Prueba Técnica – Desarrollador(a) Laravel

Duración sugerida: 3–4 horas  
Formato de entrega: Word (este documento) + Código en repositorio Git

# Parte C — Conocimientos básicos de servidores

1) ¿Qué permisos asignarías a un archivo de llave privada y por qué?

Respuesta: Permiso 600 lectura y escritura solo para el administrador

2) Comandos para limpiar cache de Laravel.

Respuesta: php artisan cache:clear

3) Cómo mantener workers de cola en producción.

Respuesta:

4) ¿Dónde revisar errores de app y servidor?

Respuesta:

App: storage/logs/laravel.log

Servidor Apache: /var/log/apache2/error.log

5) Esquema básico de .htaccess para SPA.

Respuesta:

<IfModule mod\_rewrite.c>

RewriteEngine On

RewriteBase /

RewriteRule ^index\.php$ - [L]

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule . /index.php [L]

</IfModule>

6) Comando para backup/restore MySQL.

Respuesta: ........................................................................................................................

7) ¿Qué es OPcache y por qué ayuda?

Respuesta: ........................................................................................................................

8) Riesgos de exponer .env y cómo evitarlos.

Respuesta: Filtracion de informacion delicada e importante como accesos a base de datos apis y keys, como evitarlas nunca subir el .env a un repositorio

9) Comando ufw para abrir solo 80/443.

Respuesta: ........................................................................................................................

10) Cómo asegurar permisos de storage y bootstrap/cache.

Respuesta: ........................................................................................................................

# Parte D — SQL

1) Top 3 categorías por promedio de rating.  
Respuesta SQL:

SELECT c.name AS categoria, AVG(r.rating) AS promedio\_rating

FROM restaurants r

JOIN categories c ON r.category\_id = c.id

GROUP BY c.id, c.name

ORDER BY promedio\_rating DESC

LIMIT 3;

2) Conteo de favoritos en restaurantes especiales.  
Respuesta SQL:

SELECT r.name, COUNT(f.id) AS total\_favoritos

FROM restaurants r

LEFT JOIN favorites f ON f.restaurant\_id = r.id

WHERE r.is\_special = 1

GROUP BY r.id, r.name;

3) Restaurantes con nombre duplicado.  
Respuesta SQL:

SELECT name, COUNT(\*) AS cantidad

FROM restaurants

GROUP BY name

HAVING COUNT(\*) > 1;

# Parte E — Debug rápido

Analiza el siguiente snippet y encuentra problemas (escribe tu análisis debajo):

public function index(Request $request)  
{  
 $q = Restaurant::query()->with('category');  
 if ($request->search) {  
 $q->where('name', 'like', "%$request->search%");  
 }  
 if ($request->special) {  
 $q->where('is\_special', $request->special);  
 }  
 return RestaurantResource::collection(  
 Cache::remember('restaurants', 60, fn() => $q->paginate(10))  
 );  
}

Respuesta:

Inyección de cache global: restaurants es fija, no depende de filtro. los usuarios obtendrán el mismo resultado cacheado, ignorando filtros.

SQL Injection: eloquent maneja binding, interpolar $request->search es malo si se cambia a query raw. Es mejor usar binding: ->where('name', 'like', "%{$request->search}%").

# Rúbrica de Evaluación (100 pts)

- A. Lógica/Algoritmia: 20 pts  
- B. Laravel Práctico: 55 pts  
- C. Servidores: 10 pts  
- D. SQL: 10 pts  
- E. Debug: 5 pts  
Bonus: +10 pts (Swagger, scopes, documentación clara, etc.)