Práctica Calificada 3 C8288

Instrucciones de entrega para la prueba calificada

Para asegurar una entrega adecuada y organizada de su prueba calificada, sigan cuidadosamente las siguientes instrucciones. El incumplimiento de estas pautas resultará en que su prueba no sea revisada, sin importar el escenario.

- Presenta un repositorio con todas las actividades resueltas junto con todas las respuestas y soluciones documentadas, completas junto con el codigo del lenguaje trabajado dentro de las subcarpetas correspondientes.
- No se aceptarán entregas en formatos como documentos de Word (.docx) o archivos PDF. La entrega debe cumplir estrictamente con el formato Markdown (.md) y las extensiones del código del lenguaje desarrollado.

Parte 1 (8 puntos): Presenta las respuestas de las actividades 11 hasta la 16 de la plataforma. Las actividades 15 y 16 valen 2 puntos

Parte 2 (12 puntos)
Escoge uno de los proyecto y sigue las siguientes sugerencias:

1. Dedica las primeras horas a definir claramente las tareas y priorizar las funcionalidades esenciales.
2. Aprovecha al máximo las bibliotecas y herramientas que faciliten la implementación de funcionalidades complejas, como cifrado y autenticación.
3. Utiliza plantillas y componentes preconstruidos para acelerar el desarrollo del frontend y backend.
4. Configura scripts para la contenerización con Docker y para la ejecución de pruebas desde el inicio del proyecto.
5. Prioriza las funcionalidades clave que demuestren el objetivo principal del proyecto, dejando funcionalidades adicionales para futuras expansiones si el tiempo lo permite.

1. Aplicación de compartición de archivos segura con cifrado asimétrico

Descripción:
Desarrolla una aplicación web básica que permita a dos usuarios predefinidos subir y descargar archivos de forma segura. Implementa cifrado asimétrico en el cliente utilizando una biblioteca como JSEncrypt para cifrar los archivos antes de enviarlos al servidor. Utiliza Node.js y Express en el backend, y React con TypeScript en el frontend. Almacena los archivos cifrados en MongoDB utilizando Mongoose. Implementa autenticación básica en lugar de OAuth para controlar el acceso.

Realiza pruebas unitarias simples con Jest para las funciones de cifrado y las APIs de subida/descarga. Conteneriza la aplicación con un Dockerfile básico.

Desafío:
Manejar el cifrado y descifrado en el cliente, gestionar de forma sencilla las claves públicas y privadas, y asegurar que las pruebas cubran las funcionalidades esenciales de seguridad.

2. Sistema de detección de intrusos en tiempo real con aprendizaje automático

Descripción:
Crea una aplicación que analice un conjunto de datos de tráfico de red preexistente para detectar posibles intrusiones utilizando un algoritmo básico de clustering como K-Means con TensorFlow.js. Utiliza Node.js para procesar los datos y React para visualizar alertas y estadísticas simples. Implementa una API REST básica para manejar los datos. Realiza pruebas unitarias enfocadas en el algoritmo de detección con Jest. Conteneriza el proyecto con un contenedor para el backend y otro para el frontend.

Desafío:
Integrar un algoritmo de aprendizaje automático sencillo, manejar un conjunto de datos estático y asegurar la confiabilidad mediante pruebas básicas.

3. Gestor de contraseñas con cifrado y autenticación OAuth simplificada

Descripción:
Desarrolla una aplicación web que permita a los usuarios almacenar y gestionar sus contraseñas de forma segura. Implementa cifrado simétrico en el frontend utilizando crypto-js. Utiliza React y TypeScript en el frontend, y Node.js con Express en el backend. Almacena las contraseñas cifradas en MongoDB con Mongoose. Utiliza un proveedor de OAuth existente, como Google o GitHub, para manejar la autenticación. Realiza pruebas unitarias básicas con Jest para las funciones de cifrado y autenticación. Conteneriza la aplicación con Dockerfile separados para el backend y frontend.

Desafío:
Garantizar la seguridad en el manejo de contraseñas, integrar OAuth de manera simplificada y cubrir las funcionalidades esenciales con pruebas.

4. Aplicación de recomendaciones personalizadas usando aprendizaje automático simplificado

Descripción:
Construye una aplicación que ofrezca recomendaciones básicas (como productos o contenido) basadas en un conjunto de datos estático utilizando un algoritmo sencillo de filtrado colaborativo con una biblioteca como simple-recommender. Utiliza Next.js y React en el frontend, y Node.js con TypeScript en el backend. Emplea MongoDB para almacenar datos de usuarios y preferencias

predefinidas. Implementa una API GraphQL básica que devuelve recomendaciones estáticas. Escribe pruebas unitarias para el algoritmo de recomendación y pruebas de integración para la API con Jest. Conteneriza todo con Docker, separando backend y frontend.

Desafío:
Implementar un algoritmo de recomendación sencillo, manejar datos estáticos de usuarios y asegurar la funcionalidad básica mediante pruebas.

5. Aplicación de mensajería segura con cifrado de extremo a extremo básico

Descripción:
Crea una aplicación de mensajería en tiempo real donde dos usuarios predefinidos puedan enviar y recibir mensajes cifrados de extremo a extremo utilizando OpenPGP. Utiliza Socket.IO para la comunicación en tiempo real, Node.js en el backend, y React con TypeScript en el frontend. Implementa autenticación básica para los usuarios. Almacena las claves y los mensajes cifrados en MongoDB con Mongoose. Realiza pruebas unitarias para las funciones de cifrado y pruebas de integración para la comunicación en tiempo real con Jest. Conteneriza la aplicación con `Dockerfile` separados para backend y frontend.

Desafío:
Gestionar el cifrado en tiempo real de manera eficiente, asegurar la sincronización de claves entre dos usuarios y cubrir las funcionalidades críticas con pruebas básicas.

6. API de detección de fraude con algoritmos de aprendizaje automático simplificados

Descripción:
Desarrolla una API que analice un conjunto de datos de transacciones financieras preexistente para detectar posibles fraudes utilizando un modelo sencillo de árbol de decisión con una biblioteca de JavaScript adecuada. Implementa la API con Node.js y TypeScript, exponiendo endpoints REST básicos para analizar transacciones y devolver resultados de fraude. Utiliza MongoDB y Mongoose para almacenar datos de transacciones. Implementa autenticación simplificada utilizando tokens estáticos. Escribe pruebas unitarias para el modelo de detección y pruebas de integración para los endpoints de la API con Jest. Conteneriza la API con un único Dockerfile.

Desafío:
Integrar un modelo de detección de fraude sencillo en la API, manejar datos sensibles de forma básica y asegurar que las pruebas validen correctamente el comportamiento principal del sistema.

Aquí tienes una versión simplificada de las rúbricas y las instrucciones para el desarrollo y evaluación de los proyectos, eliminando las tablas para mayor claridad.

Rúbricas

Rúbrica para el trabajo (5 puntos)

1. Funcionalidad (1 punto): El proyecto debe cumplir los requisitos esenciales y ser funcional.
2. Calidad del código (1 punto): Código organizado, claro y con comentarios básicos que sigan buenas prácticas.
3. Implementación técnica (1 punto): Uso adecuado y eficiente de las tecnologías sugeridas.
4. Pruebas (1 punto): Implementación de pruebas unitarias e integración que cubran funcionalidades clave.
5. Contenerización (1 punto): Configuración correcta de Docker para permitir un despliegue sencillo.

Rúbrica para la exposición (7 puntos)

1. Contenido y comprensión (2 puntos): Demuestra un entendimiento completo del proyecto y sus conceptos técnicos.
2. Organización y estructura (1 punto): La presentación debe estar bien organizada, con introducción, desarrollo y conclusión.
3. Implementación técnica (1 punto): Explicación detallada de la implementación, destacando las herramientas y tecnologías.
4. Materiales visuales (1 punto): Uso de materiales visuales como diapositivas o demostraciones que complementen el contenido.
5. Demostración del proyecto (1 punto): Demostración clara y funcional de las principales características.
6. Comunicación (1 punto): Comunicación clara y concisa, manteniendo el interés del público.
7. Respuesta a preguntas (1 punto): Capacidad para responder preguntas mostrando conocimiento y confianza.

Resumen de la calificación

- Trabajo: 5 puntos.
- Exposición: 7 puntos.
- Total: 12 puntos.

Interpretación de la puntuación total

- **9 - 12 puntos: Excelente.**
- **7 - 8 puntos: Bueno.**
- **4 - 6. puntos: Satisfactorio.**
- **1 - 3 puntos: Insuficiente.**
- **0 puntos: No cumple.**

Fecha de presentación: 21 de noviembre