CURSO: LENGUAJE DE PROGRAMACION
DOCENTE: PEDRO HERNAN DE LA CRUZ VELAZCO

Estimados estudiantes,

Para el proyecto final del curso de **LENGUAJE DE PROGRAMACION**, deberán desarrollar un sistema utilizando **Python**. **Se valorará significativamente que el proyecto sea una APLICACIÓN WEB**, ya que esto les permitirá adquirir habilidades cruciales y de alta demanda en el mercado actual.

Su informe y presentación final deberán adherirse estrictamente a la siguiente estructura y serán evaluados según la rúbrica adjunta.

Estructura del Proyecto de Sistemas (Java/Python)

Esta estructura busca guiarles en la organización y presentación de sus proyectos de sistemas, asegurando la inclusión de los componentes clave desde la concepción hasta la implementación y las pruebas.

Carátula

- **Nombre de la Universidad:** [Nombre completo de la Universidad]
- **Facultad/Departamento:** [Nombre de la Facultad o Departamento]
- **Título del Proyecto:** [Título completo del proyecto]
- **Asignatura:** LENGUAJE DE PROGRAMACION
- **Docente:** PEDRO HERNAN DE LA CRUZ VELAZCO
- **Integrantes del Grupo:**
  - [Nombre completo del alumno 1] - [Código/ID]
  - [Nombre completo del alumno 2] - [Código/ID]
  - [Nombre completo del alumno 3] - [Código/ID]
  - *(Añadir más según sea necesario)*
- **Fecha de Entrega:** [Día] de [Mes] de [Año]

1. Introducción

- **1.1. Resumen Ejecutivo:** Breve descripción del proyecto, sus objetivos principales y los resultados esperados.
- **1.2. Descripción del Problema:** Detalle del problema o necesidad que el proyecto busca resolver. ¿Por qué es importante este proyecto?
- **1.3. Objetivos del Proyecto:**
  - **1.3.1. Objetivo General:** El propósito global del proyecto.
  - **1.3.2. Objetivos Específicos:** Metas claras y medibles que contribuyen al objetivo general.

- **1.4. Alcance del Proyecto:** Qué incluye el proyecto y, crucialmente, qué **no** incluye. Delimitación de funcionalidades y características.

2. Marco Teórico / Tecnológico

- **2.1. Fundamentos Teóricos:** Breve revisión de conceptos teóricos relevantes para el proyecto (por ejemplo, principios de bases de datos, algoritmos, paradigmas de programación, etc.).
- **2.2. Tecnologías y Herramientas Utilizadas:**
  - **2.2.1. Lenguaje de Programación:** Python (especificar versión y frameworks como Flask/Django).
  - **2.2.2. Frameworks/Librerías:** (Ej: Flask, Django, requests, NLTK, etc.)
  - **2.2.3. Base de Datos:** (Ej: SQLite, PostgreSQL, MongoDB, etc.)
  - **2.2.4. Entorno de Desarrollo (IDE):** (Ej: VS Code, PyCharm)
  - **2.2.5. Control de Versiones:** (Ej: Git/GitHub)
  - **2.2.6. Otras Herramientas:** (Ej: Herramientas de diseño, herramientas de testing, Docker, etc.)
- **2.3. Arquitectura del Sistema:** Descripción de la arquitectura propuesta (ej: Cliente-Servidor, MVC, Microservicios, Capas). Incluir diagramas si es necesario.

3. Diseño del Sistema

- **3.1. Requisitos del Sistema:**
  - **3.1.1. Requisitos Funcionales:** Qué debe hacer el sistema. (Listado de casos de uso o historias de usuario).
  - **3.1.2. Requisitos No Funcionales:** Calidad del sistema (rendimiento, seguridad, usabilidad, mantenibilidad, etc.).
- **3.2. Diseño de la Base de Datos:**
  - **3.2.1. Modelo Entidad-Relación (ERD):** Diagrama que muestra las entidades y sus relaciones.
  - **3.2.2. Esquema de la Base de Datos:** Tablas, campos, tipos de datos, claves primarias y foráneas.
- **3.3. Diseño de la Interfaz de Usuario (UI/UX):**
  - **3.3.1. Bocetos/Wireframes:** Representaciones visuales de las pantallas principales.
  - **3.3.2. Flujos de Usuario:** Cómo el usuario interactúa con el sistema.
- **3.4. Diagramas de Diseño:** (Seleccionar los pertinentes, por ejemplo: Diagrama de Clases, Diagrama de Secuencia, Diagrama de Componentes).

4. Implementación

- **4.1. Estructura de Carpetas/Módulos:** Organización del código fuente.
- **4.2. Descripción de Componentes Clave:** Explicación de las partes más importantes del código y su funcionalidad.
- **4.3. Consideraciones de Programación:** Patrones de diseño aplicados, buenas prácticas de codificación, comentarios en el código.

- **4.4. Integración de Módulos:** Cómo se comunican las diferentes partes del sistema.

5. Pruebas y Resultados

- **5.1. Plan de Pruebas:**
  - **5.1.1. Tipos de Pruebas Realizadas:** (Ej: Pruebas unitarias, de integración, de sistema, de aceptación, de rendimiento).
  - **5.1.2. Casos de Prueba:** Escenarios de prueba con entradas esperadas y salidas esperadas.
- **5.2. Resultados de las Pruebas:** Evidencia de las pruebas ejecutadas (capturas de pantalla, logs, etc.).
- **5.3. Detección y Solución de Errores:** Errores encontrados y cómo se corrigieron.

6. Conclusiones y Recomendaciones

- **6.1. Conclusiones:** Resumen de los logros del proyecto y si se cumplieron los objetivos.
- **6.2. Limitaciones del Proyecto:** Aspectos que no se pudieron abordar o mejorar por diversas razones.
- **6.3. Trabajos Futuros/Mejoras:** Sugerencias para la expansión o mejora del proyecto en el futuro.

7. Referencias Bibliográficas

- Listado de todas las fuentes consultadas (libros, artículos, sitios web, documentación de librerías, etc.) utilizando un formato consistente (ej: APA, ISO 690).

8. Anexos (Opcional)

- Código fuente completo (enlace a un repositorio Git).
- Manual de usuario.
- Guía de instalación.
- Capturas de pantalla adicionales.
- Cualquier otra información relevante.

Rúbrica de Evaluación del Proyecto de Sistemas

Esta rúbrica evaluará su proyecto en una escala vigesimal (0-20), distribuyendo el puntaje en cuatro dimensiones clave.

Rúbrica de Evaluación de Proyecto de Sistemas (Escala Vigesimal)

Dimensión de Evaluación	0 - 5 (Deficiente)	6 - 10 (Regular)	11 - 15 (Bueno)	16 - 20 (Sobresaliente)	Puntaje
1. Contenido y Estructura del Informe	El informe es incompleto, desorganizado y carece de secciones clave. La información es insuficiente y confusa.	El informe presenta la mayoría de las secciones, pero su organización es inconsistente y la información es limitada o poco clara en algunos puntos.	El informe está bien estructurado y contiene las secciones principales. La información es clara, relevante y coherente.	El informe es excepcionalmente completo, organizado y sigue la estructura propuesta. La información es precisa, detallada y presenta un alto nivel de profesionalismo.	
2. Diseño y Calidad de la Solución	El diseño es nulo o muy deficiente. La solución es inestable, con errores frecuentes y funcionalidades incompletas. (Sistemas de escritorio o web muy básicos).	El diseño es básico, con algunas fallas. La solución funciona parcialmente, pero presenta errores significativos y limitaciones en sus funcionalidades. (Sistemas de escritorio con errores o web muy simple).	El diseño es adecuado y la solución es funcional y estable para la mayoría de los casos. Presenta una lógica clara y resuelve el problema principal. **Se prioriza el desarrollo web por encima de las aplicaciones de escritorio.**	El diseño es robusto y bien pensado (arquitectura, base de datos, UI/UX). La solución es completamente funcional, estable, eficiente y cumple con todos los requisitos. **Se valora positivamente si es una aplicación web bien implementada.**	
3. Calidad del Código y Replicabilidad	El código es ilegible, sin comentarios, sin estándares y con muy malas prácticas. La replicabilidad es imposible.	El código es difícil de entender en algunas partes, con pocos comentarios y estándares inconsistentes. La replicabilidad	El código es comprensible, con comentarios adecuados y sigue buenas prácticas de programación. La replicabilidad es posible	El código es excelente: legible, bien documentado, modular, eficiente y sigue los mejores estándares. La replicabilidad es sencilla y bien documentada.	

		requiere mucho esfuerzo.	con instrucciones claras.		
4. Presentación y Demostración	La presentación es caótica, sin coherencia. La demostración no funciona o no logra mostrar el sistema. Respuestas deficientes.	La presentación es básica, con poca fluidez. La demostración tiene dificultades o no aborda todos los puntos. Respuestas con dudas.	La presentación es clara y organizada. La demostración es funcional y muestra las características principales. Las respuestas son en su mayoría correctas.	La presentación es sobresaliente: clara, concisa, profesional y convincente. La demostración es impecable y el sistema funciona a la perfección. Las respuestas son expertas y bien fundamentadas.	
PUNTAJE FINAL (Suma de los puntajes de cada dimensión)					**/20**

Sugerencias de Proyectos Web Viables y Útiles (Python):

Les animamos a considerar proyectos que involucren el uso de frameworks web como Flask o Django. Algunas ideas que pueden explorar, que son muy pertinentes en la actualidad, incluyen:

- **Aplicación de Gestión de Tareas (To-Do List con Autenticación):** Ideal para comprender CRUD, autenticación de usuarios y persistencia de datos.
- **Blog Simple o Gestor de Contenido (CMS Básico):** Excelente para explorar modelos de datos, relaciones y paneles de administración.
- **Aplicación de Clima con API Externa:** Perfecta para aprender a consumir APIs RESTful y mostrar datos dinámicos.
- **Acortador de URLs:** Un proyecto compacto que enseña sobre redirecciones HTTP y generación de identificadores únicos.
- **Chatbot Simple con Web UI:** Para explorar procesamiento básico de texto e interacción web.
- **Convertidor de Unidades/Monedas Online:** Para práctica con formularios y consumo de APIs de datos.

¡Esperamos sus proyectos innovadores! Si tienen alguna pregunta, no duden en consultarme.