Evolución y Gestión de la Configuración

Prueba de mínimos

Grado en Ingeniería Informática – Ingeniería del Software · Universidad de Sevilla · Curso 2026-2027

Como parte de un proceso selectivo, te han citado a una entrevista con el equipo que implanta la filosofía DevOps en una empresa tecnológica. Te harán preguntas de 5 bloques temáticos: en cada bloque, una pregunta conceptual y otra técnica. Varias preguntas técnicas se refieren a la aplicación web de gestión de películas (hecha con Flask) que desarrolla la empresa.
Cada pregunta vale hasta 1 punto. Para obtener la calificación de **apto** tienes que cumplir dos criterios:

1. no obtener 0 puntos en ningún bloque temático; y
2. sumar al menos 5 puntos en total.

Importante: responde cada pregunta únicamente dentro de su recuadro en las hojas de respuesta, al final del cuadernillo (a partir de la página 4). No se corregirá nada que se escriba fuera del recuadro ni en folios aparte.
¡Aquí comienza la entrevista! ¡Mucha suerte!

Bloque 1 · CI/CD

1.1 *(Conceptual. 1 punto. Responde en la página 4.)* Una persona recién incorporada a la empresa se ha unido al equipo que desarrolla un reproductor de vídeo para escritorio. Los usuarios descargan e instalan manualmente cada nueva versión en sus ordenadores, y el equipo mantiene a la vez varias versiones del reproductor para cada sistema operativo soportado. En su primera reunión, esta persona propone pasar a despliegue continuo "porque es la última tendencia en la industria".

a) Explica la diferencia entre entrega continua y despliegue continuo.

b) ¿Encaja su propuesta con este producto? ¿Por qué?

1.2 *(Técnica. 1 punto. Responde en la página 4.)* La aplicación de películas se despliega automáticamente en preproducción con este workflow de GitHub Actions:

```
1  name: Deploy to preproduction
2  on:
3    push:
4      branches: [trunk]
5  jobs:
6    testing:
7      runs-on: ubuntu-24.04
8      steps:
9        - uses: actions/checkout@v5
10       - run: pip install -r requirements.txt
11       - run: pytest
12   deploy:
13     needs: testing
14     runs-on: ubuntu-24.04
15     steps:
16       - run: curl "${{ secrets.RENDER_PREPROD_DEPLOY_HOOK_URL }}"
```

a) ¿Qué ocurre con el despliegue si falla alguna prueba? ¿Qué línea del fichero lo determina?

b) En Render ya está configurado el servicio de producción, vinculado a la rama `main`, que solo se despliega a través de su *hook*. La URL de ese *hook* está guardada en el secreto `RENDER_PROD_DEPLOY_HOOK_URL`. Quieren un workflow equivalente que despliegue en **producción** cada vez que se actualice la rama `main`. Indica qué líneas cambiarías (por su número) y cómo quedarían.

Bloque 2 · Gestión del código fuente

2.1 *(Conceptual. 1 punto. Responde en la página 4.)* Un compañero del equipo del reproductor de vídeo ha terminado un módulo que llama a la función `calcular_tasa_reproduccion`. Antes de subir su trabajo, se trae los últimos cambios del repositorio compartido y Git completa el *merge* sin avisar de ningún conflicto. Sin embargo, al pasar las pruebas, una falla: otra persona había renombrado esa función a `calcular_velocidad_reproduccion`.

a) ¿Por qué Git no ha detectado ningún conflicto? ¿Qué tipo de conflicto es este?

b) Además de las pruebas automatizadas, ¿qué otras prácticas ayudan a detectar este tipo de problemas cuanto antes?

2.2 *(Técnica. 1 punto. Responde en la página 5.)* El equipo de la aplicación de películas integra con frecuencia las tareas terminadas en `trunk`; a `main` solo se llevan las entregas, que se etiquetan. Cada tarea se desarrolla en su propia rama. Una persona recién incorporada te enseña cómo ha subido la función `delete_movie` que acaba de programar:

```
$ git switch trunk
$ git status
On branch trunk
Changes not staged for commit:
        modified:   app/models.py
        modified:   app/tests/test_unit.py
Untracked files:
        .env
        app/__pycache__/
$ git add .
$ git commit -m "Añade delete_movie para eliminar películas por título"
$ git push
 ! [rejected]        trunk -> trunk (fetch first)
error: failed to push some refs to 'github.com:empresa/movies.git'
$ git push --force
```

a) Identifica **dos** problemas en esta secuencia y explica qué consecuencia puede tener cada uno.

b) Tras el rechazo del primer `git push`, ¿qué debería haber hecho en lugar de `git push --force`?

Bloque 3 · Pruebas

3.1 *(Conceptual. 1 punto. Responde en la página 5.)* Un departamento de la empresa ha conseguido, con ayuda de una herramienta basada en un LLM, que sus pruebas ejecuten el 100 % de las líneas de código. Por ello, afirma que su producto ya no tiene defectos.

a) Define con precisión qué son un error, un defecto y un fallo.

b) ¿Es correcta la afirmación del departamento? ¿Por qué?

3.2 *(Técnica. 1 punto. Responde en la página 5.)* La aplicación de películas recomienda películas según la nota media que obtiene de un servicio externo de valoraciones. El módulo `app/ratings.py` contiene estas funciones:

```
import requests
def get_rating(title):
    """Consulta la nota media de la pelicula en un servicio externo."""
    response = requests.get("https://api.notas-peliculas.example/rating",
                            params={"title": title})
    return response.json()["rating"]

def is_recommended(title):
    """Una pelicula se recomienda si su nota media es 7 o superior."""
    return get_rating(title) >= 7
```

Para probar `is_recommended`, el equipo ha escrito esta prueba unitaria:

```
from app import ratings
def test_gladiator_es_recomendada():
    assert ratings.is_recommended("Gladiator")
```

a) ¿Te parece una buena prueba unitaria? ¿Por qué?

b) ¿Cómo la mejorarías? Explícalo con palabras; no hace falta que escribas código.

Bloque 4 · Tareas e incidencias

4.1 *(Conceptual. 1 punto. Responde en la página 6.)* Un departamento de la empresa resuelve sus incidencias en solo dos fases: **diagnosticar** (formular hipótesis y comprobarlas con experimentos) y **reparar** (corregir el defecto y añadir una prueba de regresión). ¿Crees que deberían añadir alguna fase más? En caso afirmativo, indica el nombre de cada fase que añadirías y describe brevemente en qué consiste.

4.2 *(Técnica. 1 punto. Responde en la página 6.)* Un usuario ha enviado esta incidencia sobre la aplicación de películas:

```
# El sistema no funciona

No puedo crear películas.

## Entorno
Uso Ubuntu.

## Resultado esperado
Que funcione.

## Resultado obtenido
Sale un error.
```

Reescribe la incidencia para que el equipo pueda reproducir el problema. Incluye al menos un resumen, los pasos para reproducirlo, el resultado esperado, el resultado observado y el entorno. Puedes inventar los datos concretos que necesites.

Bloque 5 · Aislamiento

5.1 *(Conceptual. 1 punto. Responde en la página 6.)* Una persona del equipo de operaciones nos ha escrito: *"Hemos metido la aplicación en un contenedor Docker, así que está tan aislada como si usáramos una máquina virtual"*. ¿Estás de acuerdo? Explica la diferencia principal entre un contenedor y una máquina virtual, e indica una ventaja de cada uno.

5.2 *(Técnica. 1 punto. Responde en la página 7.)* Para desplegar la aplicación de películas, el equipo de operaciones ha escrito este `Dockerfile`. La imagen se construye de nuevo en cada despliegue y, aunque el código no cambie, dos construcciones hechas en días distintos no siempre se comportan igual.

```
1 FROM python:latest
2 WORKDIR /app
3 COPY . .
4 RUN pip install flask gunicorn
5 EXPOSE 5000
6 CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]
```

a) ¿Qué líneas provocan este comportamiento? ¿Por qué?

b) Reescribe esas líneas para reducir esa variabilidad fijando versiones. Basta con fijar lo que aparece en el fichero.

Hojas de respuesta

(EN MAYÚSCULAS)

Apellidos y nombre: _____

UVUS: _____ Grupo: _____ Fecha: _____

Responde cada pregunta únicamente dentro de su recuadro. Las líneas son solo una guía.

Bloque 1 · CI/CD

Respuesta 1.1

Respuesta 1.2

Bloque 2 · Gestión del código fuente

Respuesta 2.1

Respuesta 2.2

Bloque 3 · Pruebas

Respuesta 3.1

Respuesta 3.2

Bloque 4 · Tareas e incidencias

Respuesta 4.1

Respuesta 4.2

Bloque 5 · Aislamiento

Respuesta 5.1

Respuesta 5.2