Examen

Este examen evalúa las habilidades técnicas del desarrollador en la creación de una aplicación web de gestión de tareas utilizando un **enfoque full stack**. Se ha elegido **FastAPI** como framework de backend por su simplicidad y curva de aprendizaje rápida, lo que permite a desarrolladores con experiencia en **Python** adaptarse rápidamente incluso sin conocimiento previo específico de Tornado.

Se espera que el candidato demuestre buenas prácticas de desarrollo, incluyendo arquitectura limpia, documentación adecuada, manejo correcto de errores y **pruebas unitarias efectivas.**

Requisitos extendidos:

1. Backend (Fastapi obligatorio):

- Crear una API RESTful para gestionar tareas.
- Implementar autenticación JWT con distintos roles de usuario (admin,
- usuario regular)
- Añadir persistencia en base de datos PostgreSQL con migraciones
- automáticas
- Implementar websockets para actualizaciones en tiempo real **(Opcional)**
- Crear un sistema de notificaciones para tareas próximas a vencer (**Opcional**)
- Añadir paginación, filtrado y ordenamiento avanzado para las tareas
- Implementar rate limiting y protección contra ataques CSRF

Endpoints necesarios:

- GET /tasks: Obtiene la lista de tareas con opciones de filtrado y
- paginación.
- POST /tasks: Crea una nueva tarea.
- PUT /tasks/{id}: Actualiza una tarea existente.
- DELETE /tasks/{id}: Elimina una tarea.
- GET /tasks/statistics: Obtiene estadísticas de tareas
- completadas/pendientes.
- POST /tasks/{id}/comments: Añade un comentario a una tarea.
- Cada tarea debe tener los siguientes campos:
- id: Identificador único.
- title: Título de la tarea.
- description: Descripción de la tarea.
- completed: Booleano que indica si la tarea está completada.
- due_date: Fecha de vencimiento.
- priority: Nivel de prioridad (alta, media, baja).
- created_by: Usuario que creó la tarea.
- assigned_to: Usuario asignado a la tarea.
- created_at: Fecha de creación.
- updated_at: Fecha de última actualización.

2. Frontend con React JS:

- Utilizar TypeScript con tipos estrictos

- Correcta gestion de estados usando herramientas como Redux u a fin
- Usar cualquier Framework de estilos
- Añadir modo oscuro/claro con persistencia de preferencias (**Opcional**)
- Implementar lazy loading y code splitting
- Crear visualizaciones con gráficos para analítica de tareas completadas
- Componentes necesarios:
- Dashboard con métricas de tareas.
- Lista de tareas con filtrado y ordenamiento avanzado.
- Formulario para crear y editar tareas con validación.
- Sección de estadísticas con gráficos.
- Panel de administración para usuarios con rol admin.
- Sistema de notificaciones en tiempo real. (**Opcional**)

3. Pruebas:

- Implementar cobertura mínima del 85% en backend
- Añadir pruebas de componentes con Cypress
- Incluir pruebas para cada endpoint de la API.
- Implementar mocks para los servicios externos.

4. DevOps y Despliegue:

- Crear un Dockerfile para el backend y otro para el frontend.
- Configurar docker-compose.yml para levantar todos los servicios.
- Añadir logs centralizados
- Implementar health checks y circuit breakers

5. Seguridad:

- Añadir protección contra inyección SQL y XSS
- Añadir validación de datos de entrada en todas las APIs

6. Entregables:

- Código fuente del backend y frontend en un repositorio de GitHub
- Dockerfile para backend y frontend.
- docker-compose.yml para levantar los servicios.
- Pruebas unitarias con pytest para el backend.
- Documentación de la API con Swagger/OpenAPI
- Diagrama de arquitectura del sistema
- Instrucciones detalladas para instalación y puesta en marcha
- Reporte de no más de 1 cuartilla de su proceso de desarrollo detallando enfoque utilizado y herramientas utilizadas.

7. Tiempo límite:

- El examen puede ser completado a lo largo de una semana desde la entrega de este documento.

¡Buena suerte y éxito en tu desarrollo!