Prueba Técnica: Desarrollador Web Full Stack

*Versión: 2024*

**Duración máxima:** 2 horas

**Instrucciones generales:**

1. Realiza tu solución y sube los cambios a tu propio repositorio de GitHub. Comparte el enlace con el evaluador.
2. Asegúrate de utilizar buenas prácticas de desarrollo, comentando tu código cuando sea necesario y creando una documentación breve que explique tu solución y el proceso seguido.

# PARTE 1: PHP Y MYSQL (60 MINUTOS)

**Objetivo:** Desarrollar un sistema básico de gestión de usuarios con autenticación y manejo de sesiones.

Requisitos:

1. **Estructura básica:**

- Crea un sistema en PHP utilizando POO (Programación Orientada a Objetos).
- Implementa una funcionalidad para registrar y autenticar usuarios, con los siguientes datos como mínimo: *id, name. email, password.*

– Crea endpoints REST para:

- Registro de usuarios (POST /api/register)
- Autenticación de usuarios (POST /api/login) usando JWT (JSON Web Tokens).
- Obtener el perfil del usuario autenticado (GET /api/profile).

- Asegúrate de aplicar medidas de seguridad básicas, como la encriptación de contraseñas, validación de datos, etc.

– Implementa una funcionalidad para gestionar posts como si fuera un blog, con los siguientes datos como mínimo: id, title, categoryid, content, userid. Se deben poder tener

categorías y subcategorías en los cuales estén asociados los posts (sin importar qué tan profundo pueda ser el arbol de categorías).

– Crea enpoinst REST para:

- Listar todos los posts de una categoría (GET /api/posts/{categoryid})
- Crear un nuevo post (solo usuarios autenticados) (POST /api/posts)
- Actualizar un post (solo el creador del post) (PUT /api/posts/{id})
- Eliminar un post (solo el creador del post) (DELETE /api/posts/{id})
- Listar las categorías contenidas en una categoría superior (GET /api/categories/{parentcategoryid})
- Crear un nueva categoría (solo usuarios autenticados) (POST /api/categories)

2. Manejo de sesiones: JWT (JSON Web Tokens):

– Implementa un sistema de autenticación basado en tokens JWT.
– El token debe generarse al momento del login y debe ser requerido para acceder a cualquier endpoint protegido (excepto el registro y el login).

Autorización:

– Solo el creador de un post puede editarlo o eliminarlo.
– Implementar las políticas necesarias para restringir el acceso a recursos de otros usuarios.

3. Conexión a base de datos:

- El sistema debe conectarse a una base de datos MySQL o PostgreSQL.
- Crea las tablas necesarias para la aplicación y gestiona las conexiones de forma segura y eficiente.

4. Pruebas unitarias:

- Implementa pruebas unitarias para al menos dos de las funcionalidades claves (registro y login de usuarios). Usa PHPUnit o cualquier otro framework de pruebas en PHP.

## PARTE 2: FRONTEND (30 MINUTOS)

Objetivo: Crear una interfaz básica para el sistema de gestión de usuarios.

Requisitos:

1. Crea un formulario en HTML5 y CSS (puedes usar Bootstrap o cualquier framework CSS) para el registro y login de usuarios.
2. Crea una página para visualización de las categorías y los posts. Se debe poder navegar en el arbol de categorías.
3. Usa JavaScript o jQuery para validar los formularios en el frontend antes de enviarlos al backend.
4. Puntos extra si utilizas TypeScript o Angular (desafío extra) para construir una pequeña aplicación frontend que consuma la API REST creada en la Parte 1.

## PARTE 3: CONTROL DE VERSIONES Y DOCUMENTACIÓN (30 MINUTOS)

1. Control de versiones:
- Durante el desarrollo, realiza commits regulares utilizando GIT. Asegúrate de escribir mensajes claros en cada commit.
- Crea un archivo README.md explicando cómo configurar y correr tu proyecto.
- Añade un archivo de documentación donde expliques brevemente el enfoque que seguiste para resolver los retos e incluso un video no mayor a 5 minutos.

Requisitos técnicos:

- PHP: Uso de buenas prácticas, POO y si es posible, aplicar principios SOLID.
- Base de datos: MySQL o PostgreSQL.
- Frontend: HTML5, CSS, JavaScript (JQuery), deseable uso de Angular o Ionic.
- Pruebas unitarias: PHPUnit o framework equivalente.
- Control de versiones: GIT.

Criterios de evaluación:

- Correctitud y eficiencia del código.
- Uso de buenas prácticas (desacoplamiento, modularidad).
- Seguridad en el manejo de datos (contraseñas, manejo de sesiones).
- Capacidad de integrar varias tecnologías y trabajar de manera full-stack.
- Documentación clara y precisa.
- Correcto uso de control de versiones