

Group 4. Presentation

Week 1 & 2

Pregunta S1

La evidencia de pruebas que reduce incertidumbre sobre la calidad de software se refiere a artefactos, datos o documentación producidos durante las actividades de testing que permiten evaluar objetivamente qué tan bien el software satisface sus requisitos y comportamientos esperados. Esta evidencia ayuda a tomar decisiones fundamentadas sobre la preparación del producto para su uso o liberación, sin confundirla con las actividades más amplias de Quality Assurance (QA), que incluyen planificación, procesos y mejoras sistémicas.

Evidencia S1

Filter... x

Auth

CreateToken

Booking

GetBookingIds

GetBooking

CreateBooking

UpdateBooking

PartialUpdateBooking

DeleteBooking

Ping

HealthCheck

restful-booker

1.0.0

API documentation for the playground API restful-booker. [Click here to go back to Home](#)

Auth

Auth - CreateToken

1.0.0

Creates a new auth token to use for access to the PUT and DELETE /booking

POST

https://restful-booker.herokuapp.com/auth

Example 1:

curl -X POST \n https://restful-booker.herokuapp.com/auth \n -H 'Content-Type: application/json' \n -d '{\n "username": "admin",\n "password": "password123"\n }'

Header

Campo	Tipo	Descripción
Content-Type	string	Sets the format of payload you are sending Valor por defecto: application/json

Request body

Campo	Tipo	Descripción
username	String	Username for authentication

```
=====
Restful-Booker CreateBooking Test
URL básica: http://localhost:3001
Marca de Tiempo: 2026-01-21 12:22:55
Archivo de Registro: evidence/week2/createBooking 20260121_122255.log
=====

-----Inicia a ejecutar el test: CreateBooking-----
✓ PASSED: CreateBooking creado exitosamente con el ID 62
✓ PASSED: CreateBooking creado exitosamente con datos {"firstname":"Grupo-4","lastname":"Grupo-4","totalprice":112,"depo

=====
Resumen de Pruebas
=====
Pasaron: 1
Fallaron: 0

El test pasó!

La evidencia se guardó en: evidence/week2/createBooking 20260121_122255.log
```

```
=====
PASSED: autentificacion exitosa de token
=====
Restful-Booker Smoke Tests
URL básica: http://localhost:3001
Marca de Tiempo: 2026-01-21 12:22:51
Archivo de Registro: evidence/week2/smoke 20260121_122251.log
=====

-----Test 1: CreateBooking-----
✓ PASSED: CreateBooking creó la reserva exitosamente

-----Test 2: GetBookingIds-----
✓ PASSED: GetBookingIds realizado exitosamente

-----Test 3: GetBooking por Id-----
✓ PASSED: GetBooking devolvió el ID de reserva exitosamente

-----Test 4: UpdateBooking-----
✓ PASSED: UpdateBooking actualizó exitosamente

-----Test 5: PartialUpdateBooking-----
✓ PASSED: PartialUpdateBooking actualizó exitosamente

-----Test 6: DeleteBooking-----
✓ PASSED: DeleteBooking eliminó exitosamente la reserva

=====
Resumen de Resultados de Pruebas
=====
Pasaron: 7
Fallaron: 0
Total: 7

Create Jira Issue
Todos los tests pasaron!

La evidencia se guardó en: evidence/week2/smoke 20260121_122251.log
```

¿CÓMO CONVERTIR “CALIDAD” EN AFIRMACIONES FALSABLES Y MEDIBLES?

Convertir “calidad” en afirmaciones falsables y medibles consiste en formular escenarios concretos con métricas y umbrales claros, respaldados por evidencia reproducible y con límites de validez explícitos. De este modo, la calidad deja de ser una opinión subjetiva y se convierte en una hipótesis técnica evaluable, coherente con el vocabulario del ISTQB y con un enfoque de ingeniería orientado a la reducción de riesgo, tal como proponen arc42, ATAM y Software Engineering at Google.

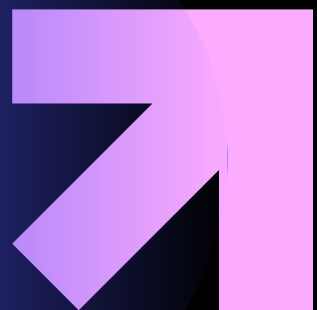
Escenario A — Operaciones CRUD básicas (Disponibilidad funcional) - Falsación: si alguna operación retorna error (5xx) o respuesta vacía
Escenario B — Robustez ante ID inválido en GET - Falsación: si devuelve HTTP 200 para ID inexistente, el escenario queda refutado

Claim	Escenario	Métrica	Evidencia (archivo)	Oráculo (pass/fail)
SUT accesible	Q1 Disponibilidad API	Captura de pantalla + interfaz web	evidence/week1/screenshot-*.png	pass si acceso confirmado visualmente
Operaciones CRUD funcionan	Q2 Smoke tests (Create, Read, Update, Delete)	http_code en logs	evidence/week2/smoke_*.log, createBooking_*.log, updateBooking_*.log, deleteBooking_*.log	pass si todos HTTP 200/201/204
GET con ID inválido no devuelve 200	Q3 Robustez entrada (ID inexistente)	http_code en log	evidence/week2/getBookingById_*.log	pass si http_code != 200 para ID inválido

MÉTODO FORMALIZADO

Slide 03

- 1) Definimos claims acotados sobre Restful Booker en entorno local:
 - SUT accesible y funcional (captura visual + logs de respuesta)
 - Operaciones CRUD básicas operativas (smoke tests)
 - Robustez ante entradas inválidas (GET con ID inexistente)
- 2) Los expresamos como escenarios verificables (Estímulo–Entorno–Respuesta–Medida) documentados en `quality/scenarios.md`.
- 3) Definimos oráculos mínimos:
 - Disponibilidad: interfaz web accesible (capturas en week1)
 - CRUD: HTTP 200/201/204 + JSON válido (logs en week2).
 - Robustez: HTTP \neq 200 para IDs inválidos (log getBookingById)
- 4) Generamos evidencia reproducible con scripts shell (`scripts/*.sh`) y la versionamos:
 - `evidence/week1/`: capturas de pantalla (accesibilidad del SUT)
 - `evidence/week2/`: logs de ejecución (*.log con timestamps y códigos HTTP)



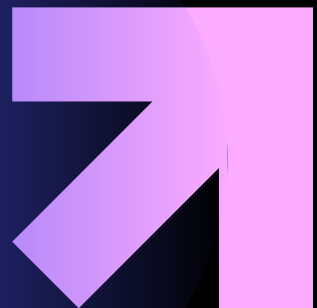
Fuentes

- ISTQB: oráculo y objetivos de prueba (criterio de pass/fail).
- SWE@Google: testing como reducción de riesgo y claridad de pruebas.
- arc42/ATAM: escenarios medibles y verificables (no adjetivos).

VALIDEZ (AMENAZAS Y LÍMITES)

Slide 04

- Interna:
 - Ejecución de scripts depende del estado del SUT (puerto 3001, Docker activo).
 - Mitigación: verificar startup con `setup/healthcheck_sut.sh` antes de pruebas y documentar estado inicial en logs.
 - Variabilidad en timestamps de ejecución → registrar fecha/hora exacta en cada log.
- Constructo:
 - CRUD local no prueba escalabilidad, concurrencia ni seguridad en producción.
 - Mitigación: acotar claims a "disponibilidad funcional local" y operaciones CRUD básicas, sin afirmar calidad para carga real.
 - Oráculos simples (HTTP code) no detectan lógica de negocio errónea.
- Externa:
 - Resultados en máquina local pueden no reproducirse en otros entornos/configuraciones Docker.
 - Mitigación: documentar versiones (Docker, Node.js, puerto), repetir ejecución de scripts (`scripts/*.sh`) múltiples veces y comparar logs.
 - Capturas de pantalla (`evidence/week1/`) reflejan un instante; no garantizan estabilidad continua



CONCLUSIONES

Evidencia más fuerte: disponibilidad funcional de CRUD en Restful Booker.

- Smoke tests (`evidence/week2/smoke_*.log`) verifican que operaciones básicas (create, read, update, delete) responden con códigos HTTP válidos.
- Falsación directa: si algún script retorna 5xx, el SUT no cumple mínimo de disponibilidad.

Límite más crítico: claims locales no generalizan a producción.

- Capturas en `evidence/week1/` muestran accesibilidad de la interfaz en un instante específico.
- Logs en `evidence/week2/` documentan ejecución única sin garantizar estabilidad en carga o multi-usuario.
- No afirmamos robustez ante ataques, concurrencia ni escalabilidad.

Mejora concreta (sin implementar hoy)

- Ejecutar scripts múltiples veces y agregar análisis de variabilidad en tiempos de respuesta.
- Documentar explícitamente configuración del SUT (versión Node.js, memoria Docker, etc.) para reproducibilidad.
- Definir umbrales de latencia (e.g., $p95 < Xms$) basados en requisitos de negocio y justificarlos.

Thank You