TP Integrador – Unidad 4: Aseguramiento de Calidad (TDD) – TESTS Node.js

15) Tests TDD para Node.js (Jest)

Este archivo contiene todos los **tests** necesarios para implementar la US *"Inscribirme a actividad"* bajo el enfoque **TDD** usando **Node.js + Jest**.

15.1. package.json

```
{
  "name": "ecoharmony-tdd",
  "version": "1.0.0",
  "scripts": {
    "test": "jest --watchAll=false"
  },
  "devDependencies": {
    "jest": "^29.7.0"
  }
}
```

15.2. Estructura del proyecto

```
/ (raíz)
├── package.json
├── src/
│   └── enrollmentService.js   // servicio (comienza vacío)
└── __tests__/
    └── enrollment.test.js     // los 6 tests TDD
```

15.3. src/enrollmentService.js

> Comienza en **ROJO** (sin implementación) — todos los tests fallarán al principio.

```
function inscribir(/* req, catalog */) {
  throw new Error('Not implemented yet');
}

module.exports = { inscribir };
```

15.4. tests/enrollment.test.js

Cada test representa un **caso CP1..CP6** del pseudocódigo del TP.

```
const { inscribir } = require('../src/enrollmentService');

const ERROR_SIN_CUPO = 'Sin cupo disponible para el horario seleccionado';
const ERROR_HORARIO  = 'Horario no disponible';
const ERROR_TYC      = 'Debe aceptar términos y condiciones';
const ERROR_TALLE    = 'Falta talle de vestimenta';

function makeCatalog() {
  const data = new Map();
  return {
    withSlot(activity, id, habilitado, cupos) {
      data.set(`${activity}:${id}`, { id, habilitado, cuposDisponibles:
cupos });
      return this;
    },
    build() { return this; },
    findSlot(activity, horarioId) { return data.get(`${activity}:${horarioId}
`) || null; },
    getSlot(activity, horarioId) { return data.get(`${activity}:${horarioId}
`) || null; },
    setCupos(activity, horarioId, cupos) {
      const slot = data.get(`${activity}:${horarioId}`);
      if (slot) slot.cuposDisponibles = cupos;
    }
  };
}

function defaultCatalog() {
  return makeCatalog()
    .withSlot('TIROLESA', 'H1', true, 5)
    .withSlot('SAFARI',   'H2', true, 10)
    .withSlot('PALESTRA', 'H3', false, 8)
    .build();
}

function p(nombre, dni, edad, talle) {
  const obj = { nombre, dni, edad };
  if (talle !== undefined) obj.talle = talle;
  return obj;
}

function req({ actividad, horarioId, participantes, terminosAceptados =
true }) {
  return { actividad, horarioId, participantes, terminosAceptados };
}
```

```
describe('US: Inscribirme a actividad – TDD (Jest)', () => {
  let catalog;

  beforeEach(() => {
    catalog = defaultCatalog();
  });

  test('CP1: éxito con datos válidos', () => {
    const solicitud = req({
      actividad: 'TIROLESA', horarioId: 'H1',
      participantes: [ p('Cami', '123', 20, 'M') ], terminosAceptados: true
    });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(true);
    expect(typeof res.codigoReserva).toBe('string');
    expect(res.errores).toEqual([]);
    expect(catalog.getSlot('TIROLESA', 'H1').cuposDisponibles).toBe(4);
  });

  test('CP2: sin cupo disponible', () => {
    catalog.setCupos('TIROLESA', 'H1', 0);
    const solicitud = req({ actividad: 'TIROLESA', horarioId: 'H1',
participantes: [ p('Cami', '123', 20, 'M') ] });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(false);
    expect(res.errores).toContain(ERROR_SIN_CUPO);
  });

  test('CP3: sin talle si la actividad no lo requiere', () => {
    const solicitud = req({ actividad: 'SAFARI', horarioId: 'H2',
participantes: [ p('Alex', '1111', 25) ] });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(true);
  });

  test('CP4: horario no disponible', () => {
    const solicitud = req({ actividad: 'PALESTRA', horarioId: 'H3',
participantes: [ p('Cami', '123', 20, 'S') ] });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(false);
    expect(res.errores).toContain(ERROR_HORARIO);
  });

  test('CP5: no acepta términos y condiciones', () => {
    const solicitud = req({ actividad: 'TIROLESA', horarioId: 'H1',
participantes: [ p('Cami', '123', 20, 'M') ], terminosAceptados: false });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(false);
    expect(res.errores).toContain(ERROR_TYC);
  });
```

```
  test('CP6: falta talle en actividad que lo requiere', () => {
    const solicitud = req({ actividad: 'TIROLESA', horarioId: 'H1',
participantes: [ p('Cami', '123', 20) ], terminosAceptados: true });
    const res = inscribir(solicitud, catalog);
    expect(res.ok).toBe(false);
    expect(res.errores).toContain(ERROR_TALLE);
  });
});
```

15.5. Cómo usar los tests (paso a paso TDD)

1. **RED:** correr `npm test` con la función vacía → todos los tests fallan (correcto).
2. **GREEN:** agregar la lógica mínima en `inscribir()` para que pase el primer test (CP1).
3. **REFACTOR:** limpiar duplicaciones, crear helpers (por ejemplo `tieneCupo`, `reservar`, etc.).
4. Repetir el ciclo para CP2..CP6 hasta que **toda la suite esté en verde**.

   Así obtenés el mismo flujo Red → Green → Refactor explicado en tu pseudocódigo, pero aplicado al entorno **Node.js + Jest**.