Respuestas Teóricas - Evaluación Desarrollador Backend

1.1 Conocimientos Técnicos

Lenguaje preferido:

Mi lenguaje preferido para desarrollo backend es **Python**. Esto se debe a su versatilidad y facilidad de uso, lo que permite desarrollar aplicaciones de manera rápida y eficiente. Python cuenta con una amplia gama de bibliotecas y frameworks como **Django** y **Flask**, que hacen que sea ideal para el desarrollo de APIs. Además, tiene una gran comunidad que contribuye constantemente a mejorar la seguridad, rendimiento, y escalabilidad.

Bases de Datos:

¿Cuál es la diferencia entre una base de datos relacional (SQL) y una no relacional (NoSQL)?

- **Base de datos relacional (SQL)**: Estas bases de datos usan un esquema estructurado con tablas y relaciones. Ejemplos: MySQL, PostgreSQL.
- **Base de datos no relacional (NoSQL)**: No tienen un esquema fijo y permiten manejar grandes volúmenes de datos no estructurados. Ejemplos: MongoDB, Cassandra.

¿Cuándo usaría una sobre la otra?

- Usaría **SQL** cuando necesito garantizar integridad y relaciones consistentes entre datos.
- Usaría **NoSQL** para manejar datos grandes, no estructurados, o cuando se requiere escalabilidad horizontal rápida.

APIs:

¿Cómo manejarías la paginación en una API RESTful?

Manejaría la paginación usando parámetros `limit` y `offset` en las consultas. Por ejemplo:

```
GET /users?limit=10&offset=20
```

Aquí se obtendrían 10 usuarios empezando desde el usuario 21. También podría incluir en la respuesta de la API los enlaces para la siguiente y la anterior página, usando el formato Link en los encabezados HTTP.

Autenticación JWT: Para implementar autenticación con JWT (JSON Web Token):

Un usuario se autentica con credenciales válidas. El servidor genera un JWT y lo envía al cliente. El cliente envía el JWT en el encabezado de las peticiones subsecuentes como:

```
Authorization: Bearer <token>
```

El servidor valida el token en cada solicitud para comprobar la autenticidad y los permisos del usuario antes de otorgar acceso.

Pruebas

¿Qué herramientas utilizas para pruebas unitarias y por qué son importantes en un entorno backend?

He utilizado Jest para pruebas unitarias en aplicaciones backend. Es una herramienta potente y fácil de usar, que permite ejecutar pruebas de manera rápida y ofrece una excelente cobertura de código.

Las pruebas unitarias son esenciales en un entorno backend porque:

- Verifican que cada unidad del código (función o método) funcione correctamente en aislamiento.
- Con un buen conjunto de pruebas, puedes estar seguro de que futuras modificaciones no romperán funcionalidades existentes.
- Permiten refactorizar el código con confianza, ya que las pruebas ayudan a asegurar que el comportamiento no cambia inesperadamente.
- Las pruebas unitarias también actúan como documentación viva, describiendo cómo debería comportarse cada componente del sistema.

1.2 Resolución de Problemas

¿Cómo manejarías una función que tarda mucho debido a consultas ineficientes a la base de datos?

Primero analizaría las consultas SQL para identificar cuellos de botella, usando herramientas como EXPLAIN para optimizar índices, y reestructuraría las consultas más complejas, dividiendo operaciones o aplicando caching en las más repetitivas.

Qué patrones de diseño aplicarías para desacoplar las diferentes capas de una aplicación?

Usaría el Patrón MVC (Modelo-Vista-Controlador) para separar la lógica de negocio de la capa de datos y presentación, manteniendo las capas desacopladas y facilitando mantenibilidad.

1.3 Arquitectura y escalabilidad

¿Qué es una arquitectura basada en microservicios?

Una arquitectura de microservicios divide la aplicación en servicios pequeños e independientes. Cada servicio se ejecuta por separado y puede comunicarse vía APIs.

- **Ventajas:** Escalabilidad independiente, fácil despliegue.
- **Desventajas:** Mayor complejidad en el manejo de servicios y comunicación.

¿Cómo escalarías una aplicación para manejar un aumento masivo de usuarios?

Escalaría horizontalmente añadiendo más instancias de servidores, y usando balanceo de carga para distribuir el tráfico. También implementaría caché en memoria con Redis y optimizaría la base de datos usando sharding.

1.4 Seguridad

¿Cómo prevenirías ataques de inyección SQL en una API?

Usaría consultas preparadas o ORMs (Object Relational Mappers) para evitar concatenación directa de entradas de usuario en consultas SQL.

¿Qué prácticas seguirías para garantizar que las contraseñas se almacenen de manera segura en la base de datos?

Almacenaría las contraseñas de forma segura usando hashing con un algoritmo fuerte como bcrypt.

1.5 Colaboración

¿Cuál ha sido tu experiencia trabajando en equipo en proyectos de desarrollo?

He trabajado en equipos de la universidad usando metodologías ágiles como Scrum y Kanban. Prefiero Scrum por sus sprints cortos y revisiones continuas, lo que permite una entrega rápida de funcionalidades y ajustes constantes según retroalimentación.