Universidad de las Fuerzas Armadas
ESPE

Maestría en Ingeniería en Software

Impacto de las pruebas automatizadas con inteligencia artificial en la calidad del desarrollo de aplicaciones web Comparativa entre SeleniumBase, Testim y Mabl

AUTORA: Gallardo Cueva, Adela Sofia

Director: Mgtr: Montaluisa Pilatasig, Edgar Fabián



El retesting manual consume tiempo y no garantiza mejores resultados

Regresión creciente

Cada sprint obliga a repetir más casos y puede retrasar la entrega.

Localizadores frágiles

Cambios en el DOM o en un id rompen scripts aunque la función siga disponible.

Oráculos incompletos

Una ejecución verde no prueba que la regla de negocio haya sido verificada.

Decisión sin evidencia

Sin una matriz común, las herramientas se eligen por demostraciones o percepción.

Pregunta central

¿Cómo cambia la calidad del desarrollo web al comparar SeleniumBase, Testim y Mabl bajo los mismos casos, defectos y métricas?

La investigación compara impacto, no solo características

Objetivo general

Evaluar el impacto de las pruebas automatizadas con inteligencia artificial en la calidad del desarrollo de aplicaciones web mediante la comparación entre SeleniumBase, Testim y Mabl.

01 Caracterizar funcionalidades, ventajas y limitaciones de las tres herramientas.

02 Evaluar cobertura funcional, detección de defectos y eficiencia del ciclo sobre MediAgenda.

03 Analizar la percepción de efectividad, facilidad de uso y beneficio de las herramientas.

La hipótesis exige mejora simultánea en cobertura, defectos y tiempo

Hipótesis

La implementación de SeleniumBase, Testim y Mabl mejora la calidad al incrementar cobertura, reducir errores y optimizar el tiempo del ciclo.

Variable independiente

Condición de ejecución

- Manual
- SeleniumBase
- Testim
- Mabl

Variables dependientes

- Cobertura funcional
- Defectos detectados
- Tiempo de ciclo
- **Calidad y mantenibilidad**

La IA ayuda con el locator, pero no sustituye el oráculo

ISO/IEC 25010

Idoneidad funcional, fiabilidad y mantenibilidad orientan la lectura de calidad.

ISTQB / ISO 29119

Planificar, diseñar, ejecutar y reportar con resultados esperados explícitos.

Automatización

Reduce repetición, pero requiere datos, aserciones y mantenimiento.

Self-healing

Puede adaptar localizadores; también puede elegir un control incorrecto.

Oráculo de prueba

Determina si la salida es realmente correcta y exige criterio de negocio.

Supervisión humana

Necesaria para revisar falsos positivos y reparaciones automáticas.

Un caso único de laboratorio permite comparar cuatro condiciones

Enfoque
Mixto, con mayor peso cuantitativo

Alcance
Descriptivo y comparativo

Diseño
Experimental sobre el artefacto; caso único

Unidad
MediAgenda: autenticación, reserva, consulta y cancelación

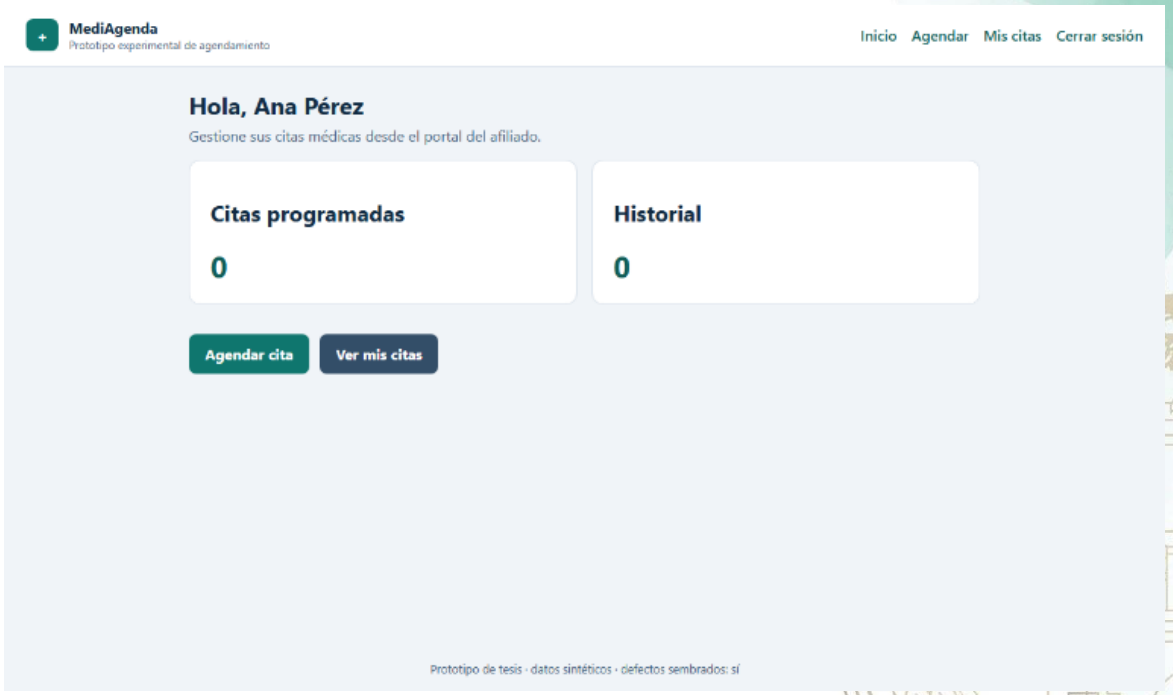
Evidencia
18 casos, 6 defectos, tiempos, mantenimiento y Likert

Percepción
8 perfiles sintéticos: 3 desarrollo, 3 QA y 2 leads

Testim y Mabl se modelaron por no disponer de licencias operativas; SeleniumBase y la ejecución manual se aplicaron sobre el prototipo.

MediAgenda concentra reglas críticas en un flujo sencillo

1. Login
2. Panel
3. Agendar
4. Confirmar
5. Mis citas
6. Cancelar / salir



Objeto medido: 18 casos funcionales + 6 defectos sembrados

La matriz cubre autenticación y reglas de agendamiento

ID	Módulo	Escenario	Tipo
CP-01	AUTH	Login válido	Positivo
CP-02	AUTH	Clave incorrecta	Negativo
CP-03	AUTH	Correo inexistente	Negativo
CP-04	AUTH	Clave vacía	Negativo
CP-05	AGENDA	Agendar cita	Positivo
CP-06	AGENDA	Sin especialidad	Negativo
CP-07	AGENDA	Médico vacío	Negativo
CP-08	AGENDA	Hora transcurrida	Negativo
CP-09	AGENDA	Horario ocupado	Negativo

La matriz también prueba oráculos, cancelación, sesión y UI

ID	Módulo	Escenario	Tipo
CP-10	AGENDA	Código confirmación	Positivo
CP-11	CONSULTA	Listar citas	Positivo
CP-12	CANCELAR	Cancelar cita	Positivo
CP-13	CANCELAR	Reagendar cupo	Positivo
CP-14	AGENDA	Doble cita	Negativo
CP-15	SESIÓN	Cerrar sesión	Positivo
CP-16	SESIÓN	Ruta sin login	Negativo
CP-17	AGENDA	Cambio especialidad	Positivo
CP-18	UI	Controles visibles	Positivo

Se sembraron seis fallos de negocio y seguridad

DEF-01 **Clave vacía aceptada**

Caso detector: CP-04

DEF-02 **Hora de hoy ya transcurrida**

Caso detector: CP-08

DEF-03 **Doble cita a la misma hora**

Caso detector: CP-14

DEF-04 **Código de pantalla ≠ lista**

Caso detector: CP-10

DEF-05 **Cancelar no libera cupo**

Caso detector: CP-12/13

DEF-06 **/citas accesible sin sesión**

Caso detector: CP-16

Las herramientas intercambian control, facilidad y transparencia

Criterio	SeleniumBase	Testim	Mabl
Base	Python + WebDriver	Low-code SaaS	Low-code/codeless SaaS
Creación	pytest / código	Grabación y ajuste	Flujo en nube
IA / robustez	Depende del programador	Locator multiatributo + self-healing	Self-healing + prueba visual
Licencia	Código abierto	Pago	Pago
Origen en el estudio	Piloto-modelo	Modelo	Modelo

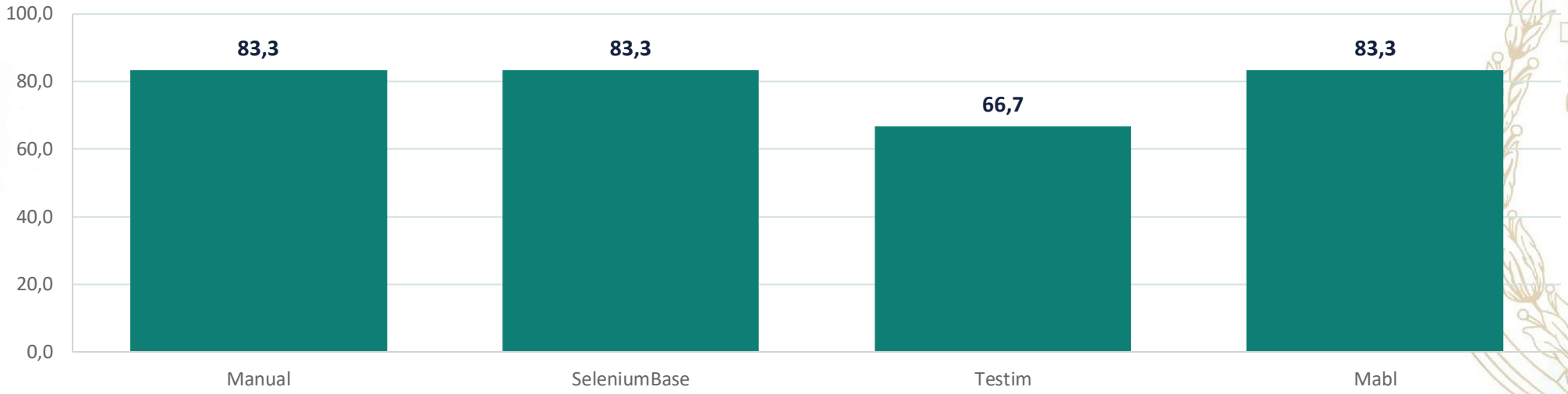
La cobertura fue idéntica; la detección no mejoró de forma uniforme

100 %
Cobertura manual

100 %
SeleniumBase

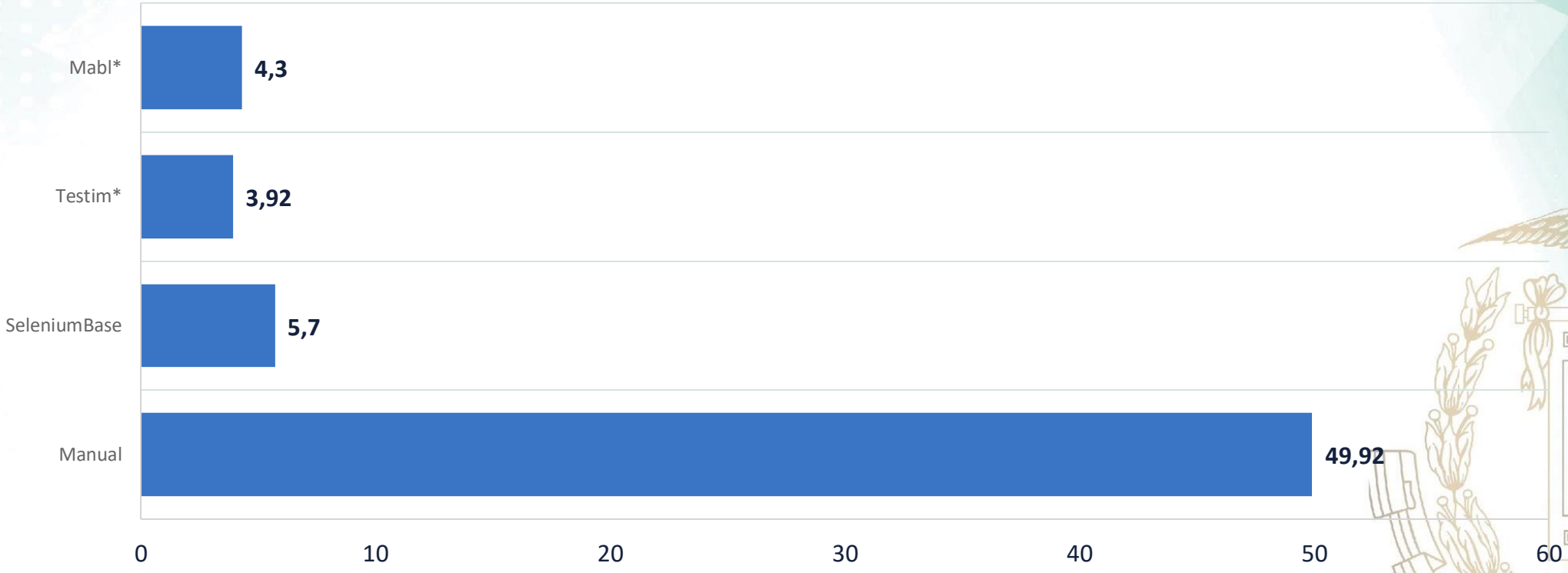
100 %
Testim

100 %
Mabl



TIEMPO TOTAL DE EJECUCIÓN

La automatización redujo el ciclo de casi 50 a menos de 6 minutos



* Testim y Mabl: tiempos modelados por ausencia de licencia operativa.

Reducción frente a manual: 88,6 % SeleniumBase · 92,1 % Testim · 91,4 % Mabl

El conteo empata, pero no se escapan los mismos defectos

Condición	Detectados	No detectado relevante	Falsos positivos
Manual	**5/6 (83,3 %)**	DEF-04: código inconsistente	0
SeleniumBase	**5/6 (83,3 %)**	DEF-02: hora transcurrida	0
Testim*	**4/6 (66,7 %)**	DEF-03: doble cita	1 (CP-17)
Mabl*	**5/6 (83,3 %)**	DEF-03: doble cita	1 (CP-18)

El defecto que queda sin identificar importa más que el empate porcentual.

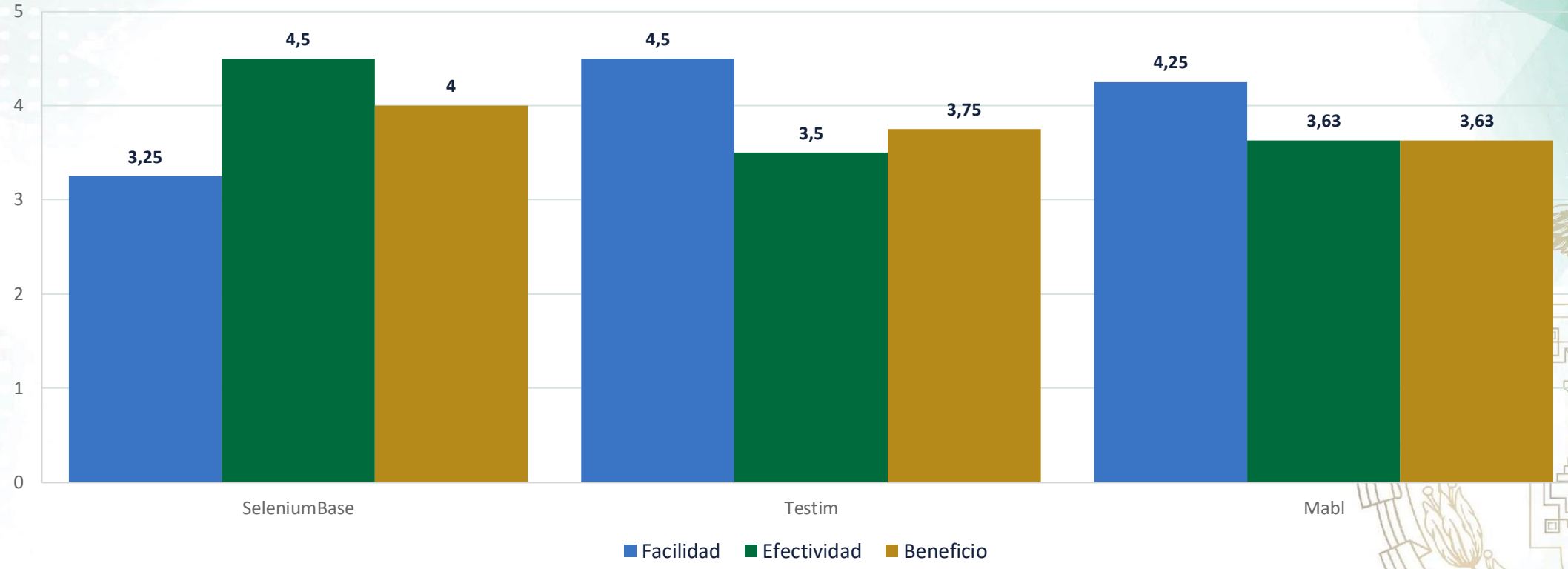
El self-healing redujo casos rotos, a cambio de revisión humana

Condición	Casos impactados	Rotos	Autocurados
Manual	**0**	**0**	**0**
SeleniumBase	**3**	**3**	**0**
Testim*	**3**	**1**	**2**
Mabl*	**3**	**1**	**2**

SeleniumBase: reparación explícita y auditable

Testim / Mabl: menos roturas, menor transparencia

La facilidad percibida no equivale a mayor efectividad



Ocho perfiles sintéticos del equipo del prototipo; no representan empleados de la empresa patrocinadora.

La hipótesis se respalda en tiempo, no en cobertura ni detección uniforme

Tiempo

Sí se respalda

49,92 min → menos de 6 min

Cobertura

No se respalda

18/18 en las cuatro condiciones

Detección

No es uniforme

5/6 manual, SeleniumBase y Mabl; 4/6 Testim

Ninguna condición resuelve por sí sola velocidad, oráculo y mantenimiento.

Automatizar acelera el retesting, pero no garantiza más calidad

01 La cobertura funcional permaneció en 100 %: automatizar no amplió la matriz.

02 SeleniumBase y Mabl igualaron la detección manual (83,3 %); Testim alcanzó 66,7 %.

03 La principal ventaja fue temporal: todos los escenarios automatizados quedaron bajo 6 minutos.

04 El tipo de defecto omitido y los falsos positivos cambian la interpretación del porcentaje.

05 Self-healing reduce roturas por cambios de id, pero exige revisar qué elemento fue elegido.

06 La calidad depende de casos bien diseñados, oráculos explícitos y supervisión humana.

La estrategia debe comenzar por la matriz y el oráculo

01

Diseñar primero

Mantener 18 casos con resultados esperados y reglas de negocio explícitas.

02

Si hay capacidad Python

Usar SeleniumBase para retesting versionable y localizadores más robustos que un id único.

03

Si prima la facilidad

Evaluar Testim o Mabl, pero revisar toda autocuración y mantener aserciones de negocio.

04

Medir el costo completo

Registrar juntos tiempo de ejecución y casos rotos después de cada cambio de interfaz.

05

Completar la validación

Validar formalmente el cuestionario antes de presentarlo como instrumento aprobado.

06

Repetir en contexto real

Ejecutar sobre el portal real y con licencias operativas antes de generalizar resultados.

Resultado central

La IA aplicada al testing ayuda a mantener y repetir pruebas; no reemplaza el criterio con el que se decide si el sistema funciona correctamente.

Gracias por su atención

Preguntas