

# PRUEBA TÉCNICA – AUTOMATION DEVELOPER II

## IN TRUCKS

### 1. Ejercicio 1 – Python (Algoritmia – LeetCode)

Resolver el siguiente ejercicio en Python:

**Letter Combinations of a Phone Number**

Link: <https://leetcode.com/problems/letter-combinations-of-a-phone-number/>

Entregar como: python/ejercicio\_1\_leetcode.py

### 2. Ejercicio 2 – Python (Procesamiento CSV → JSON)

- Leer input/sample.csv
- Procesar name, email, city
- Generar output/data.json
- Manejo de errores obligatorio

Entregar como: python/ejercicio\_2\_csv\_automation.py

### 3. Ejercicio 3 – Python (Automatización + API)

API: <https://dummyjson.com/products>

- Consumir API y filtrar por categoría
- Guardar CSV en output/resultado.csv
- Manejar errores HTTP (timeout, 404, 500)

Entregar como: python/ejercicio\_3\_api\_automation.py

### 4. Ejercicio 4 – React + Vite (3 componentes)

**UserList.jsx**: fetch de <https://jsonplaceholder.typicode.com/users>, loading, error, useState/useEffect

**SearchBar.jsx**: input controlado + filtro por nombre

**Formulario.jsx**: validaciones (nombre≥3, email regex, ciudad requerida).

### 5. Estructura esperada del repositorio

```
root/
├── python/
│   ├── ejercicio_1_leetcode.py
│   ├── ejercicio_2_csv_automation.py
│   └── ejercicio_3_api_automation.py
├── output/ (json/csv generados)
├── react/ (proyecto Vite)
├── input/sample.csv
└── README.md
```

### 6. Documentación requerida

- Cómo ejecutar Python
- Cómo ejecutar React
- Manejo de errores implementado
- Decisiones técnicas
- Tiempo aproximado invertido

### 7. Modalidad de entrega

Enviar enlace a un repositorio Git con toda la estructura final.

