🧠 Resultados de aprendiazaje - CRM Funnel

Programación

1. Elementos fundamentales

Ruta: `main.js`

- Variables: `let nombre`
- Constantes: `const MAX_CONTACTS`
- Operadores: `+`, `-`, `&&`, `===`
- Tipos: string, number, boolean, array, objeto

```
const MAX_CONTACTS = 100;
let nombre = "Ana";
let edad = 25;
```

2. Estructuras de control

Ruta: `main.js`

- Selección: `if`, `else`, `switch`
- Repetición: `for`, `forEach`, `while`

```
for (let contacto of contactos) {
  if (contacto.activo) {
    console.log(contacto.nombre);
  }
}
```

3. Control de excepciones

Ruta: `api_contacts.js`

- Uso de `try-catch` para manejar errores al leer/escribir archivos.

```
try {
  let datos = fs.readFileSync("data/contacts.json");
} catch (error) {
  console.error("Error al leer los contactos:", error);
}
```

4. Documentación del código

Ruta: `funnels/funnelLogic.js`

- Comentarios explicativos con `//`.

```
// Añade un nuevo contacto al embudo de ventas
function addToFunnel(contacto) {
  // ...
}
```

5. Paradigma aplicado

Ruta: `main.js`

- Enfoque estructurado y modular.
- Lógica dividida en funciones y módulos.
- No POO por simplicidad.

6. Clases y objetos principales

Ruta: `main.js`

- Objetos clave: contactos.

```
let contacto = {
  nombre: "Luis",
  email: "luis@email.com",
  activo: true,
};
```

7. Conceptos avanzados

- Modularidad y reutilización de funciones.
- Sin herencia ni polimorfismo (posible en el futuro).

8. Gestión de información y archivos

Ruta: `api_contacts.js`

- Lectura/escritura en JSON (`data/contacts.json`).
- Interfaz web con formularios y JS.

```
fs.writeFileSync("data/contacts.json", JSON.stringify(contactos));
```

9. Estructuras de datos

Ruta: `main.js`

- Arrays para listas de contactos.
- Objetos para cada registro.

```
let contactos = [];
```

10. Técnicas avanzadas

Ruta: `main.js`

- Expresiones regulares para validar emails.
- Flujos de E/S para archivos.

```
let emailValido = /^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(contacto.email);
```

Sistemas Informáticos

1. Hardware

- Desarrollo: MacBook (Intel/Apple Silicon, 8GB+ RAM)
- Producción: Servidor Linux básico

2. Sistema operativo

- macOS para desarrollo
- Linux para producción (estable y compatible con Node.js)

3. Redes

- Red local en desarrollo
- Despliegue en Internet
- Protocolos: HTTP/HTTPS
- Firewall configurado

4. Copias de seguridad

- Backups periódicos de `data/contacts.json` y repositorio Git
- Restauración rápida ante errores

5. Seguridad e integridad

- Validación de entradas
- Contraseñas seguras
- Permisos mínimos en archivos y usuarios

6. Usuarios y permisos

- Usuarios configurados en SO y app
- Permisos mínimos necesarios

7. Documentación técnica

- Archivos markdown (`README.md`)
- Comentarios en el código

Entornos de Desarrollo

1. IDE

- Visual Studio Code
- Extensiones: JavaScript, HTML, CSS, Git

2. Automatización de tareas

Ruta: `package.json`

- Scripts npm para iniciar servidor y dependencias

```
"scripts": {
  "start": "node app.js"
}
```

3. Control de versiones

- Git y GitHub
- Ramas para features y correcciones

4. Refactorización

- Mejoras periódicas de eficiencia y legibilidad

5. Documentación técnica

- Markdown (`README.md`)
- Comentarios en el código

6. Diagramas

- Diagramas de flujo/estructura para planificar lógica y arquitectura

Bases de Datos

1. SGBD

- Archivos JSON por simplicidad
- Escalable a MySQL/MongoDB si crece

2. Modelo entidad-relación

- Contactos: nombre, email, estado

3. Funcionalidades avanzadas

- Sin vistas ni procedimientos (archivos planos)

4. Protección y recuperación de datos

- Backups y validaciones para evitar pérdidas

Lenguajes de Marcas y Gestión de Información

1. Estructura HTML

Ruta: `views/index.html`

- Etiquetas semánticas (`<header>`, `<main>`, `<footer>`)
- Buenas prácticas

```
<header>
  <h1>CRM</h1>
</header>
<main>
```

2. Tecnologías frontend

Ruta: `views/styles/style.css`, `views/script.js`

- CSS para diseño
- JavaScript para interactividad

3. Interacción con el DOM

Ruta: `views/script.js`

- JS para modificar el DOM dinámicamente

```
document
  .getElementById("btnAgregar")
  .addEventListener("click", agregarContacto);
```

4. Validación

- Validación con herramientas online y extensiones VS Code

5. Conversión de datos

Ruta: `api_contacts.js`

- Conversión entre formatos (JSON para API)

```
let datos = JSON.parse(fs.readFileSync("data/contacts.json"));
```

6. Aplicación de gestión empresarial

- CRM básico: controla contactos y ventas

Proyecto Intermodular

1. Objetivo

- Gestionar contactos y oportunidades comerciales

2. Necesidad o problema

- Centraliza y automatiza la información comercial

3. Stack tecnológico

- Node.js, JavaScript, HTML, CSS, JSON
- Sencillez y amplia documentación

4. Desarrollo por versiones

- v1: gestión básica de contactos
- v2+: embudos, reportes, validaciones, etc.