{

"info": {

"\_postman\_id": "a61a9b06-83ce-4c3d-811b-44e87f082a98",

"name": "PetClinic Practice",

"schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json",

"\_exporter\_id": "49409721",

"\_collection\_link": "https://pauetisdev-3765426.postman.co/workspace/pauetisdev's-Workspace~1e320310-a0fd-4679-8dbf-2c5dc56c5af1/collection/49409721-a61a9b06-83ce-4c3d-811b-44e87f082a98?action=share&source=collection\_link&creator=49409721"

},

"item": [

{

"name": "Owners",

"item": [

{

"name": "Get All Owners",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta < 2s\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: És un array\r",

"pm.test(\"Response is an array\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData).to.be.an('array');\r",

"});\r",

"\r",

"// TEST 4: Array amb dades\r",

"pm.test(\"Array has data\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData.length).to.be.above(0);\r",

"});\r",

"\r",

"// TEST 5: Estructura correcta\r",

"pm.test(\"Owners have correct structure\", function () {\r",

" const jsonData = pm.response.json();\r",

" const owner = jsonData[0];\r",

" \r",

" pm.expect(owner).to.have.property('id');\r",

" pm.expect(owner).to.have.property('firstName');\r",

" pm.expect(owner).to.have.property('lastName');\r",

" pm.expect(owner).to.have.property('city');\r",

" pm.expect(owner).to.have.property('telephone');\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/owners",

"host": [

"{{baseUrl}}"

],

"path": [

"owners"

]

}

},

"response": []

},

{

"name": "Create Owner",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// --- Tests (Post-response Script) per a POST Owner (Simplificat) ---\r",

"\r",

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)\r",

"pm.test(\"Status code is 200 or 201\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Encadenament i Validació de Contingut (CORREGIT)\r",

"pm.test(\"Owner created, ID captured, and data verified\", function () {\r",

" \r",

" if (pm.response.code >= 400) {\r",

" return; // Sortim si hi ha un error HTTP (ex: 400 Bad Request)\r",

" }\r",

"\r",

" const response = pm.response.json();\r",

" \r",

" //Variables Temporals: Recuperem els valors que havíem enviat\r",

" const expectedFirstName = pm.environment.get(\"tempFirstName\");\r",

" const expectedLastName = pm.environment.get(\"tempLastName\");\r",

" const expectedTelephone = pm.environment.get(\"tempTelephone\");\r",

" \r",

" // --- Validació d'Estructura i Tipus ---\r",

"\r",

" pm.expect(response, \"Response must be a valid Owner object\").to.be.an('object').and.to.have.property('id');\r",

" pm.expect(response.id, \"ID must be a number\").to.be.a('number');\r",

"\r",

" pm.expect(response.firstName, \"FirstName must match sent data\").to.equal(expectedFirstName);\r",

" pm.expect(response.lastName, \"LastName must match sent data\").to.equal(expectedLastName);\r",

" \r",

" //Comprovem que el telèfon retornat té exactament 10 dígits\r",

" pm.expect(response.telephone.length, \"Telephone must have 10 digits\").to.equal(10);\r",

" \r",

" //Encadenament: Desem l'ID per utilitzar-lo en el PUT, DELETE, GET BY ID\r",

" pm.environment.set(\"createdOwnerId\", response.id);\r",

" console.log(\"Owner creat amb ID per a encadenament:\", response.id);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "prerequest",

"script": {

"exec": [

"// Per generar strings random (Nom, Cognom, Ciutat, Adreça)\r",

"const randomFirstName = pm.variables.replaceIn(\"{{$randomFirstName}}\");\r",

"const randomLastName = pm.variables.replaceIn(\"{{$randomLastName}}\");\r",

"const randomCity = pm.variables.replaceIn(\"{{$randomCity}}\");\r",

"const randomAddress = pm.variables.replaceIn(\"{{$randomStreetAddress}}\");\r",

"\r",

"const minTelephone = 1000000000; \r",

"const maxTelephone = 9999999999;\r",

"const randomTelephone = \_.random(minTelephone, maxTelephone).toString();\r",

"\r",

"// Guardar les dades\r",

"pm.environment.set(\"tempFirstName\", randomFirstName);\r",

"pm.environment.set(\"tempLastName\", randomLastName);\r",

"pm.environment.set(\"tempTelephone\", randomTelephone);\r",

"pm.environment.set(\"tempCity\", randomCity);\r",

"pm.environment.set(\"tempAddress\", randomAddress);\r",

"\r",

"console.log(`[PRE-REQ] Owner generat: ${randomFirstName} ${randomLastName}. Telèfon: ${randomTelephone}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [

{

"key": "Content-Type",

"value": "application/json",

"type": "text"

}

],

"body": {

"mode": "raw",

"raw": "{\r\n \"firstName\": \"{{tempFirstName}}\",\r\n \"lastName\": \"{{tempLastName}}\",\r\n \"address\": \"{{tempAddress}}\",\r\n \"city\": \"{{tempCity}}\",\r\n \"telephone\": \"{{tempTelephone}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/owners",

"host": [

"{{baseUrl}}"

],

"path": [

"owners"

]

}

},

"response": []

},

{

"name": "Update Onwer",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// --- Tests (Post-response Script) per a PUT Owner (Simplificat) ---\r",

"\r",

"// TEST 1: Codi d'estat esperat (200 OK o 204 NO CONTENT)\r",

"pm.test(\"Status code is 200 or 204\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar actualització (Només si el backend retorna dades)\r",

"pm.test(\"New LastName updated successfully (if body is returned)\", function () {\r",

" // Variable Temporal: Recuperem el valor que havíem enviat des del Pre-request\r",

" const expectedLastName = pm.environment.get(\"tempNewLastName\");\r",

"\r",

" // Si el backend respon amb 200 (OK), ha de retornar l'objecte\r",

" if (pm.response.code === 200) {\r",

" const response = pm.response.json();\r",

" \r",

" // Comprovem que el cognom retornat coincideix amb el que esperàvem\r",

" pm.expect(response.lastName).to.equal(expectedLastName);\r",

" }\r",

" \r",

" // Si el codi és 204 (No Content), la prova passa automàticament sense body per comprovar (server pot ronar 200 o 204, tot ok)\r",

" if (pm.response.code === 204) {\r",

" pm.expect(true).to.be.true; \r",

" }\r",

"});\r",

"\r",

"// TEST 4: Neteja de la variable temporal\r",

"pm.test(\"Clean up temp variable\", function () {\r",

" //Eliminem la variable temporal que només s'ha usat per enviar el Body d'aquest PUT\r",

" pm.environment.unset(\"tempNewLastName\");\r",

" pm.expect(true).to.be.true; // Assegurem que el test es registra com a PASS\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "prerequest",

"script": {

"exec": [

"// --- Pre-request Script per a PUT Owner ---\r",

"\r",

"// 1. Generar un nou valor de modificació amb el tag \"MODIFICAT\"\r",

"const newLastName = pm.variables.replaceIn(\"{{$randomLastName}}\") + \" MODIFICAT\";\r",

"\r",

"// 2. Guardar el nou valor en una variable temporal\r",

"pm.environment.set(\"tempNewLastName\", newLastName);\r",

"\r",

"// 3. Recuperar els valors originals dels altres camps des de l'entorn\r",

"const originalFirstName = pm.environment.get(\"tempFirstName\");\r",

"const originalAddress = pm.environment.get(\"tempAddress\");\r",

"const originalCity = pm.environment.get(\"tempCity\");\r",

"const originalTelephone = pm.environment.get(\"tempTelephone\");\r",

"\r",

"// Si algun camp original no existeix, avisar\r",

"if (!originalFirstName || !originalTelephone) {\r",

" console.error(\"ERROR: Dades d'Owner originals no trobades. Revisa la neteja al script POST.\");\r",

"}\r",

"\r",

"console.log(`[PUT PRE-REQ] Nou Cognom a enviar: ${newLastName}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [

{

"key": "Content-Type",

"value": "application/json",

"type": "text"

}

],

"body": {

"mode": "raw",

"raw": "{\r\n \"firstName\": \"{{tempFirstName}}\",\r\n \"lastName\": \"{{tempNewLastName}}\", //Dada a actualitzar\r\n \"address\": \"{{tempAddress}}\",\r\n \"city\": \"{{tempCity}}\",\r\n \"telephone\": \"{{tempTelephone}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}"

]

}

},

"response": []

},

{

"name": "Get Owner by Id",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"pm.test(\"Owner ID is 11 (Verification)\", function () {\r",

" if (pm.response.code !== 200) {\r",

" pm.expect(false, \"Pre-requisite failed: Status code must be 200.\").to.be.true;\r",

" return; \r",

" }\r",

" \r",

" const owner = pm.response.json();\r",

" const expectedOwnerId = pm.environment.get(\"createdOwnerId\");\r",

"\r",

" pm.expect(parseInt(owner.id)).to.equal(parseInt(expectedOwnerId));\r",

"});\r",

"\r",

"pm.test(\"Owner has all required fields\", function () {\r",

" const owner = pm.response.json();\r",

" pm.expect(owner).to.have.property('firstName');\r",

" pm.expect(owner).to.have.property('lastName');\r",

" pm.expect(owner).to.have.property('address');\r",

" pm.expect(owner).to.have.property('city');\r",

" pm.expect(owner).to.have.property('telephone');\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "prerequest",

"script": {

"exec": [

""

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}"

]

}

},

"response": []

},

{

"name": "Post add a new pet to an owner",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)",

"pm.test(\"Status code is 201 or 200\", function () {",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);",

"});",

"",

"// TEST 2: Encadenament i Validació",

"pm.test(\"Pet created, ID captured, and data verified\", function () {",

" ",

" if (pm.response.code >= 400) {",

" return; ",

" }",

"",

" const response = pm.response.json();",

" ",

" const expectedName = pm.environment.get(\"tempPetName\");",

" const expectedBirthDay = pm.environment.get(\"tempBirthDay\");",

" ",

" // 1. Validació d'estructura i ID",

" pm.expect(response, \"Response must be an object\").to.have.property('id');",

" pm.expect(response.id, \"ID must be a number\").to.be.a('number');",

" ",

" // 2. Validació de contingut: Comprovem que les dades coincideixen",

" pm.expect(response.name, \"Pet name must match sent data\").to.equal(expectedName);",

" pm.expect(response.birthDate, \"Birth date must match sent data\").to.equal(expectedBirthDay);",

" ",

" // Encadenament: Desem l'ID de la nova Mascota",

" pm.environment.set(\"createdPetId\", response.id);",

" console.log(\"Pet creada amb ID per a encadenament:\", response.id);",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "prerequest",

"script": {

"exec": [

"// --- Pre-request Script (Corregido) ---",

"",

"// Generación de Datos Temporales Únicos",

"const randomPetName = pm.variables.replaceIn(\"{{$randomFirstName}}\");",

"const fixedBirthDay = \"2025-01-01\";",

"",

"// Dades per al camp 'type'",

"const petTypeId = 2; // (ID=2 és 'dog' a la BD)",

"",

"// Guardar Dades Temporals a l'Entorn",

"pm.environment.set(\"tempPetName\", randomPetName);",

"pm.environment.set(\"tempBirthDay\", fixedBirthDay);",

"pm.environment.set(\"tempPetTypeId\", petTypeId);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"name\": \"{{tempPetName}}\",\r\n \"birthDate\": \"{{tempBirthDay}}\",\r\n \"type\": {\r\n \"id\": 2,\r\n \"name\": \"dog\"\r\n }\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}/pets",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}",

"pets"

]

}

},

"response": []

},

{

"name": "Update pet details",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const newPetName = pm.variables.replaceIn(\"{{$randomAdjective}}\") + \" UPDATED\";\r",

"pm.environment.set(\"tempNewPetName\", newPetName);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (200 OK o 204 NO CONTENT)\r",

"pm.test(\"Status code is 200 or 204\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar que el nom s'ha actualitzat (si el backend retorna body)\r",

"pm.test(\"New Pet Name updated successfully (from response body)\", function () {\r",

" if (pm.response.code !== 200) {\r",

" // Si és 204 (No Content), no hi ha body per comprovar.\r",

" return; \r",

" }\r",

" \r",

" const response = pm.response.json();\r",

" const expectedPetName = pm.environment.get(\"tempNewPetName\")\r",

"\r",

" // Comprovar que el nom tornat coincideix amb el que hem enviat\r",

" pm.expect(response.name).to.equal(expectedPetName);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"name\": \"{{tempNewPetName}}\",\r\n \"birthDate\": \"{{tempBirthDay}}\",\r\n \"type\": {\r\n \"id\": {{tempPetTypeId}},\r\n \"name\": \"{{tempPetType}}\"\r\n }\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}/pets/{{createdPetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}",

"pets",

"{{createdPetId}}"

]

}

},

"response": []

},

{

"name": "Get Pet by ID",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar que l'objecte Mascota existeix i té ID\r",

"pm.test(\"Response is a valid Pet object with ID\", function () {\r",

" //Si el Test 1 falla amb 404, no intentem llegir el JSON\r",

" if (pm.response.code !== 200) {\r",

" return; \r",

" }\r",

" \r",

" const pet = pm.response.json();\r",

" \r",

" // 1. Validació d'estructura (Correcció)\r",

" // Utilitzem 'to.have.property' només per comprovar que la clau existeix.\r",

" pm.expect(pet, \"Response must be an object\").to.be.an('object');\r",

"\r",

" //Comprovem que l'ID existeix\r",

" pm.expect(pet, 'The pet object must contain an ID').to.have.property('id');\r",

" \r",

" //Comprovem que el nom existeix\r",

" pm.expect(pet, 'The pet object must contain a name').to.have.property('name');\r",

" \r",

" // 2. Validació d'Encadenament: Comprovar que l'ID retornat coincideix amb el capturat.\r",

" const expectedPetId = pm.environment.get(\"createdPetId\");\r",

" \r",

" if (expectedPetId) {\r",

" // Assegurar que el valor de l'ID retornat sigui igual al que vam capturar (encadenament)\r",

" pm.expect(pet.id, \"The returned ID must match the createdPetId\").to.equal(parseInt(expectedPetId));\r",

" }\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}/pets/{{createdPetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}",

"pets",

"{{createdPetId}}"

]

}

},

"response": []

},

{

"name": "Post Add a vet visit for a pet",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"// Utilitzem la data d'avui en format YYYY-MM-DD\r",

"const today = new Date().toISOString().slice(0, 10); \r",

"\r",

"// Generem una descripció de la visita\r",

"const visitDescription = \"Routine checkup and annual vaccine.\";\r",

"\r",

"// 🔑 Guardar Dades Temporals\r",

"pm.environment.set(\"tempVisitDate\", today);\r",

"pm.environment.set(\"tempVisitDescription\", visitDescription);\r",

"\r",

"console.log(`[PRE-REQ] Visita programada per a: ${today}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// --- Tests (Post-response Script) per a POST Visit ---\r",

"\r",

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)\r",

"pm.test(\"Status code is 201 or 200\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);\r",

"});\r",

"\r",

"// TEST 2: Validació de Contingut i Neteja\r",

"pm.test(\"Visit validated and temp variables cleaned\", function () {\r",

" \r",

" if (pm.response.code == 204) {\r",

" return; //ignorar\r",

" }\r",

"\r",

" const response = pm.response.json();\r",

" const expectedDescription = pm.environment.get(\"tempVisitDescription\");\r",

"\r",

" // Validació: Comprovem que la descripció retornada coincideix amb la que hem enviat\r",

" pm.expect(response.description, \"Description must match sent data\").to.equal(expectedDescription);\r",

" \r",

" //Eliminem les variables temporals d'un sol ús\r",

" pm.environment.unset(\"tempVisitDescription\");\r",

" pm.environment.unset(\"tempVisitDate\");\r",

" \r",

" console.log(\"Visita creada i variables temporals netejades.\");\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"date\": \"{{tempVisitDate}}\",\r\n \"description\":\"{{tempVisitDescription}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}/pets/{{createdPetId}}/visits",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}",

"pets",

"{{createdPetId}}",

"visits"

]

}

},

"response": []

},

{

"name": "Delete Owner",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 204 NO CONTENT\r",

"pm.test(\"Status code is 204 (Success)\", function () {\r",

" pm.response.to.have.status(204);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Neteja de totes les variables de l'Owner\r",

"pm.test(\"Clean up ALL Owner environment variables\", function () {\r",

"\r",

" pm.environment.unset(\"createdOwnerId\");\r",

"\r",

" // 2.ELIMINACIÓ DE DADES TEMPORALS: Owner\r",

" pm.environment.unset(\"tempFirstName\");\r",

" pm.environment.unset(\"tempLastName\");\r",

" pm.environment.unset(\"tempAddress\");\r",

" pm.environment.unset(\"tempCity\");\r",

" pm.environment.unset(\"tempTelephone\");\r",

" pm.environment.unset(\"tempNewLastName\"); // Tambe la actualitzada\r",

"\r",

" // 3.ELIMINACIO DE DADES TEMP. DE PET I VISIT\r",

" pm.environment.unset(\"tempPetName\");\r",

" pm.environment.unset(\"tempBirthDay\"); \r",

" pm.environment.unset(\"tempPetType\");\r",

" pm.environment.unset(\"tempPetTypeId\");\r",

" pm.environment.unset(\"tempVisitDate\");\r",

" pm.environment.unset(\"tempVisitDescription\");\r",

" pm.environment.unset(\"tempNewPetName\")\r",

" console.log(\"Cicle complet Owner-Pet-Visit netejat. Variables d'entorn eliminades.\");\r",

" // Com que la funció és només netejar, forcem el PASS\r",

" pm.expect(true).to.be.true;\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "prerequest",

"script": {

"exec": [

""

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "{{baseUrl}}/owners/{{createdOwnerId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"owners",

"{{createdOwnerId}}"

]

}

},

"response": []

}

],

"description": "This folder contains endpoints to manage owner data and their associated pets within the PetClinic system. It provides comprehensive CRUD operations to create, retrieve, update, and delete owner records, including detailed owner information such as contact details and addresses. Additionally, it includes specialized endpoints to manage the relationship between owners and their pets, allowing you to add new pets to an owner's profile and track visit history. These endpoints support full lifecycle management of owner accounts and facilitate seamless integration between owner profiles, pet records, and veterinary visit tracking."

},

{

"name": "Pets",

"item": [

{

"name": "Get All Pets",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 500ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(500);\r",

"});\r",

"\r",

"// TEST 3: La resposta és un Array\r",

"pm.test(\"Response is an array (list of pets)\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData, \"The response must be a JSON array\").to.be.an('array');\r",

"});\r",

"\r",

"// TEST 4: L'array conté l'estructura bàsica si hi ha dades\r",

"pm.test(\"If data exists, first item has a 'name'\", function () {\r",

" const jsonData = pm.response.json();\r",

" if (jsonData.length > 0) {\r",

" pm.expect(jsonData[0]).to.have.property('name');\r",

" }\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/pets",

"host": [

"{{baseUrl}}"

],

"path": [

"pets"

]

}

},

"response": []

},

{

"name": "Get Pet by Id",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 300ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(300);\r",

"});\r",

"\r",

"// TEST 3: Verificar Encadenament i ID\r",

"pm.test(\"Pet retrieved successfully and ID verified\", function () {\r",

" if (pm.response.code !== 200) return;\r",

" \r",

" const pet = pm.response.json();\r",

" \r",

" //Variable d'Encadenament (Lectura)\r",

" const expectedPetId = pm.environment.get(\"createdPetId\");\r",

" \r",

" // 1. Verificació d'estructura\r",

" pm.expect(pet, \"Response must be a pet object\").to.be.an('object').and.to.have.property('id');\r",

" \r",

" // 2. Verificació de l'Encadenament (PK)\r",

" // Utilitzem parseInt per assegurar que comparem 'int'\r",

" pm.expect(parseInt(pet.id), \"The returned Pet ID must match the chained ID\").to.equal(parseInt(expectedPetId));\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/pets/{{createdPetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pets",

"{{createdPetId}}"

]

}

},

"response": []

},

{

"name": "Update Pet Details",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const newPetName = pm.environment.replaceIn(\"{{$randomAdjective}}\") + \"UPDATED\";\r",

"const fixedPetType = \"dog\";\r",

"\r",

"pm.environment.set(\"tempNewPetName\", newPetName);\r",

"pm.environment.set(\"tempPetType\", fixedPetType);\r",

"\r",

"console.log(`[PRE-REQ] Nou nom Mascota a enviar: ${newPetName}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200 or 204\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// Verificar que el nom s'ha actualitzat (si el backend retorna body)\r",

"pm.test(\"New Pet Name updated successfully (from response body)\", function () {\r",

" if (pm.response.code !== 200) {\r",

" // Si és 204 (No Content), no hi ha body per comprovar.\r",

" return; \r",

" }\r",

" \r",

" const response = pm.response.json();\r",

" const expectedPetName = pm.environment.get(\"tempNewPetName\")\r",

"\r",

" // Comprovar que el nom tornat coincideix amb el que hem enviat\r",

" pm.expect(response.name).to.equal(expectedPetName);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"id\": {{createdPetId}}, \r\n \"name\": \"{{tempNewPetName}}\", // Name Actualitzat\r\n \"birthDate\": \"{{tempBirthDay}}\", \r\n \"type\": {\r\n \"id\": {{tempPetTypeId}}, \r\n \"name\": \"{{tempPetType}}\"\r\n }\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/pets/{{createdPetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pets",

"{{createdPetId}}"

]

}

},

"response": []

},

{

"name": "Delete a Pet",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 204 NO CONTENT\r",

"pm.test(\"Status code is 204 (Success)\", function () {\r",

" pm.response.to.have.status(204);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Neteja de la variable d'encadenament\r",

"pm.test(\"Clean up createdPetId variable\", function () {\r",

" //Neteja de l'ID de la Mascota: ja no es necessita per a Pet Types ni per a cap altre Pet\r",

" pm.environment.unset(\"createdPetId\");\r",

" console.log(\"createdPetId eliminat de l'entorn.\");\r",

" pm.expect(true).to.be.true;\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "{{baseUrl}}/pets/{{createdPetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pets",

"{{createdPetId}}"

]

}

},

"response": []

}

],

"description": "This folder contains endpoints to manage pet data and medical records within the PetClinic system. It provides operations to retrieve all pets in the database, fetch individual pet details by ID, update pet information including breed, age, and medical history, and remove pet records when necessary. These endpoints support comprehensive pet profile management, enabling tracking of pet demographics, health status, and ownership associations. The API facilitates efficient pet data administration and serves as a core component for veterinary care coordination."

},

{

"name": "Vets",

"item": [

{

"name": "Get All Pets",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 500ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(500);\r",

"});\r",

"\r",

"// TEST 3: La resposta és un Array\r",

"pm.test(\"Response is an array (list of vets)\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData, \"The response must be a JSON array\").to.be.an('array');\r",

"});\r",

"\r",

"// TEST 4: Comprovar que l'array no estigui buit i que el primer element tingui 'firstName'\r",

"pm.test(\"Array is not empty and first item has 'firstName'\", function () {\r",

" const jsonData = pm.response.json();\r",

" \r",

" pm.expect(jsonData.length, \"The array of Vets should not be empty\").to.be.above(0);\r",

" pm.expect(jsonData[0]).to.have.property('firstName');\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/vets",

"host": [

"{{baseUrl}}"

],

"path": [

"vets"

]

}

},

"response": []

},

{

"name": "Add a new Vet",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const randomFirstName = pm.variables.replaceIn(\"{{$randomFirstName}}\")\r",

"const randomLastName = pm.variables.replaceIn(\"{{$randomLastName}}\")\r",

"\r",

"pm.environment.set(\"tempVetFirstName\", randomFirstName);\r",

"pm.environment.set(\"tempVetLastName\", randomLastName);\r",

"\r",

"console.log(`[PRE-REQ] Vet generat: ${randomFirstName} ${randomLastName}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)\r",

"pm.test(\"Status code is 201 or 200\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);\r",

"});\r",

"\r",

"// TEST 2: Encadenament i Validació de Contingut\r",

"pm.test(\"Vet created, ID captured, and name verified\", function () {\r",

" \r",

" if (pm.response.code >= 400) {\r",

" return; \r",

" }\r",

"\r",

" const response = pm.response.json();\r",

" \r",

" // 1. Validació d'estructura i ID\r",

" pm.expect(response, \"Response must be an object\").to.have.property('id');\r",

" pm.expect(response.id, \"ID must be a number\").to.be.a('number');\r",

"\r",

" //Lectura de les Variables Temporals per a la verificació\r",

" const expectedFirstName = pm.environment.get(\"tempVetFirstName\");\r",

" const expectedLastName = pm.environment.get(\"tempVetLastName\");\r",

" \r",

" // 2. Validació de contingut: Comprovem que el nom coincideix\r",

" pm.expect(response.firstName, \"FirstName must match sent data\").to.equal(expectedFirstName);\r",

" pm.expect(response.lastName, \"LastName must match sent data\").to.equal(expectedLastName);\r",

" \r",

" //Encadenament: Desem l'ID del nou Vet\r",

" pm.environment.set(\"createVetId\", response.id);\r",

" console.log(\"Veterinari creat amb ID per a encadenament:\", response.id);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"firstName\":\"{{tempVetFirstName}}\",\r\n \"lastName\": \"{{tempVetLastName}}\",\r\n \"specialties\": []\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/vets",

"host": [

"{{baseUrl}}"

],

"path": [

"vets"

]

}

},

"response": []

},

{

"name": "Update Vet details",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const newVetFirstName = pm.environment.replaceIn(\"{{$randomFirstName}}\") + \" UPDATED\";\r",

"pm.environment.set(\"tempVetNewFirstName\", newVetFirstName);\r",

"\r",

"\r",

"console.log(`[PRE-REQ] Nou veterianri a enviar: ${newVetFirstName}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (200 OK o 204 NO CONTENT)\r",

"pm.test(\"Status code is 200 or 204\", function () {\r",

" // 200 si retorna el body, 204 si no retorna contingut\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar actualització del PRIMER NOM\r",

"pm.test(\"Updated FirstName verified (if body is returned)\", function () {\r",

" //Lectura de la Variable Temporal\r",

" const expectedNewFirstName = pm.environment.get(\"tempVetNewFirstName\");\r",

" \r",

" // Si la resposta és 204 (No Content), la prova passa sense Body\r",

" if (pm.response.code === 204) {\r",

" pm.expect(true, \"Actualització confirmada pel codi 204.\").to.be.true; \r",

" return; \r",

" } \r",

" \r",

" // Si la resposta és 200 (OK), verifiquem el Body\r",

" else if (pm.response.code === 200) {\r",

" const response = pm.response.json();\r",

" \r",

" // Comprovem que el nom retornat coincideix amb el valor temporal nou\r",

" pm.expect(response.firstName, \"The returned FirstName must match the temporary new value\").to.equal(expectedNewFirstName);\r",

" }\r",

"});\r",

"\r",

"// NO ELIMINAR LA NOVA VARIABLE ACTUALITZADA, PER PODER COMPROVARLA EN EL GET BY ID"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"firstName\":\"{{tempVetNewFirstName}}\", //Actulitzada\r\n \"lastName\": \"{{tempVetLastName}}\",\r\n \"specialties\": []\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/vets/{{createVetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"vets",

"{{createVetId}}"

]

}

},

"response": []

},

{

"name": "Get Pet by Id",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 300ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(300);\r",

"});\r",

"\r",

"// TEST 3: Verificar l'estructura de la resposta\r",

"pm.test(\"Vet object retrieved successfully\", function () {\r",

" const vet = pm.response.json();\r",

" \r",

" // 1. Verificació d'estructura bàsica\r",

" pm.expect(vet, \"Response must be a Vet object\").to.be.an('object');\r",

" pm.expect(vet).to.have.property('id');\r",

" pm.expect(vet).to.have.property('firstName');\r",

" pm.expect(vet).to.have.property('lastName');\r",

"});\r",

"\r",

"// TEST 4: Verificar l'ID que hem demanat\r",

"pm.test(\"Verify the requested ID\", function () {\r",

" const vet = pm.response.json();\r",

" const expectedVetId = pm.environment.get(\"createVetId\");\r",

" \r",

" pm.expect(parseInt(vet.id), \"The returned Vet ID must be 1\").to.equal(expectedVetId);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/vets/{{createVetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"vets",

"{{createVetId}}"

]

}

},

"response": []

},

{

"name": "Delete Pet Type",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 204 NO CONTENT\r",

"pm.test(\"Status code is 204 (Success)\", function () {\r",

" pm.response.to.have.status(204);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Neteja COMPLETA de totes les variables\r",

"pm.test(\"Clean up createdVetId and all temporary variables\", function () {\r",

" \r",

" // 1. ELIMINACIÓ DE L'ID PRINCIPAL D'ENCADENAMENT\r",

" pm.environment.unset(\"createVetId\");\r",

" \r",

" // 2. ELIMINACIÓ DE DADES TEMPORALS SOBREVIVENTS (Primer i Cognom original)\r",

" pm.environment.unset(\"tempVetFirstName\"); \r",

" pm.environment.unset(\"tempVetLastName\");\r",

" \r",

" // 3. ELIMINACIÓ DE LA VARIABLE ACTUALITZADA\r",

" pm.environment.unset(\"tempVetNewFirstName\"); \r",

"\r",

" console.log(\"Cicle Vets complet i variables d'entorn netejades.\");\r",

" // Forçar el test com a PASS\r",

" pm.expect(true).to.be.true;\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "{{baseUrl}}/vets/{{createVetId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"vets",

"{{createVetId}}"

]

}

},

"response": []

}

],

"description": "This folder contains endpoints to manage veterinarian data and professional profiles within the PetClinic system. It includes full CRUD operations to create new vet accounts, retrieve complete veterinarian listings or individual vet details, update professional information such as specialties and credentials, and remove vet records from the system. These endpoints enable comprehensive management of veterinary staff, including their qualifications, areas of expertise, and availability. The API supports clinic administration, staff scheduling, and helps match pets with appropriate veterinary specialists based on their medical needs."

},

{

"name": "Pet Types",

"item": [

{

"name": "Get All Pet Types",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 500ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(500);\r",

"});\r",

"\r",

"// TEST 3: La resposta és un Array\r",

"pm.test(\"Response is an array (list of vets)\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData, \"The response must be a JSON array\").to.be.an('array');\r",

"});\r",

"\r",

"// TEST 4: Comprovar que l'array no estigui buit i que el primer element tingui 'firstName'\r",

"pm.test(\"Array is not empty and first item has 'name'\", function () {\r",

" const jsonData = pm.response.json();\r",

" \r",

" pm.expect(jsonData.length, \"The array of types of pets should not be empty\").to.be.above(0);\r",

" pm.expect(jsonData[0]).to.have.property('name');\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/pettypes",

"host": [

"{{baseUrl}}"

],

"path": [

"pettypes"

]

}

},

"response": []

},

{

"name": "Add a new Pet Type",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const randomTypeName = pm.variables.replaceIn(\"{{$randomAdjective}}\")\r",

"pm.environment.set(\"tempTypeName\", randomTypeName)\r",

"\r",

"console.log(`[PRE-REQ] Nou tipus generat: ${randomTypeName}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)\r",

"pm.test(\"Status code is 201 or 200\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);\r",

"});\r",

"\r",

"// TEST 2: Encadenament i Validació\r",

"pm.test(\"Pet Type created, ID captured, and name verified\", function () {\r",

" \r",

" if (pm.response.code >= 400) {\r",

" return;\r",

" }\r",

" const response = pm.response.json();\r",

" const expectedName = pm.environment.get(\"tempTypeName\"); \r",

"\r",

" pm.expect(response, \"Response must be an object\").to.have.property('id'); \r",

" \r",

" // Comprovem que el valor de l'ID és un número\r",

" pm.expect(response.id, \"ID must be a number\").to.be.a('number');\r",

" \r",

" // Validació de contingut: El nom retornat ha de ser el que hem enviat\r",

" pm.expect(response.name, \"Name must match sent data\").to.equal(expectedName);\r",

" \r",

" // Guardem l'ID amb el nom consistent\r",

" pm.environment.set(\"createPetTypeId\", response.id);\r",

" console.log(\"Pet Type creat amb ID per a encadenament:\", response.id);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"name\": \"{{tempTypeName}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/pettypes",

"host": [

"{{baseUrl}}"

],

"path": [

"pettypes"

]

}

},

"response": []

},

{

"name": "Update Pet Type details",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const newTypePet = pm.variables.replaceIn(\"{{$randomAdjective}}\")",

"const petTypeId = pm.environment.get(\"createPetTypeId\");",

"",

"pm.environment.set(\"tempNewTypeName\", newTypePet)",

"pm.environment.set(\"petTypeIdForBody\", petTypeId);",

"",

"console.log(`[PRE-REQ] Nou PetType a enviar: ${newTypePet}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"\r",

"// TEST 1: Codi d'estat esperat (200 OK o 204 NO CONTENT)\r",

"\r",

"pm.test(\"Status code is 200 or 204\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar actualització del nom\r",

"\r",

"pm.test(\"Updated Type Name verified\", function () {\r",

" const expectedNewTypeName = pm.environment.get(\"tempNewTypeName\");\r",

" \r",

" if (pm.response.code === 204) {\r",

" pm.expect(true, \"Actualització confirmada (Body buit).\").to.be.true;\r",

" return;\r",

" }\r",

" // Si la resposta és 200 (OK), verifiquem el Body\r",

" else if (pm.response.code === 200) {\r",

" const response = pm.response.json();\r",

" // Comprovem que el nom retornat coincideix amb el valor temporal noU\r",

" pm.expect(response.name, \"The returned name must match the temporary new value\").to.equal(expectedNewTypeName);\r",

" }\r",

" // No netejem la variable de tempNewTypeName, per utilizar-la al Get Pet by Id.\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"id\": {{petTypeIdForBody}},\r\n \"name\": \"{{tempNewTypeName}}\" //Actualitzat\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/pettypes/{{createPetTypeId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pettypes",

"{{createPetTypeId}}"

]

}

},

"response": []

},

{

"name": "Get Pet Type By Id",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 300ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(300);\r",

"});\r",

"\r",

"// TEST 3: Verificar Encadenament i Persistència del Nou Nom\r",

"pm.test(\"Type ID and UPDATED Name verified\", function () {\r",

" if (pm.response.code !== 200) return;\r",

" \r",

" const petType = pm.response.json();\r",

" \r",

" //Variables d'Encadenament i Temporals\r",

" const expectedPetTypeId = pm.environment.get(\"createPetTypeId\");\r",

" const expectedNewTypeName = pm.environment.get(\"tempNewTypeName\"); \r",

" \r",

" // 1. Validació de l'ID (Comprovació d'Encadenament)\r",

" pm.expect(petType).to.have.property('id');\r",

" pm.expect(parseInt(petType.id), \"The returned ID must match the chained ID\").to.equal(parseInt(expectedPetTypeId));\r",

" \r",

" // 2.Comprovem que el nom a la DB és el nom actualitzat\r",

" pm.expect(petType.name, \"The retrieved name must be the UPDATED name\").to.equal(expectedNewTypeName);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"protocolProfileBehavior": {

"disableBodyPruning": true

},

"request": {

"method": "GET",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"id\": {{petTypeIdForBody}},\r\n \"name\": \"{{tempNewTypeName}}\" //Actualitzat\r\n}"

},

"url": {

"raw": "{{baseUrl}}/pettypes/{{createPetTypeId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pettypes",

"{{createPetTypeId}}"

]

}

},

"response": []

},

{

"name": "Delete Pet Type",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 204 NO CONTENT\r",

"pm.test(\"Status code is 204 (Success)\", function () {\r",

" pm.response.to.have.status(204);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Neteja COMPLETA de totes les variables\r",

"pm.test(\"Clean up createPetTypeId and all temporary variables\", function () {\r",

" \r",

" // 1. ELIMINACIÓ DE L'ID PRINCIPAL D'ENCADENAMENT\r",

" pm.environment.unset(\"createPetTypeId\");\r",

" \r",

" // 2. ELIMINACIÓ DE DADES TEMPORALS SOBREVIVENTS (Primer i Cognom original)\r",

" pm.environment.unset(\"petTypeIdForBody\");\r",

" pm.environment.unset(\"tempTypeName\"); \r",

" \r",

" // 3. ELIMINACIÓ DE LA VARIABLE ACTUALITZADA\r",

" pm.environment.unset(\"tempNewTypeName\"); \r",

"\r",

" console.log(\"Cicle PetType complet i variables d'entorn netejades.\");\r",

" // Forçar el test com a PASS\r",

" pm.expect(true).to.be.true;\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "{{baseUrl}}/pettypes/{{createPetTypeId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"pettypes",

"{{createPetTypeId}}"

]

}

},

"response": []

}

],

"description": "This folder contains endpoints for managing pet types in the PetClinic system. Pet types represent the different categories of pets that can be registered in the clinic (e.g., dog, cat, bird, etc.).\n\nThe available operations include:\n- \*\*Get All Pet Types\*\*: Retrieve a complete list of all pet types available in the system\n- \*\*Add a new Pet Type\*\*: Create a new pet type category\n- \*\*Update Pet Type details\*\*: Modify an existing pet type's information\n- \*\*Get Pet Type By Id\*\*: Retrieve a specific pet type using its unique identifier\n- \*\*Delete Pet Type\*\*: Remove a pet type from the system\n\nThese endpoints allow complete CRUD (Create, Read, Update, Delete) operations for managing the pet type data in the PetClinic application."

},

{

"name": "Visits",

"item": [

{

"name": "Get All Visits",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat 200 OK\r",

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 500ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(500);\r",

"});\r",

"\r",

"// TEST 3: La resposta és un Array\r",

"pm.test(\"Response is an array (list of vets)\", function () {\r",

" const jsonData = pm.response.json();\r",

" pm.expect(jsonData, \"The response must be a JSON array\").to.be.an('array');\r",

"});\r",

"\r",

"// TEST 4: Comprovar que l'array no estigui buit i que el primer element tingui 'firstName'\r",

"pm.test(\"Array is not empty and first item has 'name'\", function () {\r",

" const jsonData = pm.response.json();\r",

" \r",

" pm.expect(jsonData.length, \"The array of types of pets should not be empty\").to.be.above(0);\r",

" pm.expect(jsonData[0]).to.have.property('date');\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/visits",

"host": [

"{{baseUrl}}"

],

"path": [

"visits"

]

}

},

"response": []

},

{

"name": "Add new Visit",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const visitDate = \"2025-01-01\"\r",

"const randomDescription = pm.variables.replaceIn(\"{{$randomCatchPhrase}}\")\r",

"\r",

"// Agafar el ID de la mascota\r",

"const petId = pm.environment.get(\"createdPetId\")\r",

"\r",

"pm.environment.set(\"tempVisitDate\", visitDate)\r",

"pm.environment.set(\"tempDescription\", randomDescription)\r",

"pm.environment.set(\"petIDforVisit\", petId)\r",

"\r",

"console.log(`[PRE-REQ] Nueva Visita para Pet ID ${petId}: ${visitDate} - ${randomDescription}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (201 Created o 200 OK)\r",

"pm.test(\"Status code is 201 or 200\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 201]);\r",

"});\r",

"\r",

"// TEST 2: Encadenament i Validació\r",

"pm.test(\"Visit Created\", function () {\r",

" \r",

" if (pm.response.code >= 400) {\r",

" return; \r",

" }\r",

"\r",

" const response = pm.response.json();\r",

" \r",

" const expectedDescription = pm.environment.get(\"tempDescription\");\r",

" const expectedPetId = pm.environment.get(\"petIDforVisit\");\r",

" \r",

" // 1. Validació d'estructura i ID\r",

" pm.expect(response, \"Response must be an object\").to.have.property('id');\r",

" pm.expect(response.id, \"ID must be a number\").to.be.a('number');\r",

" \r",

" // 2. Validació de contingut: Comprovem que les dades coincideixen\r",

" pm.expect(response.description, \"Description must match sent data\").to.equal(expectedDescription);\r",

" pm.expect(response.petId, \"Pet Id must match sent data\").to.equal(expectedPetId);\r",

" \r",

" // Encadenament: Desem l'ID de la nova Mascota\r",

" pm.environment.set(\"createVisitId\", response.id);\r",

" console.log(\"Visit creada amb ID per a encadenament:\", response.id);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "POST",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"date\": \"{{tempVisitDate}}\",\r\n \"description\": \"{{tempDescription}}\",\r\n \"petId\": \"{{petIDforVisit}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/visits",

"host": [

"{{baseUrl}}"

],

"path": [

"visits"

]

}

},

"response": []

},

{

"name": "Update Visit",

"event": [

{

"listen": "prerequest",

"script": {

"exec": [

"const tempNewDescription = pm.variables.replaceIn(\"{$randomCatchPhrase}\")\r",

"\r",

"pm.environment.set(\"tempNewDescription\", tempNewDescription)\r",

"\r",

"console.log(`[PRE-REQ] Nova Descripció a enviar: ${tempNewDescription}`);"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

},

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat (200 OK o 204 NO CONTENT)\r",

"pm.test(\"Status code is 200 or 204\", function () {\r",

" pm.expect(pm.response.code).to.be.oneOf([200, 204]);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar que la descripcio s'ha actualitzat (si el backend retorna body)\r",

"pm.test(\"New Pet Name updated successfully (from response body)\", function () {\r",

" if (pm.response.code !== 200) {\r",

" // Si és 204 (No Content), no hi ha body per comprovar.\r",

" return; \r",

" }\r",

" \r",

" const response = pm.response.json();\r",

" const expectedDescription = pm.environment.get(\"tempNewDescription\")\r",

"\r",

" // Comprovar que el nom tornat coincideix amb el que hem enviat\r",

" pm.expect(response.description).to.equal(expectedDescription);\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "PUT",

"header": [],

"body": {

"mode": "raw",

"raw": "{\r\n \"date\": \"{{tempVisitDate}}\",\r\n \"description\": \"{{tempNewDescription}}\",\r\n \"petId\": \"{{petIDforVisit}}\"\r\n}",

"options": {

"raw": {

"language": "json"

}

}

},

"url": {

"raw": "{{baseUrl}}/visits/{{createVisitId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"visits",

"{{createVisitId}}"

]

}

},

"response": []

},

{

"name": "Get Visit by Id",

"event": [

{

"listen": "test",

"script": {

"exec": [

"pm.test(\"Status code is 200\", function () {\r",

" pm.response.to.have.status(200);\r",

"});\r",

"\r",

"pm.test(\"Response time < 2000ms\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Verificar que l'objecte visit existeix i té ID\r",

"pm.test(\"Response is a valid Pet object with ID\", function () {\r",

" //Si el Test 1 falla amb 404, no intentem llegir el JSON\r",

" if (pm.response.code !== 200) {\r",

" return; \r",

" }\r",

" \r",

" const visit = pm.response.json();\r",

" \r",

" // 1. Validació d'estructura (Correcció)\r",

" // Utilitzem 'to.have.property' només per comprovar que la clau existeix.\r",

" pm.expect(visit, \"Response must be an object\").to.be.an('object');\r",

"\r",

" //Comprovem que l'ID existeix\r",

" pm.expect(visit, 'The visit object must contain an ID').to.have.property('id');\r",

" \r",

" //Comprovem que el nom existeix\r",

" pm.expect(visit, 'The visit object must contain a description').to.have.property('description');\r",

" \r",

" // 2. Validació d'Encadenament: Comprovar que l'ID retornat coincideix amb el capturat.\r",

" const expectedVisitId = pm.environment.get(\"createVisitId\");\r",

" \r",

" if (expectedVisitId) {\r",

" // Assegurar que el valor de l'ID retornat sigui igual al que vam capturar (encadenament)\r",

" pm.expect(visit.id, \"The returned ID must match the createdPetId\").to.equal(parseInt(expectedVisitId));\r",

" }\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "GET",

"header": [],

"url": {

"raw": "{{baseUrl}}/visits/{{createVisitId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"visits",

"{{createVisitId}}"

]

}

},

"response": []

},

{

"name": "Delete Visit",

"event": [

{

"listen": "test",

"script": {

"exec": [

"// TEST 1: Codi d'estat esperat 204 NO CONTENT\r",

"pm.test(\"Status code is 204 (Success)\", function () {\r",

" pm.response.to.have.status(204);\r",

"});\r",

"\r",

"// TEST 2: Temps de resposta ràpid\r",

"pm.test(\"Response time is fast (< 2000ms)\", function () {\r",

" pm.expect(pm.response.responseTime).to.be.below(2000);\r",

"});\r",

"\r",

"// TEST 3: Neteja de totes les variables de Visit\r",

"pm.test(\"Clean up ALL Visit environment variables\", function () {\r",

"\r",

" pm.environment.unset(\"createVisitId\"); // important\r",

"\r",

" // 2.ELIMINACIÓ DE DADES TEMPORALS: Visit\r",

" pm.environment.unset(\"tempVisitDate\");\r",

" pm.environment.unset(\"tempDescription\");\r",

"\r",

" pm.environment.unset(\"tempNewDescription\"); // Tambe la actualitzada\r",

"\r",

" // 3.ELIMINACIO DE DADES TEMP. DE PET I VISIT\r",

" pm.environment.unset(\"tempPetName\");\r",

" pm.environment.unset(\"tempBirthDay\"); \r",

" pm.environment.unset(\"tempPetType\");\r",

" pm.environment.unset(\"tempPetTypeId\");\r",

" pm.environment.unset(\"tempVisitDate\");\r",

" pm.environment.unset(\"tempVisitDescription\");\r",

" pm.environment.unset(\"tempNewPetName\")\r",

" console.log(\"Cicle complet Visit netejat. Variables d'entorn eliminades.\");\r",

" // Com que la funció és només netejar, forcem el PASS\r",

" pm.expect(true).to.be.true;\r",

"});"

],

"type": "text/javascript",

"packages": {},

"requests": {}

}

}

],

"request": {

"method": "DELETE",

"header": [],

"url": {

"raw": "{{baseUrl}}/visits/{{createVisitId}}",

"host": [

"{{baseUrl}}"

],

"path": [

"visits",

"{{createVisitId}}"

]

}

},

"response": []

}

],

"description": "# Visits\n\nManage pet clinic visits with full CRUD operations.\n\n## Endpoints\n\n\*\*Get All Visits\*\* - `GET {{baseUrl}}/visits` \nRetrieve all visit records.\n\n\*\*Add new Visit\*\* - `POST {{baseUrl}}/visits` \nCreate a new visit record.\n\n\*\*Update Visit\*\* - `PUT {{baseUrl}}/visits/{{createVisitId}}` \nModify an existing visit.\n\n\*\*Get Visit by Id\*\* - `GET {{baseUrl}}/visits/{{createVisitId}}` \nRetrieve a specific visit by ID.\n\n\*\*Delete Visit\*\* - `DELETE {{baseUrl}}/visits/{{createVisitId}}` \nRemove a visit record.\n\n## Notes\n\n- Use `{{createVisitId}}` variable for visit-specific operations\n \n- Configure `{{baseUrl}}` in your environment"

}

]

}