1.- Un Panel de Configuración (aside) que contenga un formulario con los siguientes campos:
Un fieldset para configurar la interfaz con un campo legen con el texto “Configuración Visual”. Dentro se incluirá:

- Un h4 con el texto en negrita “Tema de interfaz”
- Tres inputs “radio” para elegir tema: **Claro** (por defecto), **Oscuro** (clases dark-mode del css), **Alto Contraste** (fondo negro y letra amarilla).
- Un input “range” para cambiar el tamaño de fuente global. Permitirá valores entre 10 y 24 de máximo. Valor inicial: 16;
- Un botón para “Resetear Configuración”.

2.- Otro fieldset con un campo legend con el texto “Alta de Empleado” que contenga:

- Inputs para: Nombre, Email, Pass, Confirmar Pass.
- Un campo para introducir una url con el perfil de Linkedin.
- Un checkbox de “Acepto térmicos”.
- Un select múltiple para “Habilidades” (JS, CSS, HTML, Python, SQL). [Ojo la propiedad **selectedOptions**].
- Un botón de envío con el texto “Registrar empleado”. Se le aplicará desde js el siguiente estilo: margin-top: 15px; width: 100%; background: #007bff; color: white; padding: 10px; border: none; cursor: pointer;

Todos estos campos <u>son obligatorios</u> y deben tener válidos y rellenos correctamente cuando se pulsa el botón de Registrar empleado.
Nota: Todos los elementos deben tener sus etiquetas label asignados mediante métodos del DOM.

Zoom Dinámico: Al mover el range todo el texto de la interfaz debe agrandarse o empequeñecerse mientras se desplaza a un lado o a otro.

Temas de la interfaz: Al seleccionar de los temas cambiará el estilo de la interfaz.
El tema Claro deja fondo blanco y letra negra, el tema Oscuro deja fondo negro y letra blanca y el tema Alto Contraste deja el fondo negro y la letra amarilla. Si se vuelve a seleccionar el tema que ya es el que está viéndose, se lanzará un error en un bloque try-catch y se notificará en el h3 con id=”er”, evitando la reaplicación.

3.- Lógica de Negocio y Validación

- **Validación de URL:** El campo Linkedin debe validarse usando una Expresión Regular, que asegure que empieza por https://www.linkedin.com/ o https://linkedin.com/ .
- **Validación de Password:** El password debe tener mayúsculas, minúsculas, números y otros símbolos ($%&...), y tener una longitud de, al menos, 8 caracteres. A medida que se va escribiendo el password, se verificarán estas condiciones. Para mostrar las condiciones que se van mostrando, se muestra el estado en una lista dinámica creada justo encima del password. Las reglas que se vayan cumpliendo mientras se escribe el password se irán poniendo de color verde y tachado (textDecoration = ‘line-throught’).
- **Cálculo de Antigüedad:** Usa el objeto Date para calcular días exactos trabajados.

- **Los errores producidos en el formulario se mostrarán en el h1 con el id=**”error” ya creado en index y, además, durarán 3 segundos mostrándose.

4.- La “Fábrica de Tarjetas”
Al registrar un empleado, debes crear una “Tarjeta de Empleado” (article) nodo a nodo:

- Crear un div cabecera con el Nombre(H3) y el email y el número de días en la empresa (small).
- Crear un meter o progress que indique aleatoriamente el nivel de “eficiencia” del empleado con un valor de 0 a 100.
- Crear una lista ul con las habilidades seleccionadas de la lista múltiple.
- Crear un botón “Clonar Ficha”. Clonará la ficha con los mismos datos, se verá con un 70% de opacidad para diferenciar las fichas clonadas de las demás.
- Crear un botón “Despedir” para borrar la ficha del empleado.

Nota: los nodos clonados no heredan los eventos.

5.- Venta de informe
Botón “Generar informe”.

- Abre una ventana 500x400 con el fichero informe.html ya creado y escribir en ella un resumen de cuántos empleados hay, la fecha actual del informe y los nombres y eficiencia de cada empleado.

Cuidado!! Asegúrate de cargar todo el HTML de informe.html antes acceder a él.

const regexLinkedin = /^https:\/\/(www\.)?linkedin\.com\//;

- `^` → empieza por…
- `https:\/\/` → literal `https://`
- `(www\.)?` → el `www.` es opcional
- `linkedin\.com` → dominio obligatorio
- `\/` → debe ir seguido de `/`
- `/` final → delimita la expresión

```
function validarLinkedin() {

  const link = document.querySelector("input[placeholder='https://linkedin.com']").value;

  const regex = /^https:\/\/(www\.)?linkedin\.com\//;

  if (!regex.test(link)) {

    alert("La URL de LinkedIn no es válida.");

    return false;

  }

  return true;

}
```

Posible Examen 2024

Sistema de Gestión de Tareas – WebTasks.js

1️⃣ Estructura general de la aplicación

La aplicación estará compuesta por:

- Un **sidebar (aside)** fijo con:
  - Nombre/logo de la aplicación
  - Menú de navegación
- Un **main** donde se mostrará:
  - Sección de proyectos
  - Sección de tareas
- Uso de **HTML + CSS + JavaScript modular (type="module")**

📌 **No se permite el uso de librerías externas**

2️⃣ Gestión de tareas (CRUD completo)

La aplicación debe permitir:

🔹 Alta de tareas

Mediante un **formulario dinámico creado desde JavaScript**, con los siguientes campos (deducidos de `main.js`):

- Nombre de la tarea
- Descripción
- Prioridad
- Responsable
- Fecha
- Estado

📌 El formulario:

- **No está fijo en el HTML**
- Se inserta dinámicamente en el DOM
- Incluye un botón **Validar / Guardar**

🔹 Validación de formulario

Al pulsar el botón de validar:

- Todos los campos son **obligatorios**
- Se valida:
  - Campos vacíos

        - Formato correcto de fecha
    - Se contabilizan errores
    - Si hay errores:
        - No se crea la tarea
    - Si todo es correcto:
        - Se crea un **objeto Tarea**
        - Se añade a la tabla

📌 Uso de:

- `preventDefault`
- Contadores de errores
- Manipulación directa del DOM

3️⃣ Modelo de datos (Programación orientada a objetos)

Se debe crear una clase:

📦 `Tarea` (archivo `Tarea.js`)

- Constructor con atributos:
    - nombre
    - descripción
    - prioridad
    - responsable
    - fecha
    - estado

- Métodos para:
    - Mostrar la tarea
    - Actualizar datos

📌 Se evalúa:

- Uso de `class`
- `export` / `import`
- Separación modelo–lógica

4️⃣ Visualización de tareas

Las tareas se muestran en:

- Una **tabla HTML**
- Cada fila representa una tarea
- Cada fila incluye:
    - Botón **Editar**
    - Botón **Eliminar**

📌 Los botones:

- Se gestionan con arrays (`Array.from`)
- Se asignan eventos dinámicamente

5️⃣ Modificación de tareas

Al pulsar **Editar**:

- Se carga el formulario con los datos actuales
- Se permite modificar la tarea
- Se actualiza la fila correspondiente

📌 Importante:

- No se recarga la página
- Se reutiliza el mismo formulario

6️⃣ Eliminación de tareas

Al pulsar **Eliminar**:

- Se elimina la fila de la tabla
- Se actualiza el DOM
- Se gestiona mediante eventos asociados dinámicamente

7️⃣ Ventana secundaria (MUY IMPORTANTE EN EL EXAMEN)

Desde la aplicación principal:

- Se abre una **ventana secundaria** (`window.open`)
- Tamaño controlado
- Se carga `vSec.html`
- Se gestiona desde `vSec.js`

La ventana secundaria se usa para:

- Mostrar información adicional
- Simular vista extendida / detalle

📌 Se evalúa:

- Comunicación entre ventanas
- `onload`
- Control de referencias a ventanas abiertas

8️⃣ Gestión avanzada del DOM

Durante todo el examen se evalúa:

- `getElementById`
- `getElementsByTagName`
- `getElementsByClassName`
- Creación y borrado de nodos
- Inserción dinámica de formularios
- Reutilización de componentes

9️⃣ CSS y diseño

No se evalúa diseño avanzado, pero sí:

- Uso correcto de clases
- Diferenciación visual de secciones
- Uso de un segundo CSS para la vista secundaria

📚 ¿QUÉ TEMAS ENTRABAN REALMENTE EN ESTE EXAMEN?

🔥 Imprescindibles

- DOM puro (crear, modificar, eliminar nodos)
- Eventos (`onclick`, `addEventListener`)
- Formularios dinámicos
- Validación manual
- Clases JS (`class`)
- Módulos JS (`import/export`)
- Tablas dinámicas
- `window.open` y ventanas secundarias

⚠️ Nivel medio-alto

- Reutilización de formularios
- Edición de elementos existentes
- Control de estado de la aplicación
- Separación lógica / modelo