **TALLER** | **QUIZ** | **PARCIAL** | **FINAL** | **OTRO – CUÁL?** | |
|  | **X** |  |  |  | |

### Requisitos Técnicos

* Java para desarrollo de Servlets
* Apache Tomcat como servidor de aplicaciones
* Una clase DAO para almacenar información en memoria usando ArrayList
* Uso de GSON para manejo de respuestas JSON
* Control de versiones:
  + Inicializar un repositorio Git local
  + Realizar al menos 3 commits durante el desarrollo:
    - Commit inicial con la estructura del proyecto
    - Commit con la implementación de la clase DAO y modelo
    - Commit con la implementación del Servlet
* Requisitos de estructura:
  + Crear un proyecto Java Web con la estructura adecuada (WEB-INF, clases, etc.)
  + Incluir las bibliotecas necesarias (GSON, servlet)

### Entregables

* Código fuente completo del proyecto subido a un repositorio público en GitHub
* El repositorio debe tener el nombre "**Platon Api**".
* Un archivo README.md en el repositorio explicando:
  + Qué variante implementaste según tu número de documento
  + Cómo se puede probar el servicio (URLs y formatos)

### Criterios de Evaluación

* Implementación correcta del Servlet y los endpoints REST (45%)
* Uso adecuado de colecciones Java para almacenar datos (25%)
* Implementación de la lógica de negocio y validaciones (20%)
* Uso correcto de Git con commits significativos (10%)

### Instrucciones para el Estudiante

* Verifica el último dígito de tu documento de identidad:
  + Si termina en número PAR (0,2,4,6,8): Implementa el punto 1 o Variante A (Gestión de Cursos)
  + Si termina en número IMPAR (1,3,5,7,9): Implementa la Variante B (Gestión de Inscripciones)
* Crea un proyecto Java Web en tu IDE favorito
* Implementa las clases necesarias según la variante que te corresponda
* Utiliza Git para el control de versiones
* Al finalizar, sube el proyecto a un repositorio público en GitHub con el nombre "**Platon Api**" y se envía por el slack al docente de la materia.

### Este ejercicio evalúa:

* Capacidad para implementar servicios REST básicos con Java
* Uso de colecciones en Java
* Aplicación de lógica de negocio simple
* Manejo básico de Git
* Conocimientos de Java Web con Tomcat

## 1. (100%) Variante A: Gestión de Cursos (Documentos que terminan en NÚMERO PAR)

1. **Crear la clase modelo Curso**:  
   * Atributos: id (int), nombre (String), código (String), profesor (String), cupoMaximo (int), estudiantesInscritos (int), facultad (String), prerequisitos (ArrayList<String>), nivel (int), fechaInicio (String)
2. **Implementar CursoService**:  
   * Método para agregar un curso
   * Método para buscar cursos por facultad
   * Método para encontrar rutas de aprendizaje (cursos que deben tomarse en secuencia según prerequisitos)
3. **Crear un CursoServlet que implemente los siguientes endpoints**:  
   * POST /cursos: Permite registrar un nuevo curso (recibiendo la información por JSON o parámetros)
   * GET /cursos/facultad?nombre=X: Retorna los cursos de una facultad específica
   * GET /cursos/ruta?codigo=X: Retorna todos los cursos prerequisitos necesarios para tomar el curso con código X, ordenados por nivel académico
4. **Implementar validaciones de negocio**:  
   * No permitir crear cursos con el mismo código
   * Verificar que el cupo máximo sea mayor que cero
   * Validar que los prerequisitos especificados existan en el sistema

## 2. (100%) Variante B: Gestión de Inscripciones (Documentos que terminan en NÚMERO IMPAR)

1. **Crear la clase modelo Inscripcion**:  
   * Atributos: id (int), estudiante (String), documento (String), carrera (String), asignatura (String), semestre (String), fechaInscripcion (String), estado (String - "Activa", "Cancelada"), creditos (int), prioridad (int), promedioAcumulado (double)
2. **Implementar InscripcionService**:  
   * Método para registrar una inscripción
   * Método para buscar inscripciones por carrera
   * Método para generar horarios optimizados evitando colisiones
3. **Crear un InscripcionServlet que implemente los siguientes endpoints**:  
   * POST /inscripciones: Permite registrar una nueva inscripción
   * GET /inscripciones/carrera?nombre=X: Retorna las inscripciones de una carrera específica
   * GET /inscripciones/priorizadas: Retorna las inscripciones ordenadas por un algoritmo de prioridad (basado en promedio académico, créditos aprobados y fecha de inscripción)
4. **Implementar validaciones de negocio**:  
   * No permitir más de 7 inscripciones para un mismo estudiante
   * No permitir inscripciones con el mismo estudiante y asignatura
   * Implementar un algoritmo que calcule la prioridad de inscripción basado en: promedioAcumulado \* 0.6 + (creditos/10) \* 0.3 + (días desde la inscripción) \* 0.1
   * Implementar este método donde considere pertinente:

