



MANUAL UNIFICADO DE 5 AGENTES



SISTEMA COMPLETO EN UN SOLO ARCHIVO

Este es el ÚNICO archivo que necesitas para el sistema de 5 agentes.



CÓMO FUNCIONA EL SISTEMA

Comando Simple:

```
# Asignación automática de agentes según la tarea
node scripts/enhanced_agent1_coordinator_fixed.cjs
```

Ejemplos de Asignación Automática:

"Modificar HTML" → AGENT-2 (principal) + AGENT-4 (apoyo) + AGENT-1 (supervisor) + AGENT-5 (validador)

"Optimizar performance" → AGENT-3 (principal) + AGENT-2 (apoyo) + AGENT-1 (supervisor) + AGENT-5 (validador)




"Cambiar CSS" → AGENT-5 (principal) + AGENT-2 (apoyo) + AGENT-1 (supervisor) + AGENT-4 (validador)



AGENTE 1 - COORDINADOR MAESTRO

ROL: Supervisión y coordinación general

PERMISOS:

-  **Leer:** TODOS los archivos
-  **Escribir:** scripts/*, reportes, instrucciones
-  **Eliminar:** Ningún archivo del proyecto

TAREAS:

- Asignar agentes automáticamente según la tarea
- Verificar que no hay duplicados (0 tolerancia)
- Coordinar trabajo entre agentes
- Generar reportes finales

CONVENCIONES:

```
// Funciones
function agent1VerifyProject() { ... }
function agent1CoordinateTask() { ... }



// Variables
const AGENT1_CONFIG = { ... };
const AGENT1_RESULTS = { ... };
```



AGENTE 2 - APLICACIÓN PRINCIPAL

ROL: Archivo principal y página index

PERMISOS:

-  **Leer:** index.html, flashcard-app-final.js
-  **Escribir:** index.html, flashcard-app-final.js

- **✗ Prohibido:** services/, utils/, tests/*

TAREAS:

- Modificar estructura HTML principal
- Gestionar lógica JavaScript principal
- Integrar con servicios de otros agentes
- Manejar eventos principales de la aplicación

CONVENCIONES:

```
// Funciones
function agent2InitApp() { ... }
function agent2HandleMainEvents() { ... }

// Variables
const AGENT2_CONFIG = { ... };
let agent2CurrentState = null;

// IDs HTML
<div id="agent2-main-container">
<button id="agent2-submit-btn">
```



AGENTE 3 - GESTIÓN DE DATOS

ROL: Lógica de datos y backend

PERMISOS:

- **✓ Leer:** flashcard-app-final.js (secciones de datos)
- **✓ Escribir:** flashcard-app-final.js (solo funciones de datos)
- **✗ Prohibido:** HTML, CSS, servicios UI

TAREAS:

- Optimizar funciones de datos
- Gestionar almacenamiento local

- Implementar algoritmos de estudio
- Manejar sincronización de datos

CONVENCIONES:




```
// Funciones
function agent3LoadData() { ... }
function agent3SaveData() { ... }
function agent3OptimizeStorage() { ... }

// Variables
const AGENT3_DATA_CONFIG = { ... };
let agent3DataCache = new Map();
```

AGENTE 4 - UI Y NAVEGACIÓN

ROL: Interfaz de usuario y navegación

PERMISOS:

-  **Leer:** services/NavigationService.js, index.html (UI)
-  **Escribir:** services/NavigationService.js, index.html (solo UI)
-  **Prohibido:** Lógica de datos, CSS principal

TAREAS:

- Gestionar navegación entre páginas
- Optimizar experiencia de usuario
- Manejar componentes UI interactivos
- Verificar accesibilidad

CONVENCIONES:

```
// Funciones
function agent4Navigate() { ... }
function agent4UpdateUI() { ... }
function agent4HandleNavigation() { ... }




// Variables
const AGENT4_UI_CONFIG = { ... };
let agent4CurrentRoute = '/';
```



AGENTE 5 - UTILIDADES Y TESTING

ROL: Utilidades, estilos y testing

PERMISOS:

-  **Leer:** utils/, tests/, styles.css
-  **Escribir:** utils/, tests/, styles.css
-  **Eliminar:** tests/* (archivos obsoletos)

TAREAS:

- Gestionar utilidades y helpers
- Mantener estilos CSS
- Ejecutar y mantener tests
- Validar funcionalidad general

CONVENCIONES:

```
// Funciones
function agent5ValidateForm() { ... }
function agent5RunTests() { ... }
function agent5UpdateStyles() { ... }

// Variables
const AGENT5_UTILS_CONFIG = { ... };
let agent5TestResults = [];
```

COMUNICACIÓN ENTRE AGENTES

Sistema de Eventos:

```
// Enviar comunicación
window.dispatchEvent(new CustomEvent('agent-communication', {
  detail: {
    from: 'AGENT-X',
    to: 'AGENT-Y',
    action: 'ACTION_NAME',
    data: { ... }
  }
}));

// Escuchar comunicaciones
window.addEventListener('agent-communication', (event) => {
  if (event.detail.to === 'AGENT-X') {
    // Procesar comunicación
  }
});
```





Registro de API:

```
// Registrar funciones
window.AGENT_API = window.AGENT_API || {};
window.AGENT_API['AGENT-X'] = {
  functionName: functionReference
};

// Usar funciones de otros agentes
if (window.AGENT_API && window.AGENT_API['AGENT-Y']) {
  window.AGENT_API['AGENT-Y'].functionName(params);
}
```

PROTOCOLO DE TRABAJO UNIFICADO

ANTES DE CUALQUIER MODIFICACIÓN:

1.  Verificar que la función NO existe ya
2.  Comprobar permisos para el archivo
3.  Revisar si puedes reutilizar código existente
4.  Confirmar que no duplicas funcionalidad

DURANTE LA MODIFICACIÓN:

- 1. ☒ Usar nombres únicos con prefijo del agente
- 2. ☒ Mantener comunicación con otros agentes
- 3. ☒ Documentar cambios importantes
- 4. ☒ Probar funcionalidad inmediatamente

DESPUÉS DE LA MODIFICACIÓN:

- 1. ☒ Verificar que no se crearon duplicados
- 2. ☒ Validar que las referencias funcionan
- 3. ☒ Confirmar comunicación entre archivos
- 4. ☒ Reportar al Agente 1 (Supervisor)





MATRIZ DE ASIGNACIÓN AUTOMÁTICA

Tarea	Principal	Apoyo	Validador
Modificar HTML	AGENT-2	AGENT-4	AGENT-5
Modificar JS	AGENT-2	AGENT-3	AGENT-5
Modificar CSS	AGENT-5	AGENT-2	AGENT-4
Servicios UI	AGENT-4	AGENT-2	AGENT-5
Optimizar datos	AGENT-3	AGENT-2	AGENT-5
Testing	AGENT-5	AGENT-2	AGENT-1
Performance	AGENT-3	AGENT-2, AGENT-5	AGENT-4







REGLAS CRÍTICAS





CERO DUPLICACIONES:

-  NO crear funciones con nombres similares
-  NO duplicar configuraciones
-  NO crear archivos redundantes
-  SIEMPRE reutilizar código existente

INDEPENDENCIA DE AGENTES:

-  NO modificar archivos de otros agentes
-  NO interferir con trabajo en progreso
-  USAR comunicación para coordinarse
-  REPORTAR al Agente 1 si hay conflictos

COMUNICACIÓN OBLIGATORIA:

-  REGISTRAR tus funciones en AGENT_API
 -  USAR eventos para comunicación
 -  NOTIFICAR cambios importantes
 -  COORDINAR con agentes relacionados
-



COMANDOS PRINCIPALES

Verificación General:

```
node scripts/enhanced_agent1_coordinator_fixed.cjs
```


Limpieza Automática (NUEVO):

```
# Ejecutar limpieza automática de código obsoleto
node scripts/auto_cleanup_system.cjs

# Verificación con limpieza automática incluida
node -e "
const { EnhancedAgent1Coordinator } =
require('./scripts/enhanced_agent1_coordinator_fixed.cjs');
const coordinator = new EnhancedAgent1Coordinator();
coordinator.verifyProjectWithCleanup();
"
```

Asignación de Tarea:






```
const coordinator = new EnhancedAgent1Coordinator();
const assignment = coordinator.assignAgentsForTask("descripción de la tarea",
["archivos"]);
```


Verificar Estado:

```
git status
git log --oneline -5
```

MÉTRICAS DE ÉXITO

El sistema funciona correctamente cuando:

-  **0 duplicados** en todo el proyecto
-  **100% comunicación** entre archivos funciona
-  **Todos los agentes** trabajan independientemente
-  **Asignación automática** funciona perfectamente
-  **Calidad garantizada** en todas las modificaciones

 **ESTE ES EL ÚNICO ARCHIVO QUE NECESITAS.** Todo el sistema de 5 agentes está documentado aquí de manera unificada y sin duplicaciones.