«Talento Tech»

Back-End

Java

Clase 16

Buenos Aires Ciudad

Clase 16: Presentación de Proyectos y Corrección en Vivo

Índice

1. Preguntas de entrevistas laborales y respuestas.
2. Presentación de los Proyectos.
3. Demostración de la Funcionalidad del Backend Integrado con el Frontend.
4. Corrección y Feedback en Vivo por Parte del Profesor.
5. Discusión de las Soluciones Implementadas y Buenas Prácticas.
6. Reflexión Final y Recomendaciones para Futuros Desarrollos.
7. Ejemplos de Preguntas de Entrevistas Laborales y Respuestas.
8. Próximos Pasos en el Roadmap de un Desarrollador.
9. Materiales y Recursos Adicionales.
10. Preguntas para Reflexionar.

Objetivos de la Clase

- Presentar los proyectos finales de cada estudiante.
- Demostrar el backend funcionando junto con el frontend.
- Recibir correcciones y comentarios constructivos del profesor.
- Analizar las soluciones implementadas y discutir buenas prácticas.
- Reflexionar sobre el proceso de desarrollo y recibir recomendaciones para futuros proyectos.
- Prepararse para responder preguntas técnicas en entrevistas laborales.

Preguntas de Entrevistas Laborales y Respuestas

Para ayudar a los estudiantes a prepararse para entrevistas técnicas, se incluyen algunas preguntas comunes con sus respuestas y explicaciones.

Java y Spring Boot

1)¿Cuál es la diferencia entre `==` y `.equals()` en Java?

- `==` compara referencias de objetos en la memoria, mientras que `.equals()` compara el contenido de los objetos si está sobrescrito correctamente en la clase.

Ejemplo:
String a = new String("Hola");

String b = new String("Hola");

System.out.println(a == b); // false (distintas referencias en memoria)

- System.out.println(a.equals(b)); // true (contenido igual)

2)¿Qué es la inyección de dependencias en Spring Boot?

- Es un patrón de diseño donde las dependencias son proporcionadas por el framework en lugar de ser creadas manualmente.

Ejemplo con `@Autowired`:

```
@Service
public class UsuarioService {
   @Autowired
   private UsuarioRepository usuarioRepository;
}
```

Bases de Datos y JPA

3)¿Qué es una relación `OneToMany` y cómo se define en JPA?

- Una relación `OneToMany` significa que un registro en una tabla puede estar relacionado con múltiples registros en otra tabla.

Ejemplo:

```
@Entity

public class Usuario {

   @Id

   @GeneratedValue(strategy = GenerationType.IDENTITY)

   private Long id;

   @OneToMany(mappedBy = "usuario", cascade = CascadeType.ALL)

   private List<Pedido> pedidos;

}
```

APIs REST y Seguridad

4)¿Cuáles son los diferentes métodos HTTP y en qué casos se utilizan?

- `GET`: Obtener datos.
- `POST`: Crear recursos.
- `PUT`: Actualizar un recurso existente.
- `DELETE`: Eliminar un recurso.

5)¿Qué es CORS y cómo se soluciona en Spring Boot?

- CORS (Cross-Origin Resource Sharing) es una medida de seguridad que impide que un dominio externo acceda a la API sin permiso.

Solución:

```
@RestController

@RequestMapping("/api")

@CrossOrigin(origins = "http://localhost:3000")

public class ApiController {

   // Controlador

}
```

Frontend y Consumo de APIs

6) ¿Cómo manejar errores en una petición a una API desde el frontend?

Utilizando `try-catch` en JavaScript:

```
fetch("http://localhost:8080/api/usuarios")
 .then(response => {
   if (!response.ok) {
     throw new Error("Error en la API");
   }
   return response.json();
 })
 .catch(error => console.error(error));
```

Próximos Pasos en el Roadmap de un Desarrollador

El aprendizaje en tecnología es continuo, y hay diversas ramas en las que un desarrollador backend puede especializarse. A continuación, se presentan algunos caminos recomendados y tecnologías clave para continuar el desarrollo profesional:

DevOps y Site Reliability Engineering (SRE)

Docker: Es una plataforma de contenedores que permite empaquetar aplicaciones con todas sus dependencias para asegurar que funcionen en cualquier entorno. Facilita el despliegue y escalabilidad de aplicaciones.

- **Ejemplo:** Ejecutar una aplicación Java en un contenedor Docker:

```
docker build -t mi-app .
docker run -p 8080:8080 mi-app
```

Kubernetes: Es un orquestador de contenedores que permite gestionar despliegues escalables y automatizados. Se usa para balanceo de carga, autoescalado y recuperación de fallos.

- **Ejemplo:** Desplegar una aplicación en Kubernetes:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mi-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mi-app
  template:
    metadata:
      labels:
        app: mi-app
    spec:
      containers:
      - name: mi-app
        image: mi-app:latest
```

CI/CD (Integración y Despliegue Continuo): Automatiza el proceso de construcción, prueba y despliegue del software usando herramientas como Jenkins, GitHub Actions y GitLab CI/CD.

- **Ejemplo:** Pipeline de GitHub Actions:

```yaml
name: CI/CD
on: push
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
      - name: Build and Test
        run: mvn test
```

Bases de Datos Modernas

MongoDB: Base de datos NoSQL basada en documentos JSON. Ideal para aplicaciones con datos semiestructurados y escalabilidad horizontal.

- **Ejemplo:** Insertar un documento en MongoDB:

```
{
  "nombre": "Juan",
  "edad": 30,
  "email": "juan@example.com"
}
```

Redis: Base de datos en memoria, utilizada para almacenamiento en caché y optimización del rendimiento.

- **Ejemplo:** Almacenar una clave en Redis:

```
SET usuario:1 "Juan Perez"
GET usuario:1
```

Cassandra: Base de datos distribuida altamente escalable, ideal para aplicaciones con grandes volúmenes de datos.

- **Ejemplo:** Creación de una tabla en Cassandra:

```
CREATE TABLE usuarios (
  id UUID PRIMARY KEY,
  nombre text,
  email text
);
```

Seguridad

JWT (JSON Web Tokens): Utilizado para autenticación de usuarios en APIs REST.

- **Ejemplo:** Generación de un token JWT en Java:

```
String token = Jwts.builder()
  .setSubject("usuario123")
  .signWith(SignatureAlgorithm.HS256, "secreto")
  .compact();
```

- **OAuth 2.0**: Protocolo de autorización utilizado para autenticación segura con terceros.
- **Spring Security**: Framework para la gestión de seguridad en aplicaciones Spring Boot.

Microservicios

- **Spring Boot**: Framework para el desarrollo rápido de aplicaciones Java.
- **Kafka**: Plataforma de mensajería para la comunicación entre microservicios.
- **API Gateway**: Herramientas como Kong o API Gateway de AWS permiten gestionar las peticiones a microservicios.

Cloud Computing

- **AWS (Amazon Web Services)**: Servicios en la nube como EC2, S3, Lambda y DynamoDB.
- **Google Cloud Platform (GCP)** y **Microsoft Azure**: Alternativas con herramientas similares para despliegue en la nube.

Lenguajes Complementarios

- **Node.js**: Para desarrollo backend con JavaScript.
- **Python (Django y Flask)**: Lenguajes populares para el desarrollo web y machine learning.
- **Rust y Go**: Lenguajes eficientes y seguros para aplicaciones de alto rendimiento.

Inteligencia Artificial

- **TensorFlow y PyTorch**: Frameworks para construir modelos de machine learning y deep learning.
- **OpenAI APIs**: Para integración con inteligencia artificial generativa.

Materiales y Recursos Adicionales

- Documentación oficial de Java y Spring Boot.
- Guías sobre pruebas automatizadas con JUnit y Mockito.
- Tutoriales sobre seguridad en APIs con JWT y OAuth.

Preguntas para Reflexionar

- ¿Qué tecnologías me interesan para seguir aprendiendo?
- ¿Cuáles son las habilidades más demandadas en el mercado laboral?
- ¿Debería especializarme en DevOps, Cloud, Microservicios o Seguridad?
- ¿Qué herramientas puedo implementar en mi próximo proyecto personal?


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad