



Listado de Preguntas Técnicas

HTML

1. ¿Cuál es la forma correcta de crear un enlace que abra en una nueva pestaña?
 - a) ``
 - ☒ b) ``
 - c) ``
 - d) ``
2. ¿Qué etiqueta semántica HTML5 se usa para contenido independiente?
 - a) `<section>`
 - b) `<div>`
 - ☒ c) `<article>`
 - d) `<aside>`

CSS

3. ¿Qué propiedad CSS se usa para crear un diseño de cuadrícula?
 - a) `display: flex`
 - ☒ b) `display: grid`
 - c) `display: table`
 - d) `display: inline-grid`
4. ¿Cuál es la especificidad más alta en CSS?
 - a) Elemento
 - b) Clase
 - c) ID



☒ d) Estilos inline

JavaScript

5. ¿Qué método se usa para agregar un elemento al final de un array?

a) array.add()

☒ b) array.push()

c) array.append()

d) array.insert()

6. ¿Cuál es la diferencia principal entre let y const?

a) let es para números, const para strings

☒ b) let permite reasignación, const no

c) const es más rápido que let

d) No hay diferencia

React

7. ¿Qué hook se usa para manejar efectos secundarios en React?

a) useState

☒ b) useEffect

c) useContext

d) useReducer

8. ¿Cómo se pasan datos de un componente padre a hijo en React?

a) A través de state

☒ b) A través de props

c) A través de context

d) A través de refs



Vue

9. ¿Qué directiva de Vue se usa para renderizado condicional?

a) v-show

☒ b) v-if

c) v-display

d) v-render

10. ¿Cómo se define una propiedad reactiva en Vue 3 Composition API?

a) reactive()

b) ref()

c) computed()

☒ d) Todas las anteriores pueden usarse

Angular

11. ¿Qué decorador se usa para definir un componente en Angular?

a) @NgModule

b) @Injectable

☒ c) @Component

d) @Directive

12. ¿Qué sintaxis se usa para enlazar propiedades en Angular?

a) {property}

b) (property)

☒ c) [property]

d) {{property}}



Node.js

13. ¿Qué objeto global de Node.js permite acceder a variables de entorno?

- ☒ a) process.env
- b) global.env
- c) system.env
- d) node.env

14. ¿Qué módulo de Node.js se usa para trabajar con rutas de archivos?

- a) fs
- ☒ b) path
- c) url
- d) file

Express

15. ¿Qué método de Express se usa para manejar peticiones POST?

- ☒ a) app.post()
- b) app.get()
- c) app.send()
- d) app.request()

16. ¿Cómo se define un middleware en Express que se ejecute en todas las rutas?

- a) app.middleware()
- ☒ b) app.use()
- c) app.all()
- d) app.apply()



Django

17. ¿Qué comando se usa para crear migraciones en Django?

- a) python manage.py migrate
- ☒ b) python manage.py makemigrations
- c) python manage.py createmigrations
- d) python manage.py db migrate

18. ¿Qué ORM usa Django por defecto?

- a) SQLAlchemy
- b) Sequelize
- ☒ c) Django ORM
- d) TypeORM

Flask

19. ¿Qué decorador se usa para definir una ruta en Flask?

- a) @app.path()
- ☒ b) @app.route()
- c) @app.url()
- d) @app.endpoint()

20. ¿Cómo se habilita el modo debug en Flask?

- a) app.debug = True
- b) app.run(debug=True)
- c) flask.debug()
- ☒ d) Ambas a y b son correctas



Spring Boot

21. ¿Qué anotación se usa para marcar una clase como controlador REST en Spring Boot?

- a) @Controller
- ☒ b) @RestController
- c) @RequestMapping
- d) @Service

22. ¿Qué archivo se usa para configurar propiedades en Spring Boot?

- a) config.properties
- ☒ b) application.properties
- c) settings.properties
- d) spring.properties

API REST

23. ¿Qué código de estado HTTP indica que un recurso fue creado exitosamente?

- a) 200
- ☒ b) 201
- c) 204
- d) 202

24. ¿Qué método HTTP se usa para actualizar completamente un recurso?

- a) POST
- b) PATCH
- ☒ c) PUT
- d) UPDATE



SQL

25. ¿Qué cláusula SQL se usa para filtrar resultados agrupados?

- a) WHERE
- b) FILTER
- ☒ c) HAVING
- d) GROUP BY

26. ¿Qué tipo de JOIN devuelve todas las filas de ambas tablas?

- a) INNER JOIN
- b) LEFT JOIN
- c) RIGHT JOIN
- ☒ d) FULL OUTER JOIN

n8n

27. ¿Qué es n8n?

- a) Una base de datos NoSQL
- ☒ b) Una herramienta de automatización de flujos de trabajo
- c) Un framework de JavaScript
- d) Un servidor web

28. ¿Cómo se conectan los nodos en n8n?

- a) Mediante código JavaScript
- b) Mediante archivos de configuración
- ☒ c) Mediante conexiones visuales (drag and drop)
- d) Mediante comandos CLI



Ejercicio Práctico: Sistema de Gestión de Proyectos y Tareas

Problema

Eres parte del equipo de desarrollo de **TaskFlow**, una startup que construye herramientas de gestión de proyectos para equipos pequeños y medianos.

Actualmente, los equipos usan una combinación de hojas de cálculo, mensajes de chat y notas dispersas para gestionar proyectos, lo que genera:

- Pérdida de información en conversaciones
- Falta de visibilidad del estado de proyectos
- Duplicación de esfuerzos
- Dificultad para priorizar tareas

Tu misión es construir un MVP que permita a los equipos crear proyectos, asignar tareas a miembros y colaborar mediante comentarios.

Lo que Debes Construir

Una aplicación web full-stack donde:

1. Los usuarios puedan **registrarse e iniciar sesión** de forma segura
2. Los líderes de proyecto puedan **crear proyectos** y agregar miembros del equipo
3. Los miembros puedan **ver todos los proyectos** en los que participan
4. Cada proyecto tenga un **tablero tipo Kanban** con tareas en tres columnas:

"Por hacer", "En progreso", "Completada"

5. Las tareas puedan **asignarse a miembros** con prioridades (alta/media/baja) y fechas límite
6. Los usuarios puedan **agregar comentarios** en las tareas

Funcionalidades Requeridas

Autenticación y Usuarios

- Registro con email y contraseña
- Login con JWT (access token + refresh token)
- Rutas protegidas en frontend
- Ver y editar perfil personal



Proyectos

- Crear proyecto (nombre, descripción, fecha inicio/fin, color personalizado)
- Listar mis proyectos con búsqueda
- Ver detalle de proyecto con tablero Kanban
- Editar proyecto
- Archivar proyecto (soft delete)
- Agregar/remover miembros del proyecto
- Tareas
- Crear tarea en un proyecto (título, descripción, prioridad, fecha límite, asignar a miembro)
- Ver tareas organizadas en tablero Kanban ("Por hacer", "En progreso", "Completada")
- Editar estado de tarea ("Por hacer", "En progreso", "Completada")
- Eliminar tarea
- Filtrar tareas por estado o por título

Comentarios

- Agregar comentarios en una tarea
- Ver historial de comentarios con fecha y usuario

Stack Tecnológico Requerido

Frontend

Elige **UNA** de las siguientes opciones:

Opción 1: React

Opción 2: Vue

Opción 3: Angular

Backend

Elige **UNA** de las siguientes opciones:

Opción 1: Node.js + Express

Opción 2: Django

Opción 3: Flask

Opción 4: Spring Boot



Base de datos

Elige **UNA** de las siguientes opciones:

Opción 1: MySQL

Opción 2: PostgreSQL

Opción 3: SQL Server

Opción 4: MongoDB

Requisitos Técnicos Obligatorios

Backend

1. **Autenticación JWT** con access y refresh tokens
2. **Middleware de autenticación** en todas las rutas protegidas
3. **Middleware de autorización** (solo miembros del proyecto pueden ver/editar)
4. **Hash de contraseñas** con bcrypt (o similar)
5. **Validación de datos** en todos los endpoints
6. **Manejo de errores** centralizado con códigos HTTP apropiados
7. **Variables de entorno** para secrets (.env)
8. **Soft delete** en proyectos (no eliminar físicamente)
9. **Queries con JOINS** para obtener datos relacionados

Frontend

1. **Rutas protegidas** con guards/middlewares
2. **Interceptor HTTP** para agregar JWT automáticamente
3. **Manejo de refresh token** automático
4. **Estados de carga** (spinners)
5. **Manejo de errores** con notificaciones visuales
6. **Validación de formularios** en tiempo real
7. **Responsive design** (mobile, tablet, desktop)
8. **Estado global** para usuario y autenticación
9. **Lazy loading** de rutas (recomendado)



Validaciones Críticas

Backend debe validar:

- Email único y formato válido
- Contraseña mínimo 8 caracteres
- Usuario solo ve proyectos donde es miembro
- Solo creador puede agregar/remover miembros
- No asignar tareas a no-miembros del proyecto
- Fechas de proyecto coherentes (fin \geq inicio)
- Estados y prioridades solo valores permitidos

Frontend debe validar:

- Campos requeridos no vacíos
- Formato de email válido
- Contraseñas coinciden en registro
- Fechas no en el pasado

Entregables

1. **Repositorio Git** con:
 - Código fuente completo (frontend + backend)
 - .gitignore apropiado
 - Commits descriptivos y atómicos
 - Branches: main y develop mínimo
2. **README.md** que incluya:
 - Descripción del proyecto
 - Tecnologías utilizadas y versiones
 - Requisitos previos (Node.js, Python, Java, etc.)
 - Instrucciones de instalación paso a paso
 - Variables de entorno necesarias (con .env.example)
 - Comandos para ejecutar el proyecto
 - Estructura de la base de datos (puede ser diagrama o descripción)
 - Decisiones técnicas importantes que tomaste
3. **Documentación de API:**
 - Puede ser: Swagger/OpenAPI, Postman collection, o archivo markdown
 - Debe incluir todos los endpoints con ejemplos
4. **Base de datos con datos de ejemplo:**
 - Script SQL
 - Al menos 2 usuarios, 2 proyectos, 10 tareas