Clase	Método	Primer Parámetro	Segundo Parámetro	Resultado	Resultado obtenido
Payments	getPaymentHistoryByUserId_ok	new User(id=1,email=ana@exa.co), new List<Entities>	N/A		List<Entities>
Payments	getPaymentNumber_ok	startDate = 01/01/2025, endDate = 10/01/2025	N/A		expectedNumber = 5
Payments	earningsByCat_ok	category = "SIMIT"	N/A		expectedEarnings = 500000
Payments	getNetWorth_ok	N/A	N/A		expectedNetworth = 2000000
Payments	savedUsrMoney_ok	N/A	N/A		expectedSavings = 300000
Payments	graduatedNum_ok	N/A	N/A		expectedGraduates = 10
Usr	registUser_success	newUser(email=new@exa.co, id=CC123, pass=plain)	N/A	Retorna el Usr no nulo	Retorna Usr no nulo con password = "encoded-pass"; se verifican llamadas: findByEmail, findByIdentification, encode, save
Usr	login_success	email=login@exa.co	password=secret	Retorna el Usr no nulo	Retorna Usr no nulo con email = "login@exa.co"; se verifica usrRepository.findByEmail(email)
Usr	findByEmail_success	email=find@exa.co	N/A	Retorna el Usr no nulo	Retorna Usr no nulo con email = "find@exa.co"; se verifica usrRepository.findByEmail("find@exa.co")
Usr	deleteByEmail_success	email=del@exa.co	N/A	Retorna el Usr no nulo	Retorna Usr no nulo del email indicado; se verifican findByEmail(email) y delete(dbUser)
Usr	update_success_changeNameAndEmail NotTaken	currentEmail=old@exa.co	req: { email: "new@exa.co", name: "New Name" }	Retorna el Usr	Retorna Usr con name = "New Name" y email = "new@exa.co"; se verifican findByEmail(current), findByEmail(new) y save(existing)
Usr	changePassword_success	req.email=cp@exa.co	req.password=newSecret	Retorna el Usr	Retorna Usr; el password persistido matchea BCrypt("newSecret"); se verifica save(...)
Usr	updatePassword_success	email=up@exa.co	current=currPwd, next=nextPwd	Retorna el Usr	Retorna Usr; el password persistido matchea BCrypt("nextPwd"); se verifica save(...)
Vehicle	saveVehicle_success_newPlate_assigns User_and_UppercasesPlate_andPersists	email=owner@exa.co	req=Vehicle(plate="abc123")		Retorna Vehicle con plate="ABC123" y usr asignado al dueño; se verifican findByPlate("abc123"), usrService.findByEmail(email) y save(..)
Vehicle	getByPlate_success	plate="XYZ999"	N/A		Retorna Vehicle con plate="XYZ999"; se verifica vehicleRepository.findByPlate("XYZ999")
Vehicle	getVehicles_success_returnsUsersVehicl es	email=fleet@exa.co	N/A		Retorna List<Vehicle> de tamaño 2 con placas AAA111, BBB222; se verifica usrService.findByEmail (email) y no hay interacciones con vehicleRepository
Vehicle	deleteByPlate_success_findsByOwnerAn dPlate_deletesAndReturnsVehicle	email=owner@exa.co	plate="CCC333"		Retorna Vehicle con plate="CCC333"; se verifican findByUsr_EmailAndPlate(email, plate) y delete(v)
ExpirationReminder	sendMonthlyExpiringReminders_sendsO neEmail_perUser_andFormatsBody	0	N/A	Envío del correo	Consulta vehicleRepository.findVehiclesExpiringBy(hoy + 1 mes); envía 1 correo a ana@exa.co con asunto "Recordatorio: vencimientos próximos de SOAT/Tecnomecánica". El cuerpo: saluda por nombre ("Hola Ana"), lista placas ABC123 y XYZ987, incluye fecha formateada yyyy-MM-dd (SOAT ABC123), muestra "día(s)", el encabezado "Tarifa SOAT" y la firma "Equipo SmartTraffic"
ExpirationReminder	sendMonthlyExpiringReminders_groupsB yUser_sendsTwoEmails	0	N/A	Envío de correos	Agrupa por usuario y envía 2 correos: uno a ana@exa.co (contiene AAA111) y otro a luis@exa.co (contiene BBB222). Verifica exactamente 2 invocaciones a emailService.enviarCorreo(...)
ExpirationReminder Job	runDailyReminder_callsService	0	N/A	Envío del correo	Invoca exactamente 1 vez ExpirationReminderService.sendMonthlyExpiringReminders(); se verifica con verify(svc, times(1))...

Usr	getNearestPartner_success	email = "exatest@gmail.com"
		type = "SOAT"
		maxDistanceSimulated = 10280.0
			N/A	Retorna la Lista de partners	Retorna la lista de Partners que prestan el servicio
					indicado por el usuario, ordenados por cercanía en
					el radio indicado por el usuario
					Ejemplo del objeto retornado:
					List<Partner> outPartners = {
					(3, "Partner C", 4.685, -74.118),
					(1, "Partner A", 4.6838, -74.13)
					}

		Usr	getPartnersByTypeServicesNR_success	type = "SOAT"	N/A	Retorna el mapa de partners	Retorna el mapa de Partners que prestan el servicio indicado por el usuario, sin duplicar Partners por verificación de ID y sin objetos nulos Ejemplo del objeto retornado: Map<Integer, Partner> outMap = 1, (1, "Partner A", 4.6838, -74.13), 2, (2, "Partner B", 4.6862, -74.119), 3, (3, "Partner C", 4.685, -74.118)
		Usr	getAllCourses_success	email = "exatest@gmail.com"	N/A	Retorna lista de cursos disponibles	Retorna List<CoursesData> con dos elementos (Curso A, Curso B); se verifican llamadas a usrRepository.findByEmail(email) y coursesDataRepository.findAvailableCourses()
		Usr	getCoursesByUser_success	email = "exatest@gmail.com"	N/A	Retorna lista de cursos asociados al usuario	Retorna List<CoursesData> con dos elementos (Curso A, Curso B); se verifican llamadas a usrRepository.findByEmail(email) y coursesDataRepository.getCoursesByUser(user.id)
		Usr	registerUserToCourse_success	email = "exatest@gmail.com"	course = CoursesData(id=1, name="Curso A", capacity=20, parcialCapacity=0)	Registra el usuario en el curso y aumenta la capacidad parcial	Retorna el objeto CoursesData actualizado con parcialCapacity = 1; el usuario queda inscrito (user.getCourses() contiene el curso); se verifican usrRepository.findByEmail(email), coursesDataRepository.findById(course.id) y usrRepository.save(user)
		Usr	deleteUserFromCourse_success	email = "exatest@gmail.com"	course = CoursesData(id=1, name="Curso A", capacity=20, parcialCapacity=1)	Elimina al usuario del curso y reduce la capacidad parcial	Retorna el objeto CoursesData actualizado con parcialCapacity = 0; el curso ya no está en user.getCourses(); se verifican usrRepository.findByEmail(email), coursesDataRepository.findById(course.id) y usrRepository.save(user)
		Usr	getPartnerByService_TECNO_success	type = "TECNO"	N/A	Retorna lista de Partners que prestan servicio técnico vehicular	Retorna List<Partner> con un único objeto (3, "Partner C", 4.6850, -74.118); se verifica llamada a partnerRepository.getPartnersByTechno()
		Usr	getPartnerByService_SOAT_success	type = "SOAT"	N/A	Retorna lista de Partners que prestan servicio de SOAT	Retorna List<Partner> con dos objetos (1, "Partner A") y (2, "Partner B"); se verifica llamada a partnerRepository.getPartnersBySoat()
		Usr	getPartnerByService_default_success	type = "COURSE"	N/A	Retorna lista de Partners de tipo general (CIA)	Retorna List<Partner> con un único objeto (9, "Partner X", 4.60, -74.09); se verifica llamada a partnerRepository.getCIA()
		Service	createServices_ok	user = new user(1,"nico@exa.co"), vehicle = new vehicle("ABC123","101",2018,"Motocicleta")	N/A	Crea el Servicio	Retorna el objeto servicio service = service(id, "SOAT", "ABC123", 300000);
		Service	getServicesByUser_ok	user = new user(1,"nico@exa.co")	N/A	Obtiene los servicios de un usuario determonado	Retorna los servicios asociados a un usuario service(1, "SOAT", "ABC123", 300000), service(2, "TECNO", "XYZ789", 200000)
		Service	getSpecificServices_ok	ServiceId = 1	N/A	Obtiene un servicio en especifico	retorna el servicio service(serviceId, "SOAT", "ABC123", 300000);
		Service	deleteEspecificServices_ok	ServiceId = 1	N/A	Elimina un servicio en especifico	Retorna el objeto eliminado service(serviceId, "SOAT", "ABC123", 300000);
		Service	updateGraduated_ok	ServiceId = 1	N/A	Actualiza el estado de un curso	Retorna True
		Payments	getSpecificPayments_ok	PaymentId = 1	N/A	Obtiene un pago en especifico	Retorna el pago payment(paymentId, LocalDate.of(2023, 5, 1), 100000, "COMPLETED");
		Payments	deleteEspecificPayments_ok	PaymentId = 1	N/A	Elimina un pago en especifico	Retorna el pago eliminado payment(paymentId, LocalDate.of(2023, 5, 1), 100000, "COMPLETED")
		Payments	createCheckout_ok	serviceId = 10	N/A	Crear un pago en específico	Retorna la URL a la que redirige automáticamente la pasarela de pagos MercadoPago para ejecutar el pago del servicio. Ejemplo: https://www.mercadopago.com.co/checkout/v1/payment/redirect/4d2de502-730f-4a7b-9482-63e0facafcab/review/?preference-id=2951551977-d0e9a4b0-f991-4ab3-b3d1-9da3945ff21e&router-request-id=26a4530d-9d88-4045-8305-95c020ebc424&p=0c068e403f21405 1695df2b665e8f079
		CoursesData	createCourse_ok	email = usermail@exa.co	Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)	Crea un curso con los parámetros ingresados como atributos	Retorna automáticamente el curso que se está guardando en la base de datos. Ejemplo: Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)

		CoursesData	getUsersByCourse_ok	email = usermail@exa.co	Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)	Devuelve los usuarios que están inscritos en un curso determinado	Retorna una lista de los usuarios que están inscritos en un curso en específico
							Ejemplo: List<Usr> users= {
							(1, "Test01"),
							(2, "Test02")
							}

		CoursesData	getSpecificCourse_ok	courseId = 1	N/A	Devuelve un curso en específico	Retorna el curso según el ID que se pase como parámetro del mismo: Ejemplo: Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)
		CoursesData	deleteCourse_ok	Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)	N/A	Devuelve el curso que se eliminó	Retorna el curso que se borró de la base de datos Ejemplo: Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)
		CoursesData	updateCourse_ok	CoursesData = 1, "Curso de Conducción", "A1", 100000, 25, 0, false, false);	N/A	Devuelve el curso que fue actualizado	Retorna el curso que se acrualizó en la la base de datos Ejemplo: Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)

		CoursesData	getAllCourses_ok	N/A	N/A	Devuelve todos los cursos disponibles ofertados	Retorna una lista de los cursos que están disponibles para su inscripción
							Ejemplo: List<CoursesData> courses= {
							(1, "Curso de Conducción", "A1", 100000, 25, 0, false, false),
							(2, "Curso de Conducción", "B1", 110000, 30, 0, false, false);
							}

		CoursesData	enroll_ok_CaseNormalCapacity	userId = 1	courseId = 2	Inscribe un usuario a un curso que no ha llegado al límite de capacidad	Devuelve un mensaje de confirmación por meido de un correo al usuario informandole que fue inscrito exitosamente Ejemplo: mensaje = "Cliente inscrito a curso exitosamente"
		CoursesData	enroll_ok_CaseFirstTimeQueue	userId = 1	courseId = 2	Crea un nuevo curso de las mismas características a las que el usuario quiere inscribirse, inscribe al usuario en el mismo pero no se publica aún el curso	Devuelve internamente un null, para luego hacer un envio de correo informandole al usuario de la situación Ejemplo: mensaje = "Tu pago fue confirmado. No pudimos inscribirte al curso. De haber más interesados en este curso, te notificaremos y abriremos nuevos cupos."
		CoursesData	enroll_ok_CaseReachMinimum	userId = 1	courseId = 2	Publica el curso encolado al superar el número mínimo de interesados	Devuelve internamente un null, para luego hacer un envio de correo informandole al usuario de la situación Ejemplo: mensaje = "Curso Disponible, nos complace informarte que el curso al que te inscribiste superó el mínimo de interesados y fue abierto"
		CoursesData	unroll_ok	userId = 1	courseId = 2	Desenrola a un usuario de un curso determinado al que ya se encontraba inscrito	Devuelve un mensaje confimando el éxito de la operación Ejemplo: mensaje = Cliente desinscrito de curso exitosamente

		CoursesData	getUsersByCourseId_ok	email = usermail@exa.co	Course = (1, "Curso de Conducción", "A1", 100000, 25, 0, false, false)	Devuelve los usuarios que están inscritos en un curso determinado	Retorna una lista de los usuarios que están inscritos en un curso en específico
							Ejemplo: List<Usr> users= {
							(1, "Test01"),
							(2, "Test02")
							}

		Partner	createPartner_ok	email = "testEmail@gmail.com"	Partner = (1, "Partner1", 4.654, -74.054, true, true, services) Internamente la variable services es una lista de servicios que no se va a especificar	Crea un partner según los atributos que se ingresaron como parámetros	Devuelve el Partner creado y guaradado en la base de datos Ejemplo: Partner = (1, "Partner1", 4.654, -74.054, true, true, services)

		Partner	getAllPartners_ok	N/A	N/A	Muestra todos los partner disponibles	Devuelve una lista de todos los partner disponibles
							Ejemplo: List<Partner> partners= {
							(1, "Partner1", 4.654, -74.054, true, true, services1),
							(2, "Partner2", 4.654, -74.054, true, true, services2)
							}

		Partner	getSpecificPartner_ok	partnerId = 1	N/A	Muestra un partner en específico	Devuelve el partner en específico seleccionado Ejemplo: Partner = (1, "Partner1", 4.654, -74.054, true, true, services1)

		Partner	getPartnerByService_ok	type = SOAT	N/A	Muestra todos los partner que prestan el servicio que el usuario ingresó	Devuelve una lista de los partner que prestan el servicio proporcionado por el usuario
							Ejemplo: List<Partner> partners= {
							(1, "Partner1", 4.654, -74.054, true, true, services1),
							(2, "Partner2", 4.654, -74.054, true, true, services2)
							}

		Partner	deleteSpecificPartner_ok	partnerId = 1	N/A	Elimina un partner en específico	Devuelve el registro del Partner que se está eliminando de la base de datos Ejemplo: Partner = (1, "Partner1", 4.654, -74.054, true, true, services)
		Partner	updatePartner_ok	"Partner = (1, ""Partner1"", 4.654, -74.054, true, true, services)	N/A	Actualiza un partner con la información ingresada	Devuelve el registro del partner que se actualizó en la base de datos Ejemplo: Partner = (1, "Partner1", 4.654, -74.054, true, true, services)