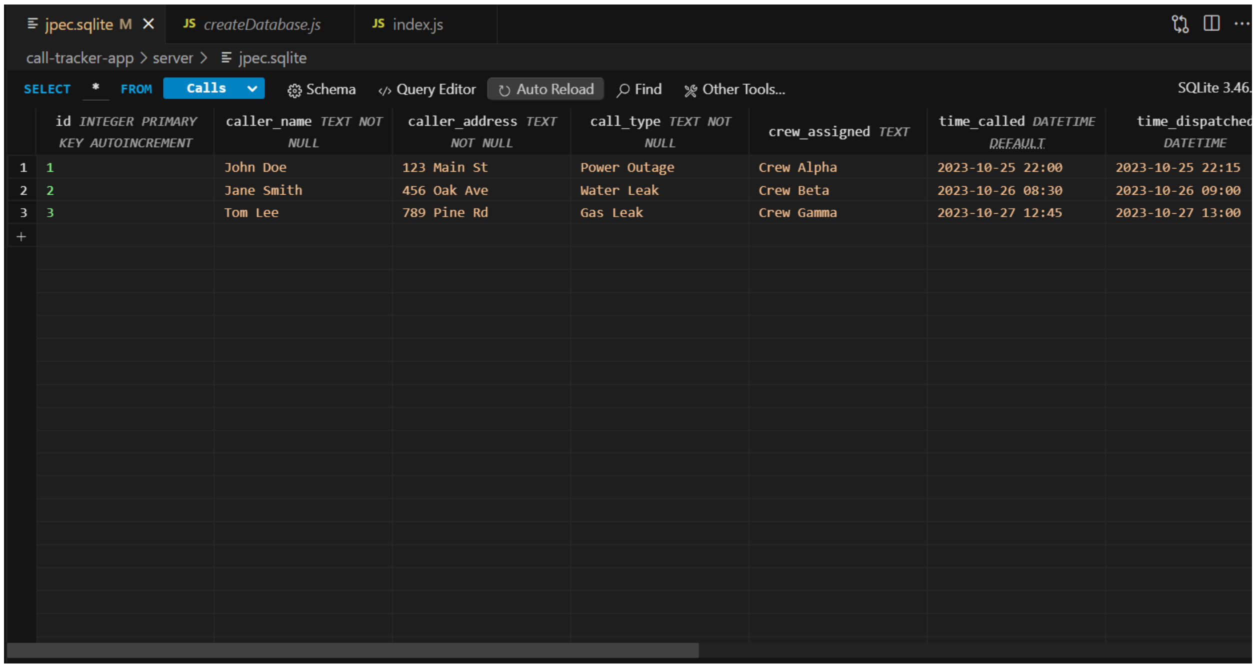
Deliverable 2 Postman Results

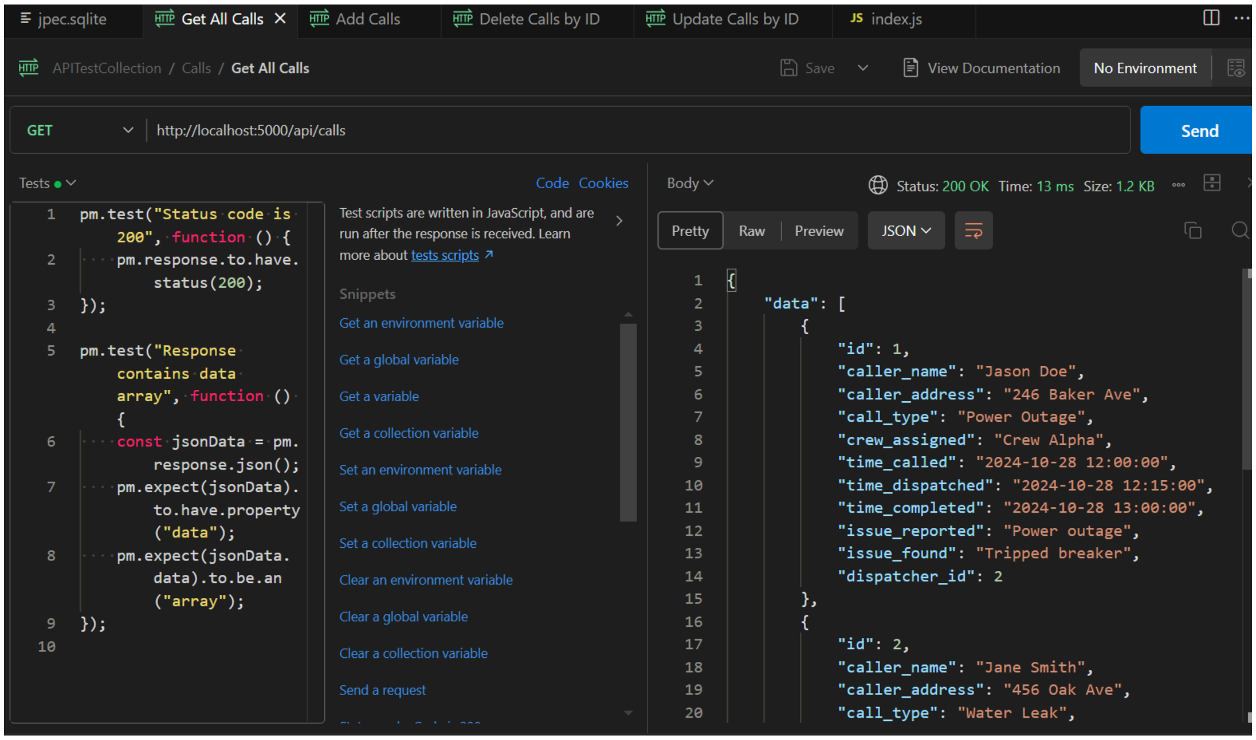
Calls

Calls Table Starting Out:



	id INTEGER PRIMARY KEY AUTOINCREMENT	caller_name TEXT NOT NULL	caller_address TEXT NOT NULL	call_type TEXT NOT NULL	crew_assigned TEXT	time_called DATETIME DEFAULT	time_dispatched DATETIME
1	1	John Doe	123 Main St	Power Outage	Crew Alpha	2023-10-25 22:00	2023-10-25 22:15
2	2	Jane Smith	456 Oak Ave	Water Leak	Crew Beta	2023-10-26 08:30	2023-10-26 09:00
3	3	Tom Lee	789 Pine Rd	Gas Leak	Crew Gamma	2023-10-27 12:45	2023-10-27 13:00

Getting All Calls:



jpec.sqlite | Get All Calls | Add Calls | Delete Calls by ID | Update Calls by ID | index.js

APITestCollection / Calls / Get All Calls

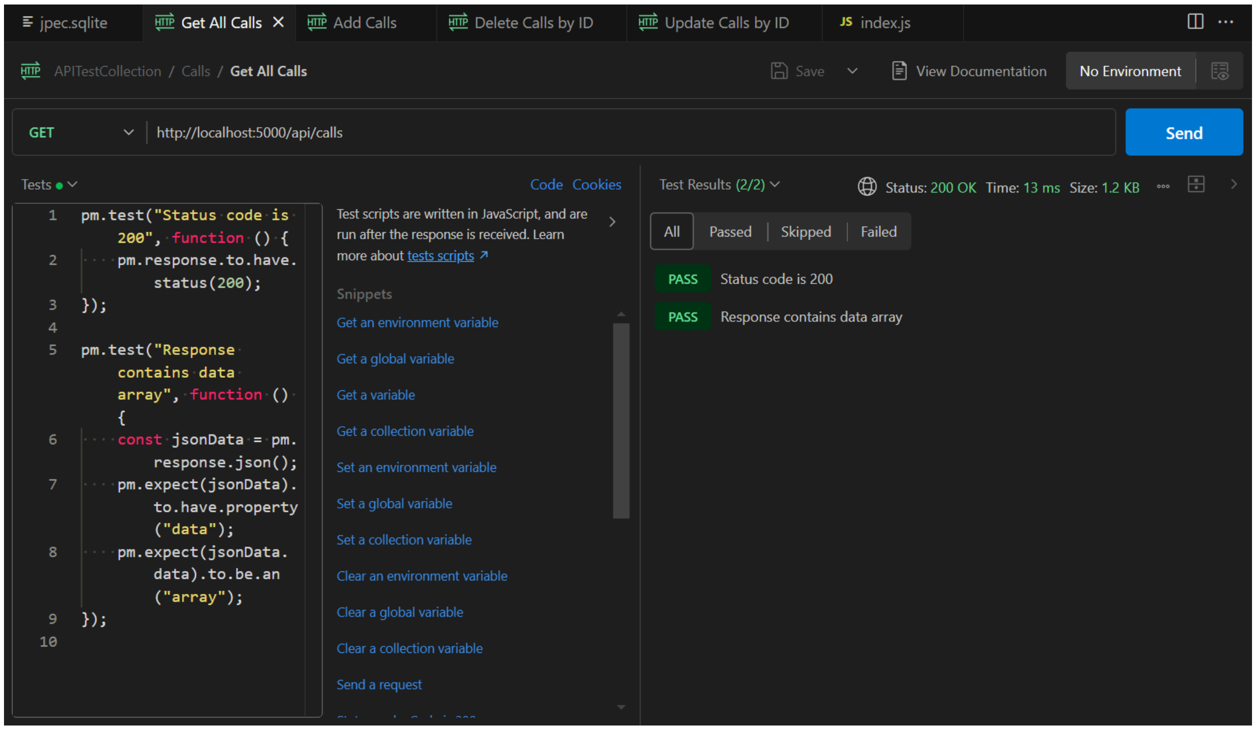
GET http://localhost:5000/api/calls

Tests

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains data array", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data");
    pm.expect(jsonData.data).to.be.an("array");
});
```

Status: 200 OK Time: 13 ms Size: 1.2 KB

```
{
    "data": [
        {
            "id": 1,
            "caller_name": "Jason Doe",
            "caller_address": "246 Baker Ave",
            "call_type": "Power Outage",
            "crew_assigned": "Crew Alpha",
            "time_called": "2024-10-28 12:00:00",
            "time_dispatched": "2024-10-28 12:15:00",
            "time_completed": "2024-10-28 13:00:00",
            "issue_reported": "Power outage",
            "issue_found": "Tripped breaker",
            "dispatcher_id": 2
        },
        {
            "id": 2,
            "caller_name": "Jane Smith",
            "caller_address": "456 Oak Ave",
            "call_type": "Water Leak",
```



GET http://localhost:5000/api/calls

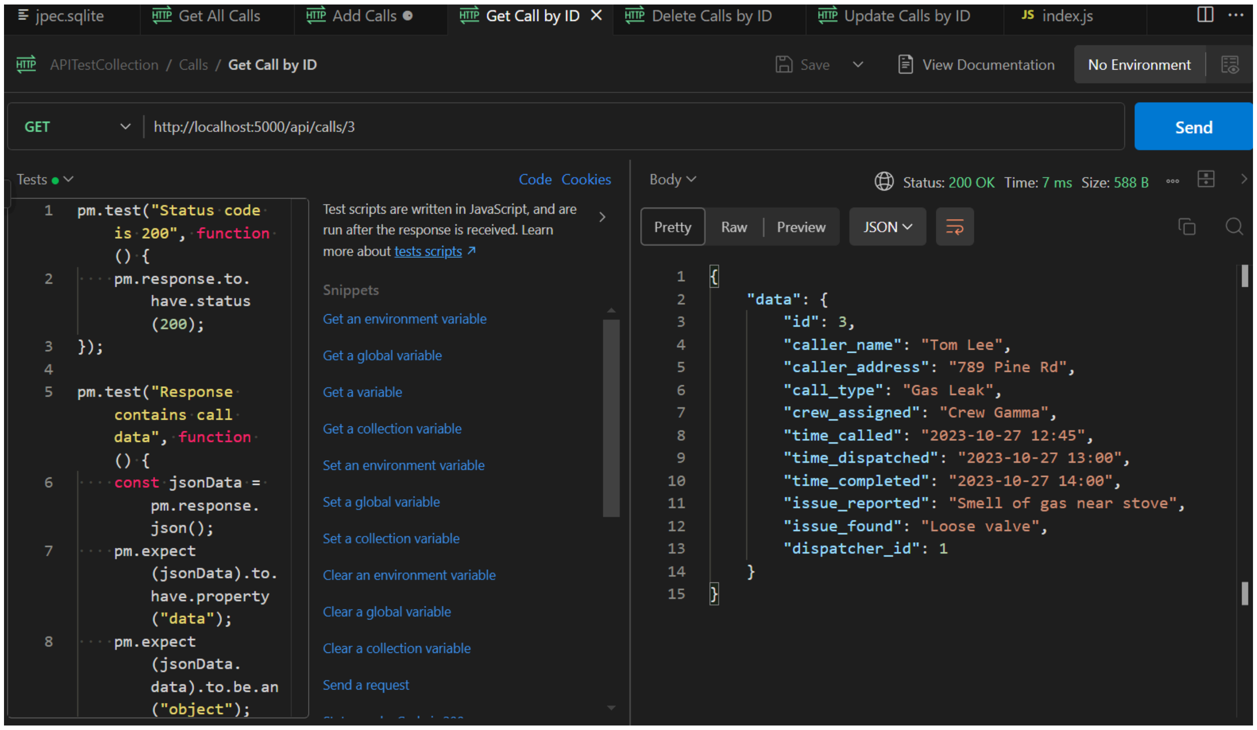
Test Results (2/2)

Status: 200 OK Time: 13 ms Size: 1.2 KB

PASS Status code is 200

PASS Response contains data array

Get Call by ID:



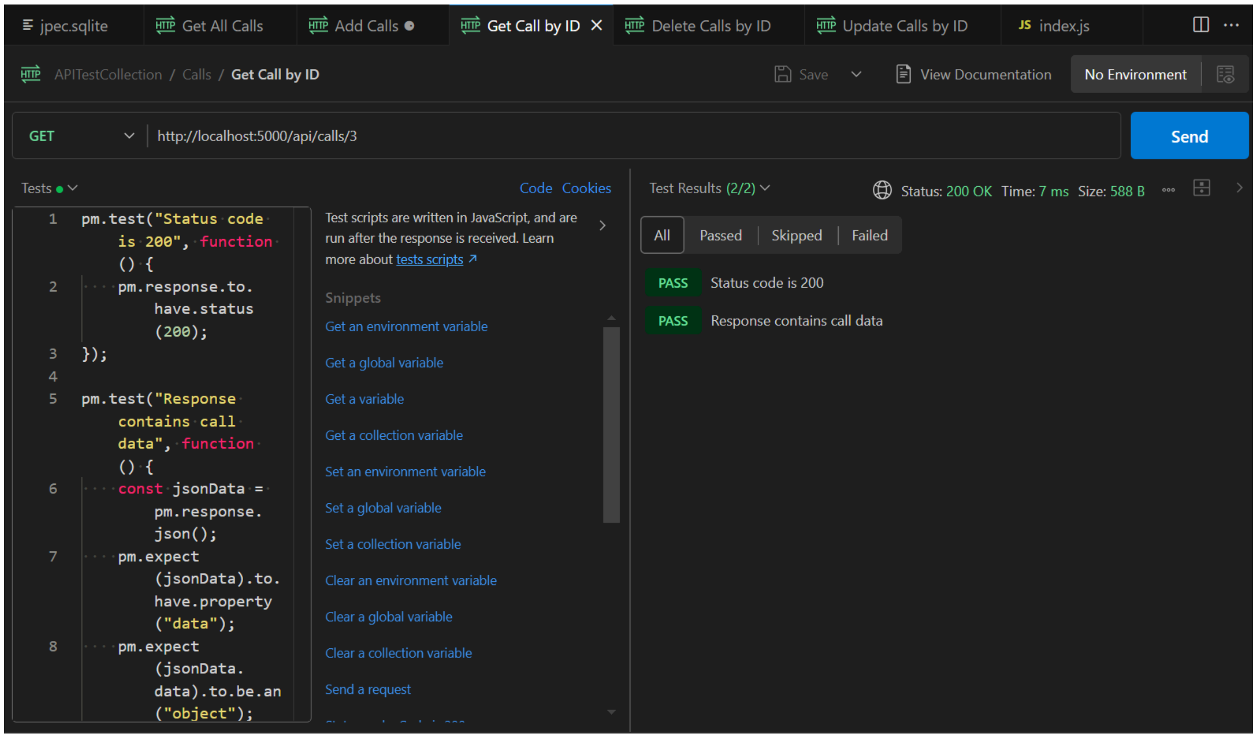
GET http://localhost:5000/api/calls/3

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains call data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data");
    pm.expect(jsonData.data).to.be.an("object");
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Status: 200 OK Time: 7 ms Size: 588 B

```
{
    "data": {
        "id": 3,
        "caller_name": "Tom Lee",
        "caller_address": "789 Pine Rd",
        "call_type": "Gas Leak",
        "crew_assigned": "Crew Gamma",
        "time_called": "2023-10-27 12:45",
        "time_dispatched": "2023-10-27 13:00",
        "time_completed": "2023-10-27 14:00",
        "issue_reported": "Smell of gas near stove",
        "issue_found": "Loose valve",
        "dispatcher_id": 1
    }
}
```



GET http://localhost:5000/api/calls/3

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response contains call data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data");
    pm.expect(jsonData.data).to.be.an("object");
```

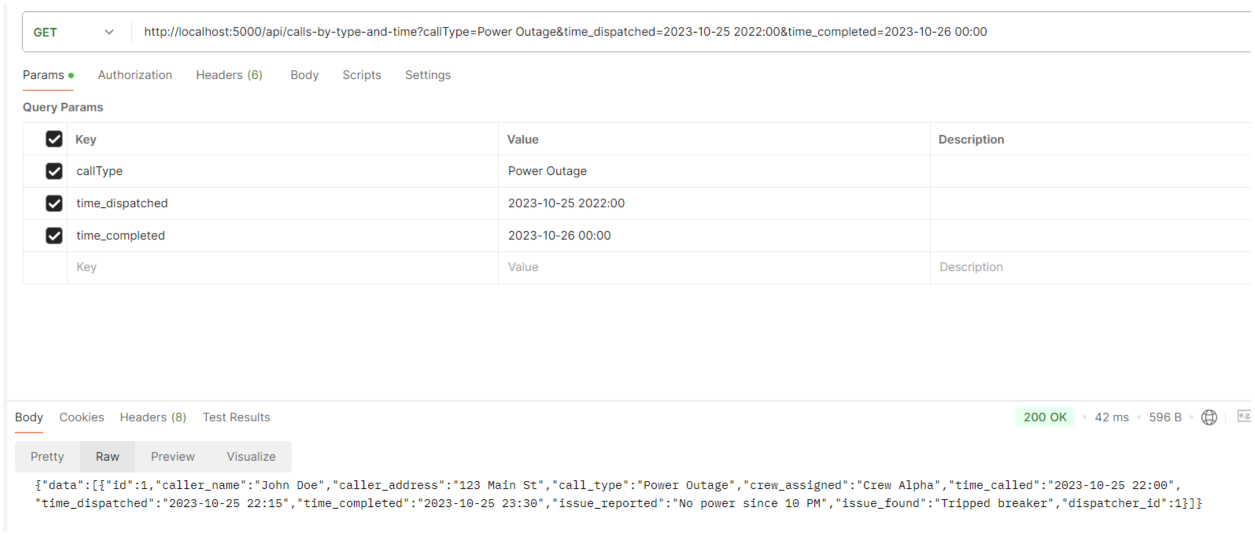
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2)

Status: 200 OK Time: 7 ms Size: 588 B

```
PASS  Status code is 200
PASS  Response contains call data
```

Get Call by Time and Type:



http://localhost:5000/api/calls-by-type-and-time?callType=Power Outage&time_dispatched=2023-10-25 2022:00&time_completed=2023-10-26 00:00

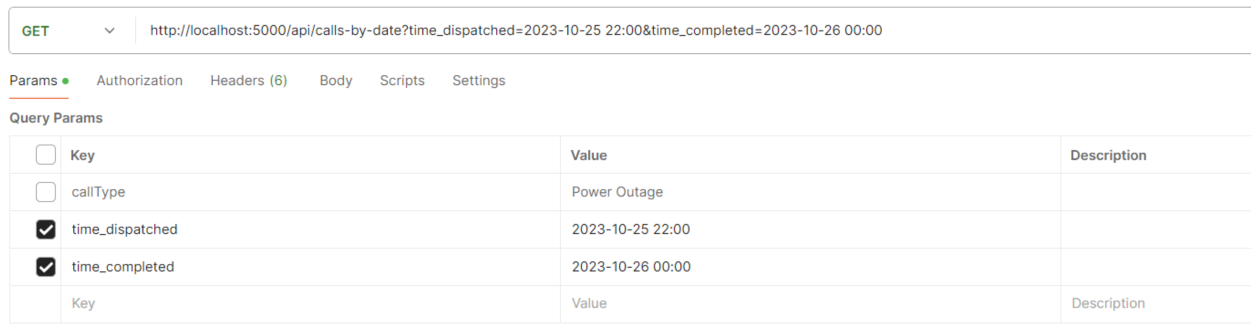
Query Params

	Key	Value	Description
✓	callType	Power Outage	
✓	time_dispatched	2023-10-25 2022:00	
✓	time_completed	2023-10-26 00:00	
	Key	Value	Description

200 OK • 42 ms • 596 B

```
{"data":[{"id":1,"caller_name":"John Doe","caller_address":"123 Main St","call_type":"Power Outage","crew_assigned":"Crew Alpha","time_called":"2023-10-25 22:00",
"time_dispatched":"2023-10-25 22:15","time_completed":"2023-10-25 23:30","issue_reported":"No power since 10 PM","issue_found":"Tripped breaker","dispatcher_id":1}]}
```

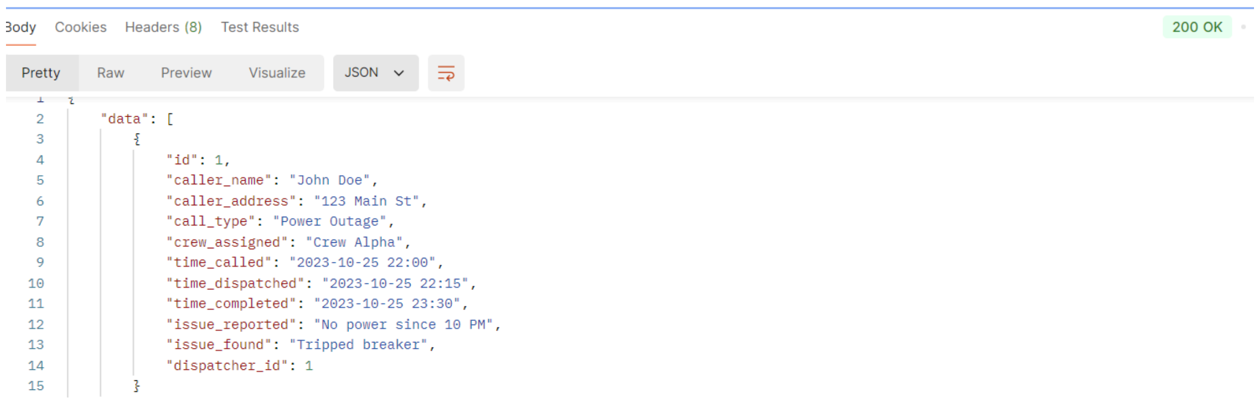
Get Call by Time:



http://localhost:5000/api/calls-by-date?time_dispatched=2023-10-25 22:00&time_completed=2023-10-26 00:00

Query Params

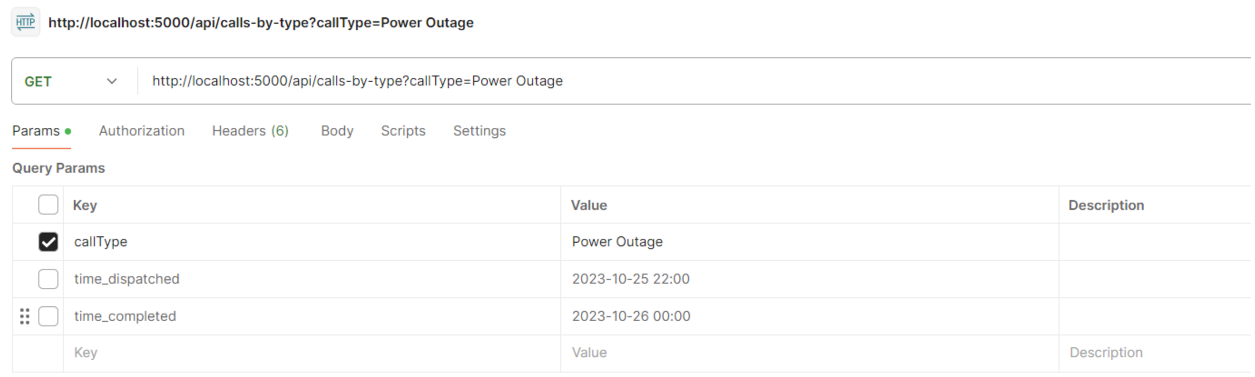
	Key	Value	Description
	callType	Power Outage	
✓	time_dispatched	2023-10-25 22:00	
✓	time_completed	2023-10-26 00:00	
	Key	Value	Description



200 OK

```
{
    "data": [
        {
            "id": 1,
            "caller_name": "John Doe",
            "caller_address": "123 Main St",
            "call_type": "Power Outage",
            "crew_assigned": "Crew Alpha",
            "time_called": "2023-10-25 22:00",
            "time_dispatched": "2023-10-25 22:15",
            "time_completed": "2023-10-25 23:30",
            "issue_reported": "No power since 10 PM",
            "issue_found": "Tripped breaker",
            "dispatcher_id": 1
        }
```

Get Call by Type:

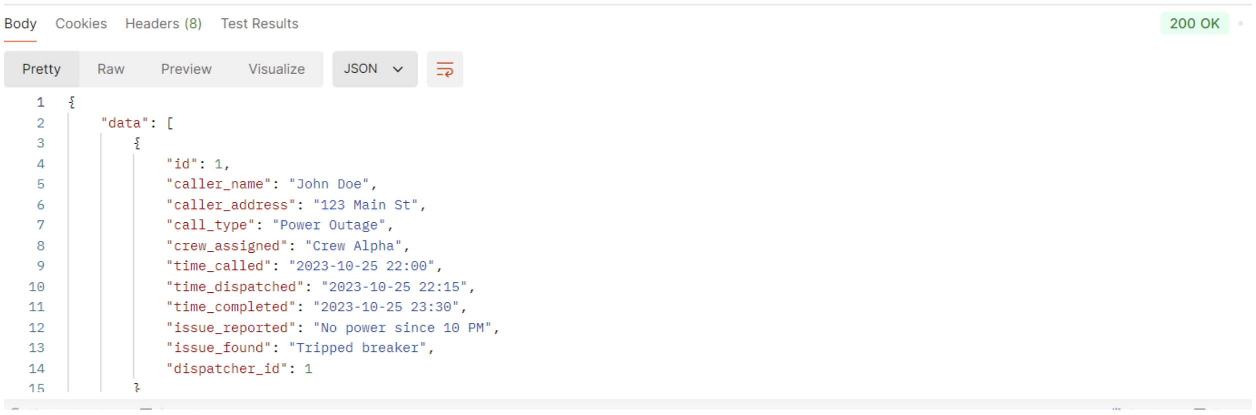


http://localhost:5000/api/calls-by-type?callType=Power Outage

GET http://localhost:5000/api/calls-by-type?callType=Power Outage

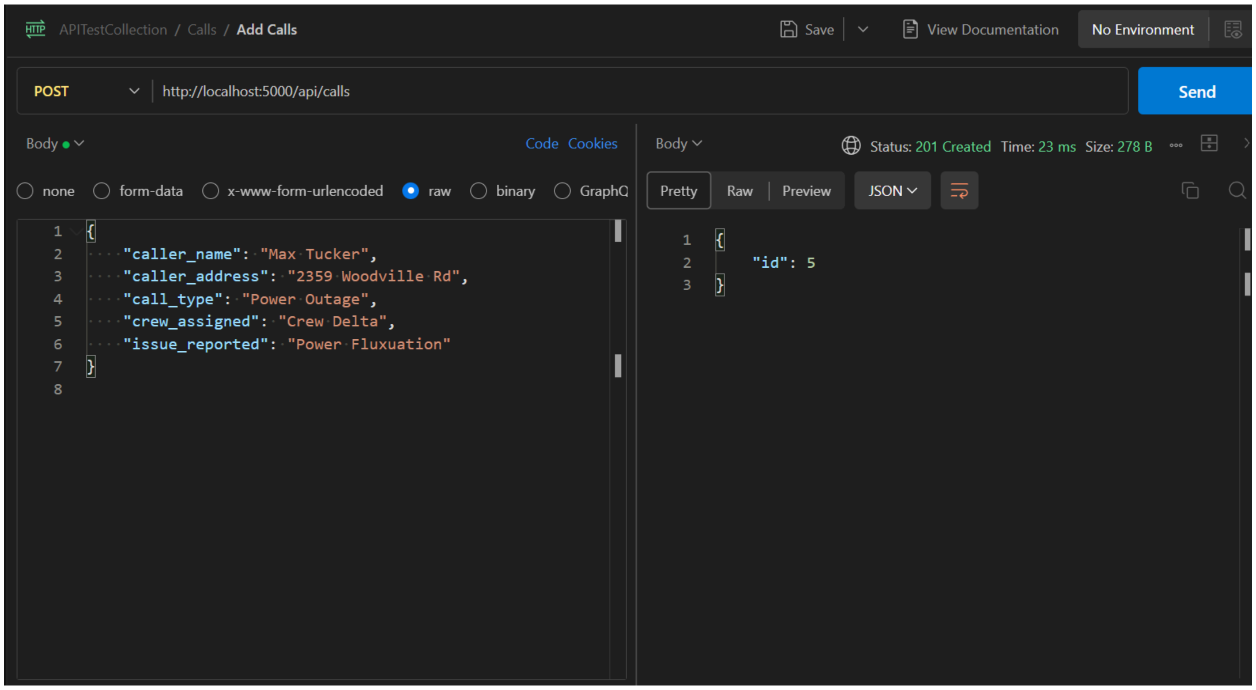
Query Params

	Key	Value	Description
✓	callType	Power Outage	
	time_dispatched	2023-10-25 22:00	
	time_completed	2023-10-26 00:00	
	Key	Value	Description



```
{
    "data": [
        {
            "id": 1,
            "caller_name": "John Doe",
            "caller_address": "123 Main St",
            "call_type": "Power Outage",
            "crew_assigned": "Crew Alpha",
            "time_called": "2023-10-25 22:00",
            "time_dispatched": "2023-10-25 22:15",
            "time_completed": "2023-10-25 23:30",
            "issue_reported": "No power since 10 PM",
            "issue_found": "Tripped breaker",
            "dispatcher_id": 1
        }
```

Add Call:



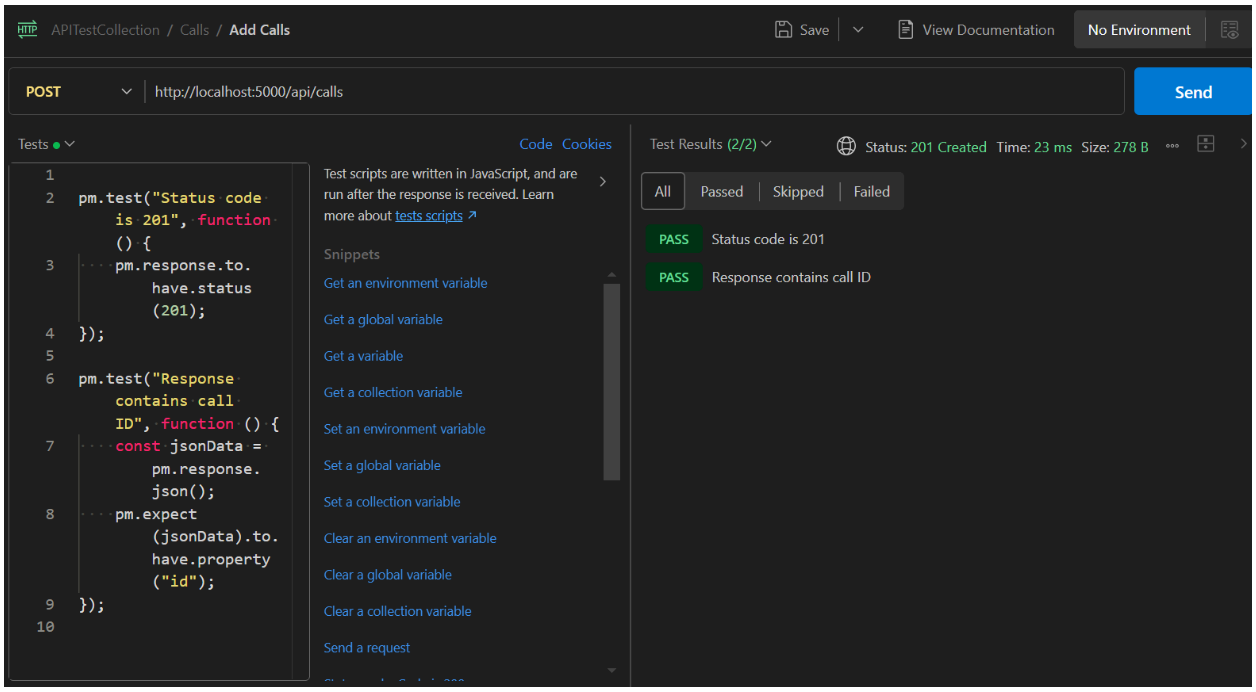
POST http://localhost:5000/api/calls

Body: [✓ raw] (none, form-data, x-www-form-urlencoded, binary, GraphQ)

```
{
    "caller_name": "Max Tucker",
    "caller_address": "2359 Woodville Rd",
    "call_type": "Power Outage",
    "crew_assigned": "Crew Delta",
    "issue_reported": "Power Fluxuation"
}
```

Status: 201 Created Time: 23 ms Size: 278 B

```
{
    "id": 5
}
```


APITestCollection / Calls / Add Calls
Save
View Documentation
No Environment
Send
Tests
Code Cookies
Snippets
Get an environment variable
Get a global variable
Get a variable
Get a collection variable
Set an environment variable
Set a global variable
Set a collection variable
Clear an environment variable
Clear a global variable
Clear a collection variable
Send a request
All Passed Skipped Failed

POST http://localhost:5000/api/calls

```

pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

pm.test("Response contains call ID", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
});

```

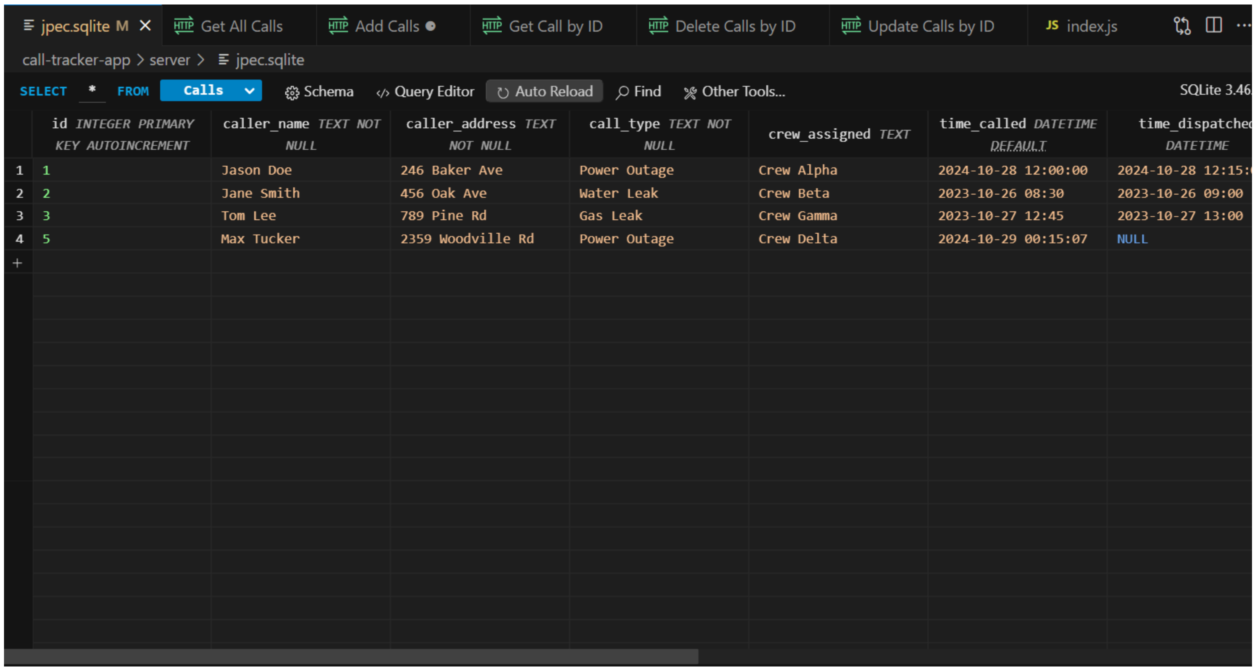
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2)

Status: 201 Created Time: 23 ms Size: 278 B

PASS Status code is 201

PASS Response contains call ID



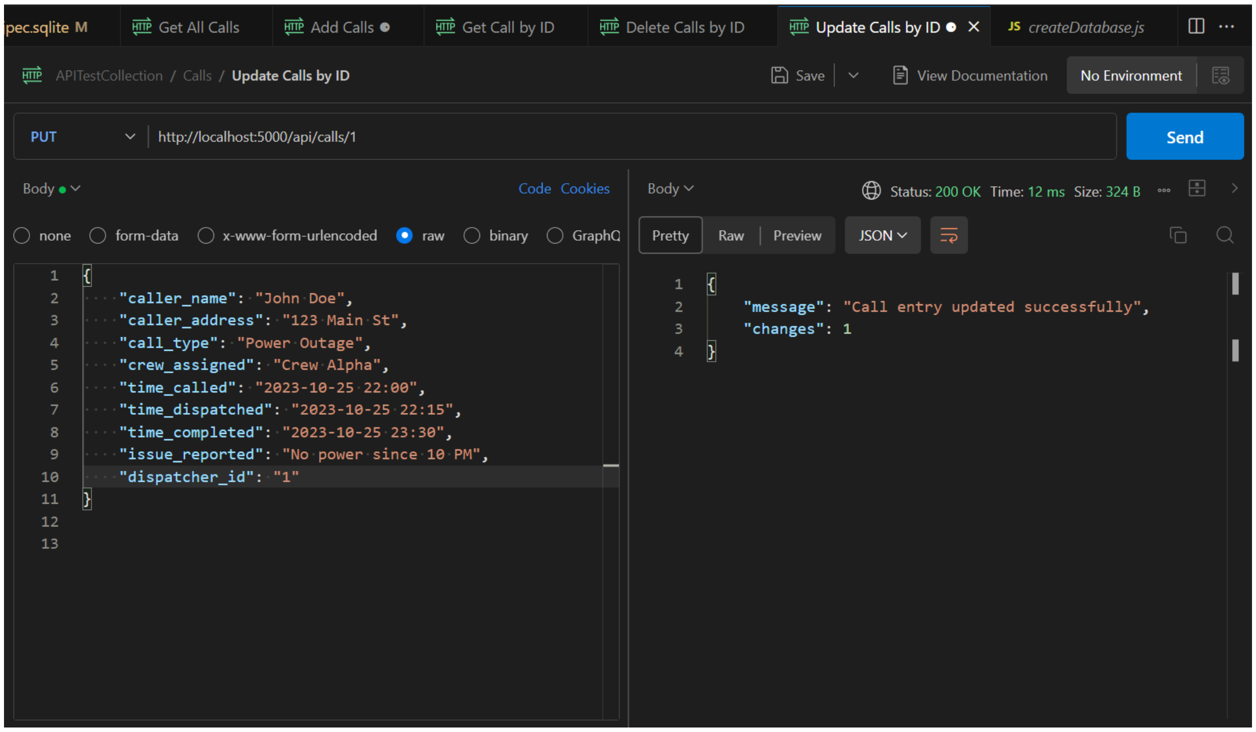
jpec.sqlite M | Get All Calls | Add Calls | Get Call by ID | Delete Calls by ID | Update Calls by ID | index.js

call-tracker-app > server > jpec.sqlite

SELECT * FROM Calls | Schema | Query Editor | Auto Reload | Find | Other Tools... | SQLite 3.46

	id INTEGER PRIMARY KEY AUTOINCREMENT	caller_name TEXT NOT NULL	caller_address TEXT NOT NULL	call_type TEXT NOT NULL	crew_assigned TEXT	time_called DATETIME DEFAULT	time_dispatched DATETIME
1	1	Jason Doe	246 Baker Ave	Power Outage	Crew Alpha	2024-10-28 12:00:00	2024-10-28 12:15:
2	2	Jane Smith	456 Oak Ave	Water Leak	Crew Beta	2023-10-26 08:30	2023-10-26 09:00
3	3	Tom Lee	789 Pine Rd	Gas Leak	Crew Gamma	2023-10-27 12:45	2023-10-27 13:00
4	5	Max Tucker	2359 Woodville Rd	Power Outage	Crew Delta	2024-10-29 00:15:07	NULL

Updating Call by ID:

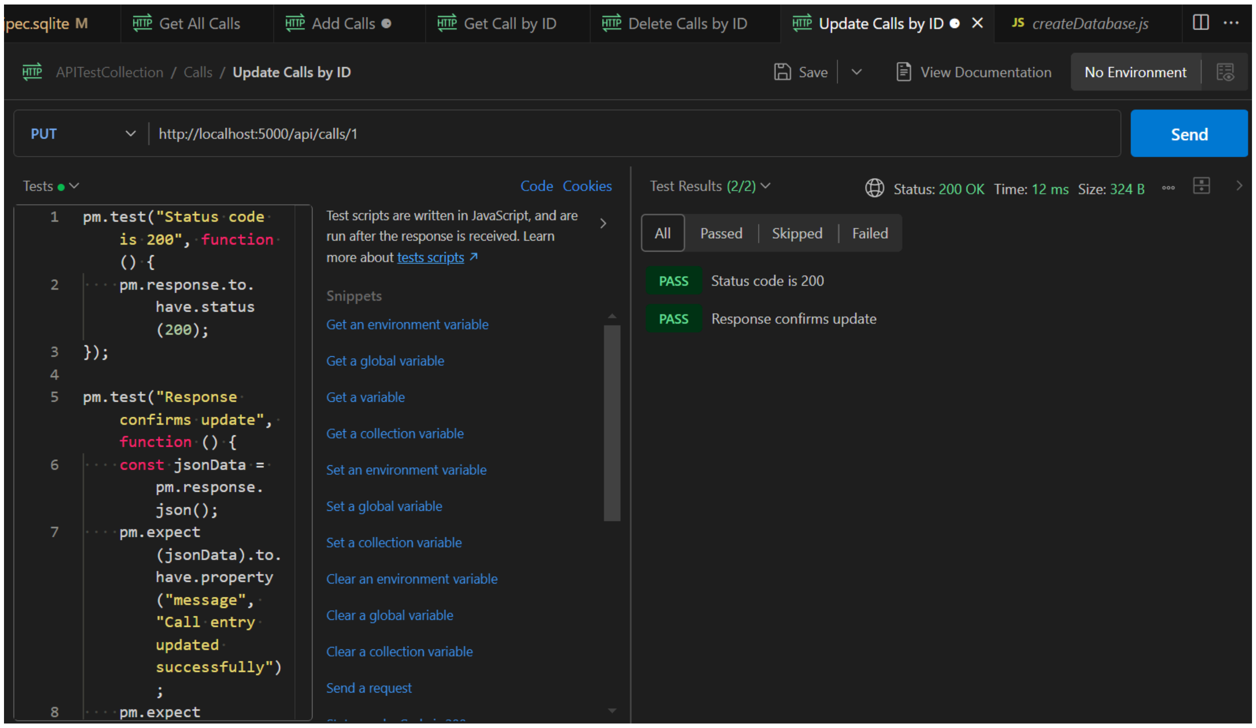


PUT http://localhost:5000/api/calls/1

```
{
    "caller_name": "John Doe",
    "caller_address": "123 Main St",
    "call_type": "Power Outage",
    "crew_assigned": "Crew Alpha",
    "time_called": "2023-10-25 22:00",
    "time_dispatched": "2023-10-25 22:15",
    "time_completed": "2023-10-25 23:30",
    "issue_reported": "No power since 10 PM",
    "dispatcher_id": "1"
}
```

Status: 200 OK Time: 12 ms Size: 324 B

```
{
    "message": "Call entry updated successfully",
    "changes": 1
}
```



PUT http://localhost:5000/api/calls/1

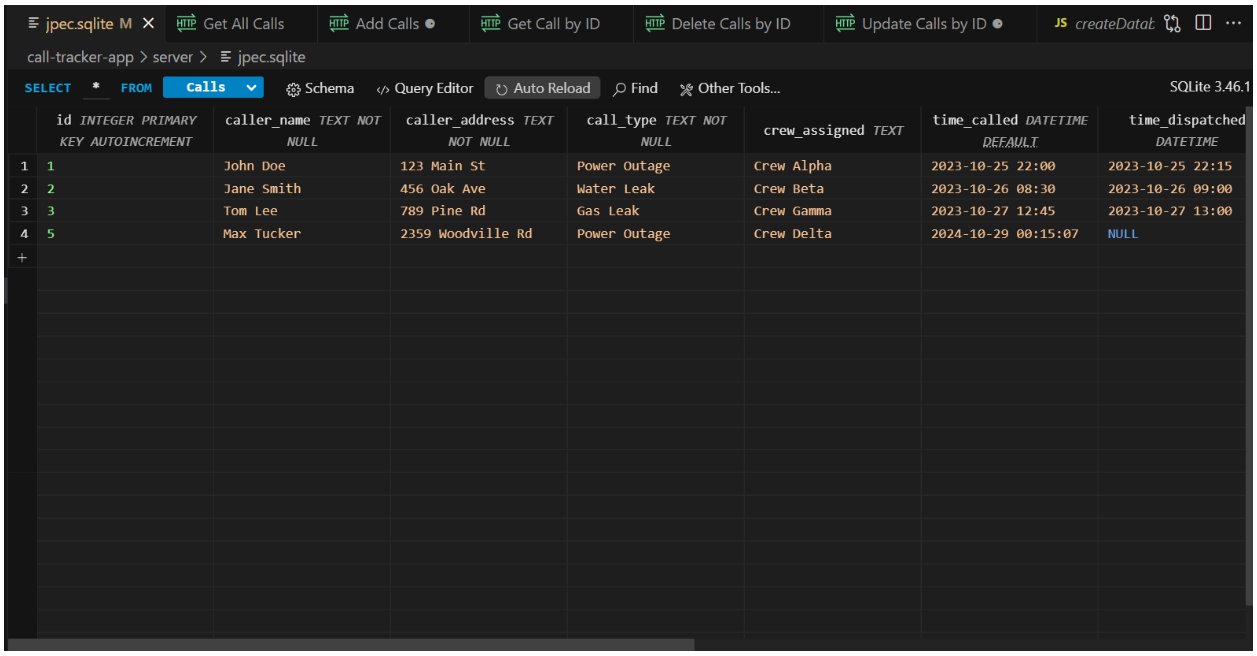
```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response confirms update", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "Call entry updated successfully");
    pm.expect
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2)

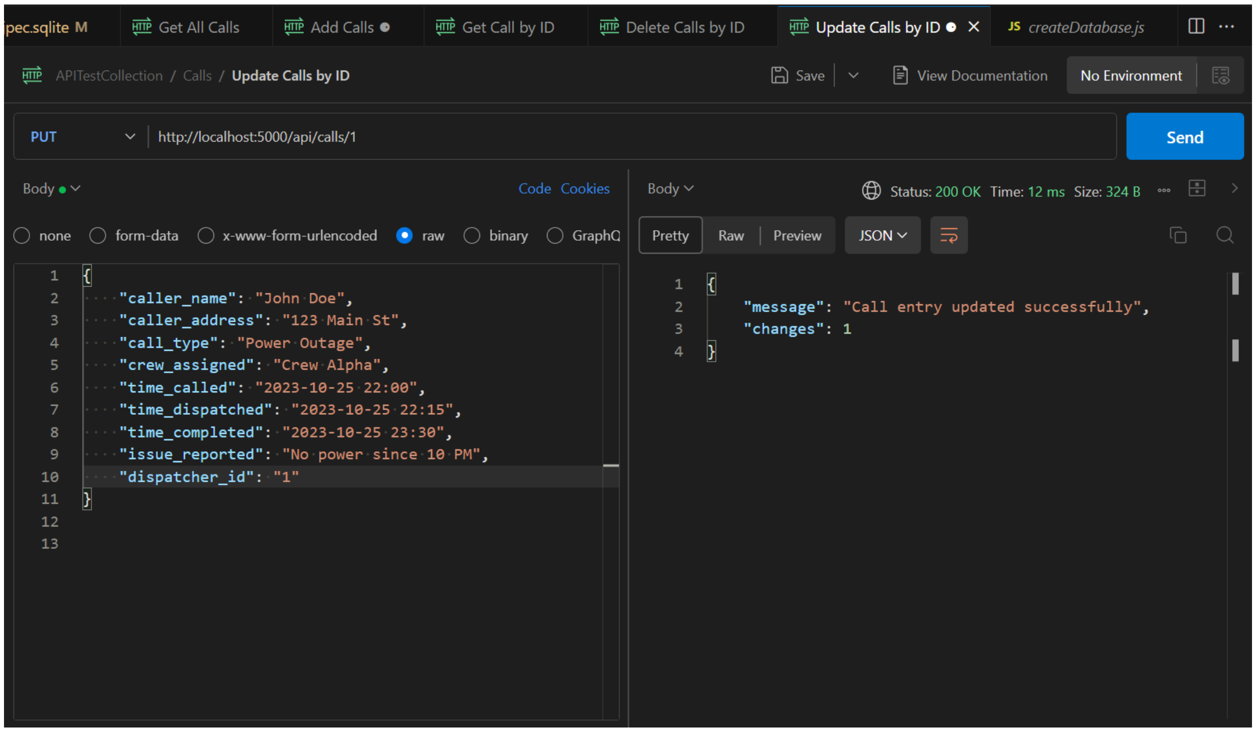
Status: 200 OK Time: 12 ms Size: 324 B

```
PASS   Status code is 200
PASS   Response confirms update
```



	id INTEGER PRIMARY KEY AUTOINCREMENT	caller_name TEXT NOT NULL	caller_address TEXT NOT NULL	call_type TEXT NOT NULL	crew_assigned TEXT	time_called DATETIME DEFAULT	time_dispatched DATETIME
1	1	John Doe	123 Main St	Power Outage	Crew Alpha	2023-10-25 22:00	2023-10-25 22:15
2	2	Jane Smith	456 Oak Ave	Water Leak	Crew Beta	2023-10-26 08:30	2023-10-26 09:00
3	3	Tom Lee	789 Pine Rd	Gas Leak	Crew Gamma	2023-10-27 12:45	2023-10-27 13:00
4	5	Max Tucker	2359 Woodville Rd	Power Outage	Crew Delta	2024-10-29 00:15:07	NULL

Delete Call by ID:



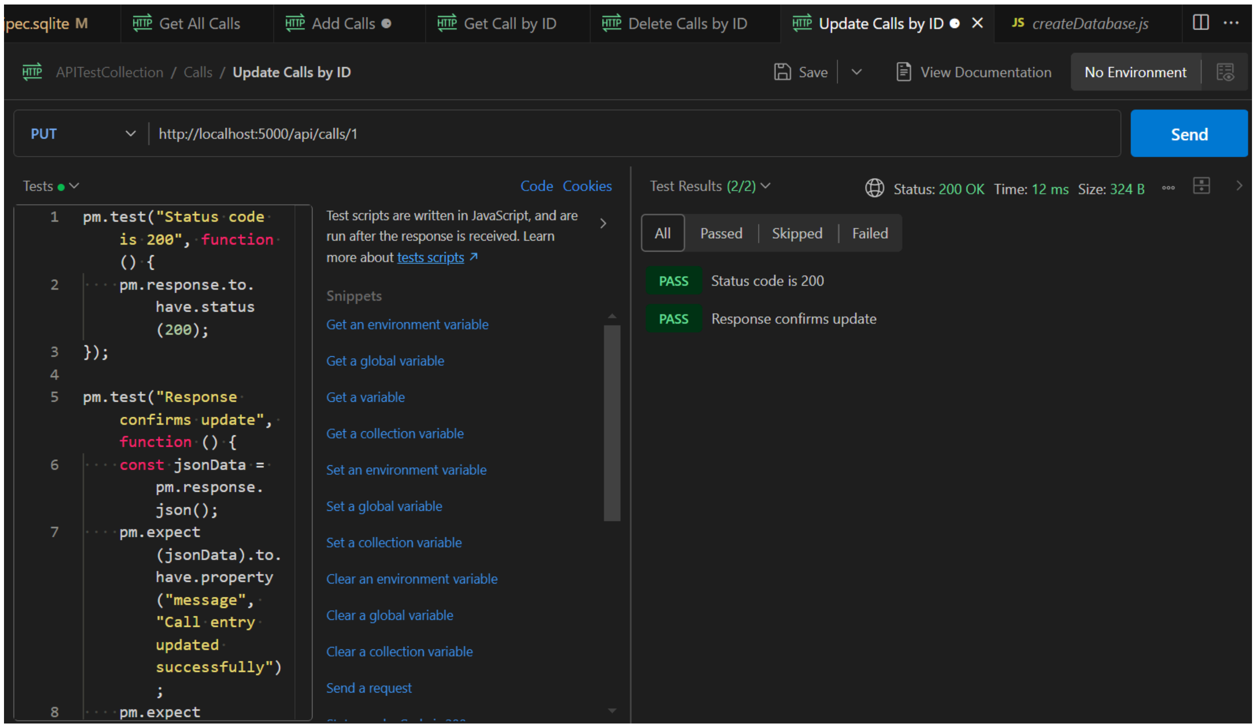
PUT http://localhost:5000/api/calls/1

Body: none, form-data, x-www-form-urlencoded, [✓ raw], binary, GraphQ

```
{
    "caller_name": "John Doe",
    "caller_address": "123 Main St",
    "call_type": "Power Outage",
    "crew_assigned": "Crew Alpha",
    "time_called": "2023-10-25 22:00",
    "time_dispatched": "2023-10-25 22:15",
    "time_completed": "2023-10-25 23:30",
    "issue_reported": "No power since 10 PM",
    "dispatcher_id": "1"
}
```

Status: 200 OK Time: 12 ms Size: 324 B

```
{
    "message": "Call entry updated successfully",
    "changes": 1
}
```



PUT http://localhost:5000/api/calls/1

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response confirms update", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "Call entry updated successfully");
    pm.expect
```

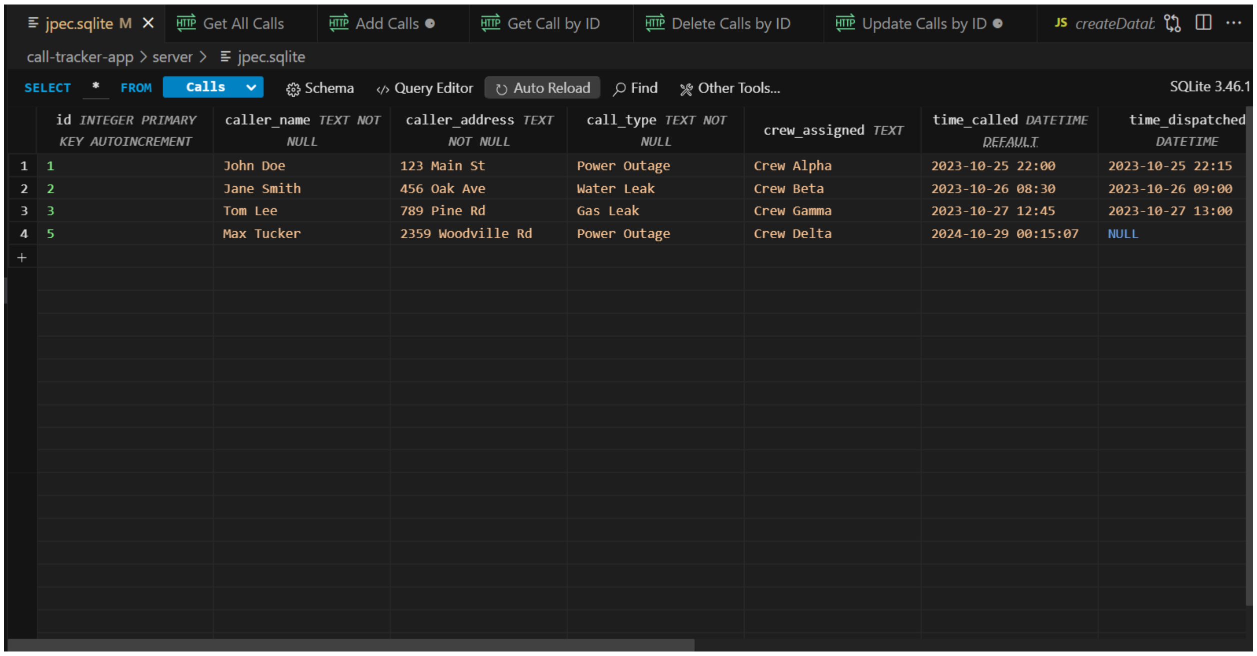
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2)

Status: 200 OK Time: 12 ms Size: 324 B

PASS Status code is 200

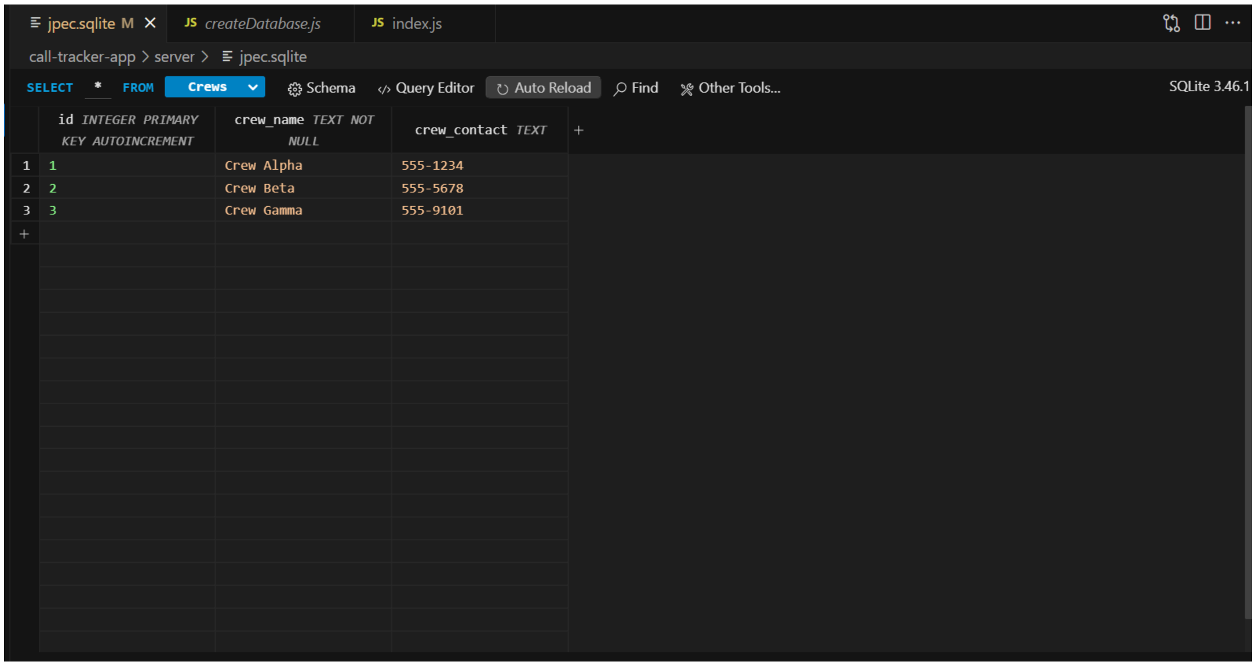
PASS Response confirms update



	id INTEGER PRIMARY KEY AUTOINCREMENT	caller_name TEXT NOT NULL	caller_address TEXT NOT NULL	call_type TEXT NOT NULL	crew_assigned TEXT	time_called DATETIME DEFAULT	time_dispatched DATETIME
1	1	John Doe	123 Main St	Power Outage	Crew Alpha	2023-10-25 22:00	2023-10-25 22:15
2	2	Jane Smith	456 Oak Ave	Water Leak	Crew Beta	2023-10-26 08:30	2023-10-26 09:00
3	3	Tom Lee	789 Pine Rd	Gas Leak	Crew Gamma	2023-10-27 12:45	2023-10-27 13:00
4	5	Max Tucker	2359 Woodville Rd	Power Outage	Crew Delta	2024-10-29 00:15:07	NULL

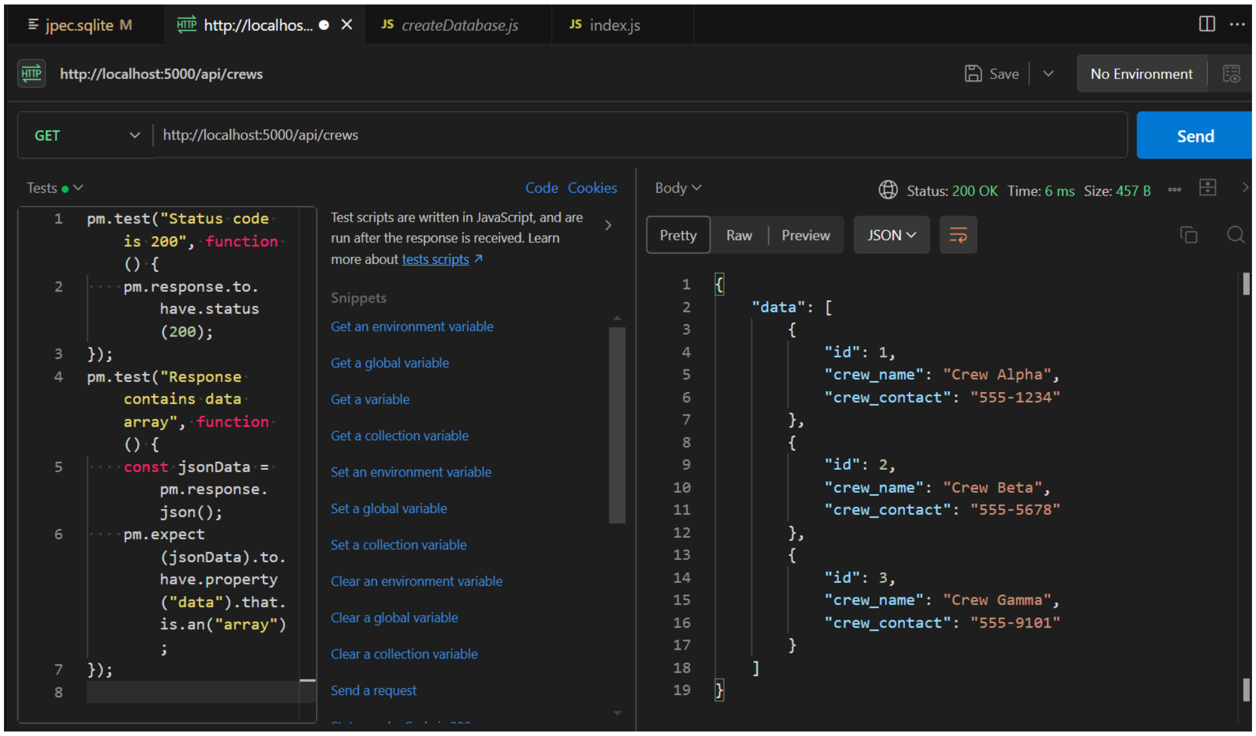
Crews

Crew table Starting Out:



	id INTEGER PRIMARY KEY AUTOINCREMENT	crew_name TEXT NOT NULL	crew_contact TEXT
1	1	Crew Alpha	555-1234
2	2	Crew Beta	555-5678
3	3	Crew Gamma	555-9101

Getting All Crews:



http://localhost:5000/api/crews

GET http://localhost:5000/api/crews

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains data array", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("array");
});
```

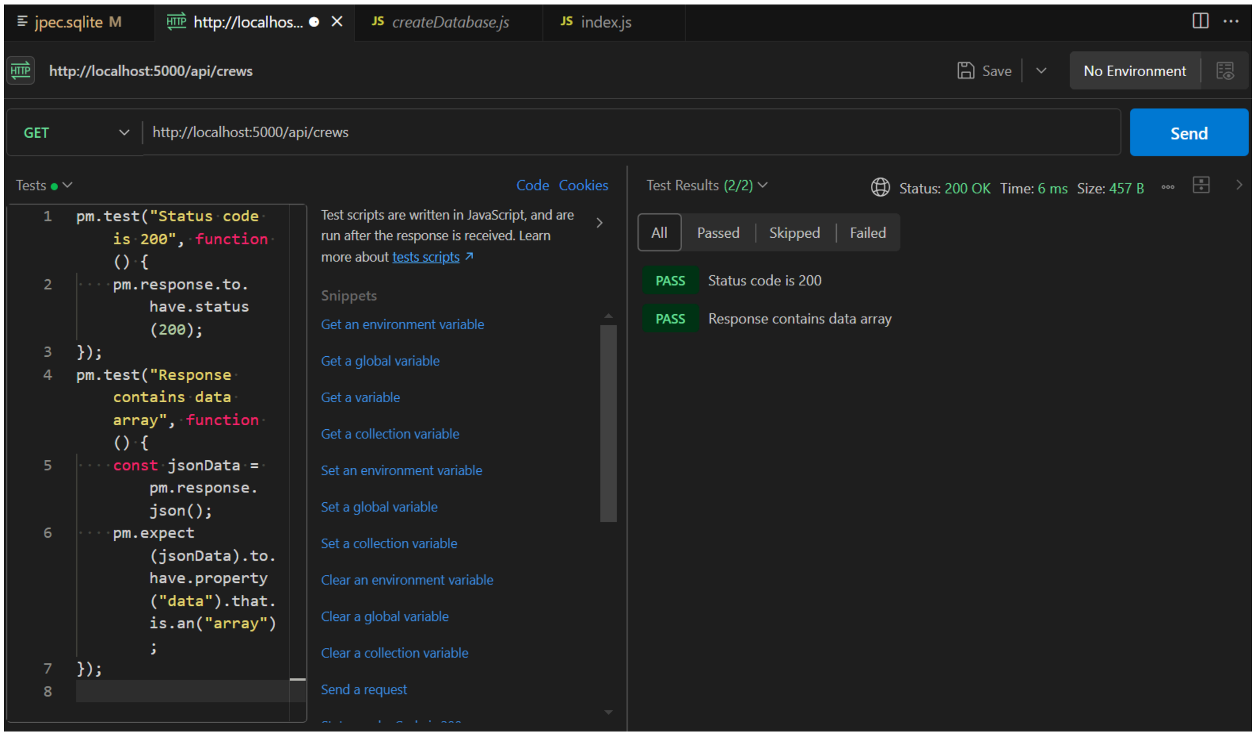
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request

Status: 200 OK Time: 6 ms Size: 457 B

```
{
    "data": [
        {
            "id": 1,
            "crew_name": "Crew Alpha",
            "crew_contact": "555-1234"
        },
        {
            "id": 2,
            "crew_name": "Crew Beta",
            "crew_contact": "555-5678"
        },
        {
            "id": 3,
            "crew_name": "Crew Gamma",
            "crew_contact": "555-9101"
        }
    ]
}
```



http://localhost:5000/api/crews

GET http://localhost:5000/api/crews

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains data array", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("array");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Snippets

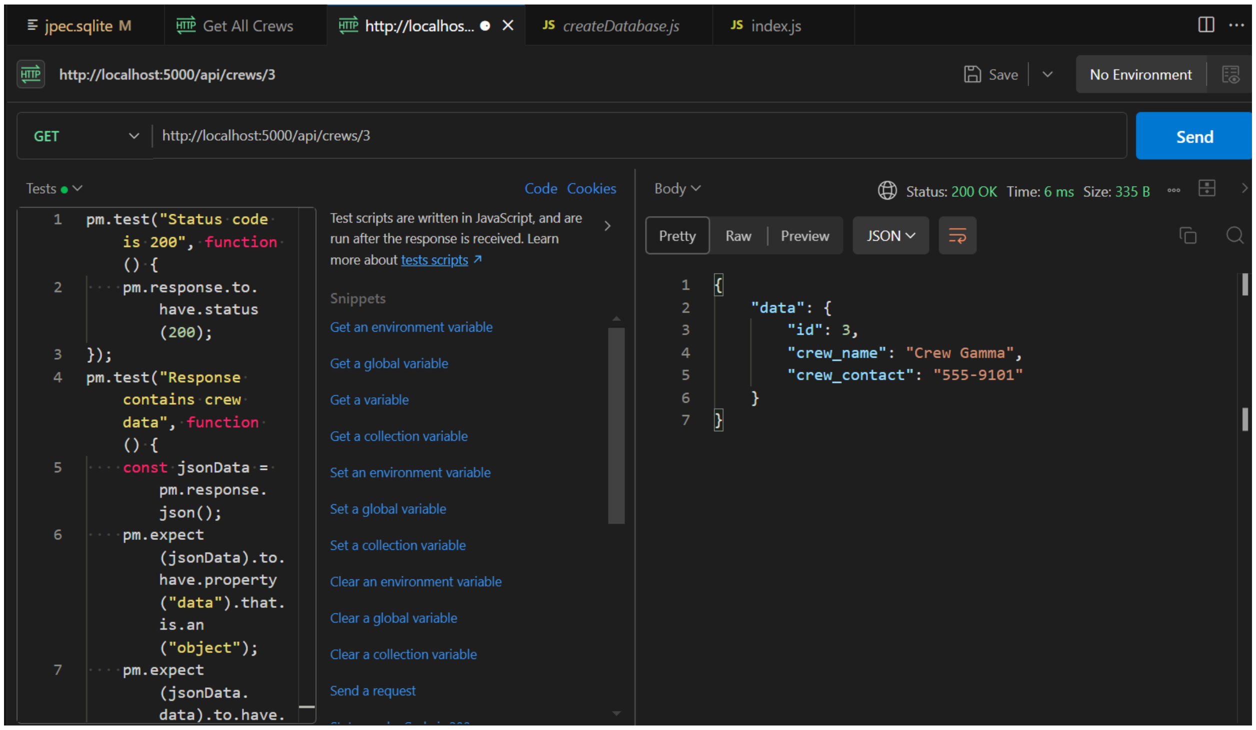
- Get an environment variable
- Get a global variable
- Get a variable
- Get a collection variable
- Set an environment variable
- Set a global variable
- Set a collection variable
- Clear an environment variable
- Clear a global variable
- Clear a collection variable
- Send a request

Test Results (2/2)

Status: 200 OK Time: 6 ms Size: 457 B

```
PASS  Status code is 200
PASS  Response contains data array
```

Get Crew by ID:



jpec.sqlite M | Get All Crews | http://localhos... | createDatabase.js | index.js

http://localhost:5000/api/crews/3

GET http://localhost:5000/api/crews/3

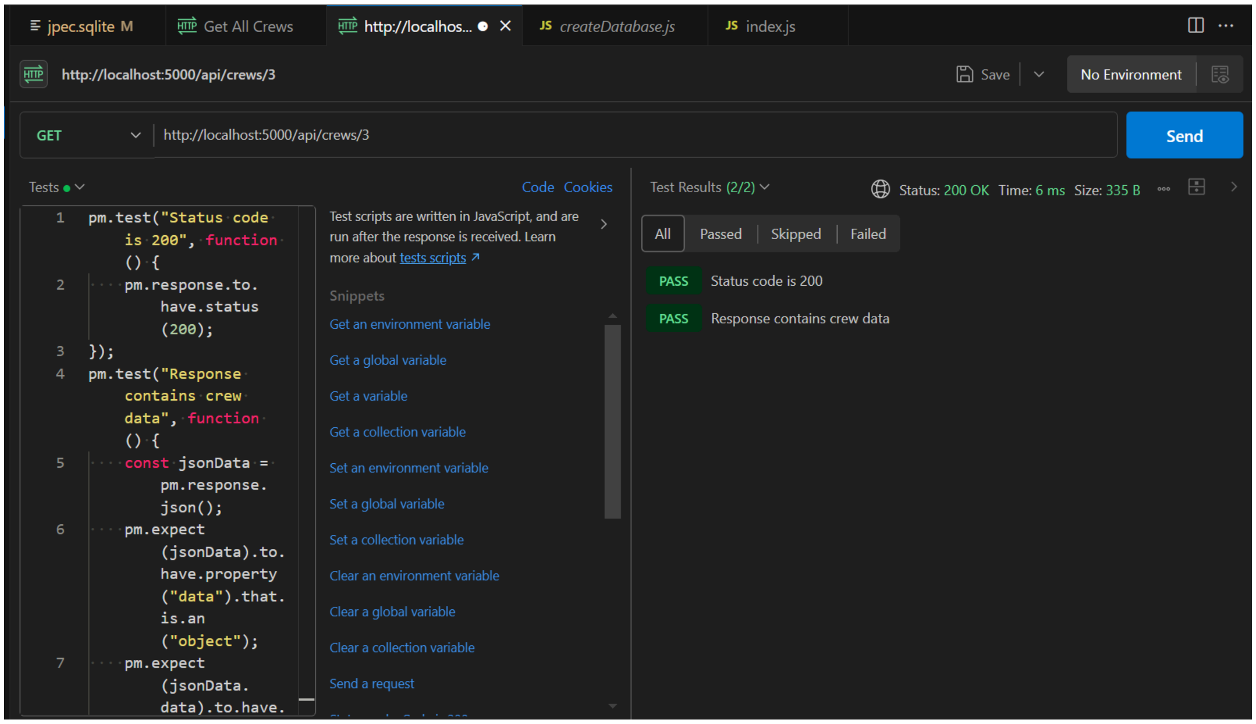
Tests

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains crew data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("object");
    pm.expect(jsonData.data).to.have.
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts

Status: 200 OK Time: 6 ms Size: 335 B

```
{
    "data": {
        "id": 3,
        "crew_name": "Crew Gamma",
        "crew_contact": "555-9101"
    }
}
```



http://localhost:5000/api/crews/3

GET http://localhost:5000/api/crews/3

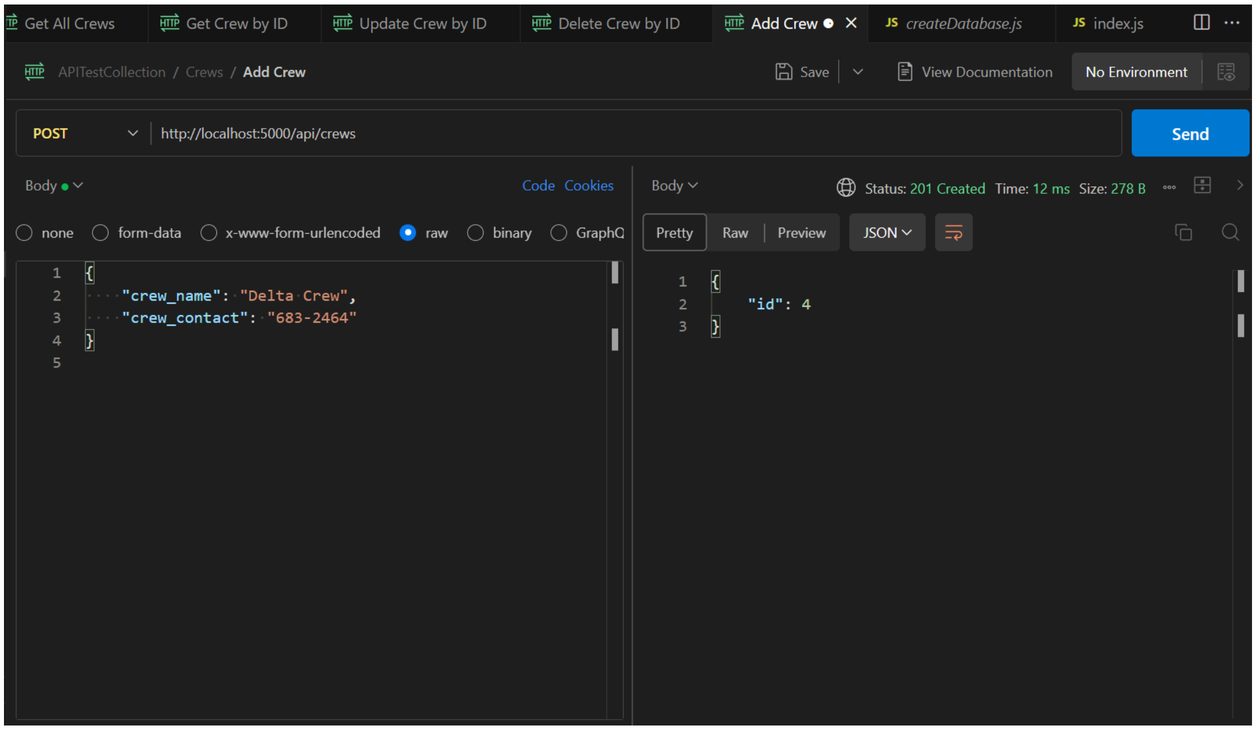
Test Results (2/2)

Status: 200 OK Time: 6 ms Size: 335 B

PASS Status code is 200

PASS Response contains crew data

Add Crews:

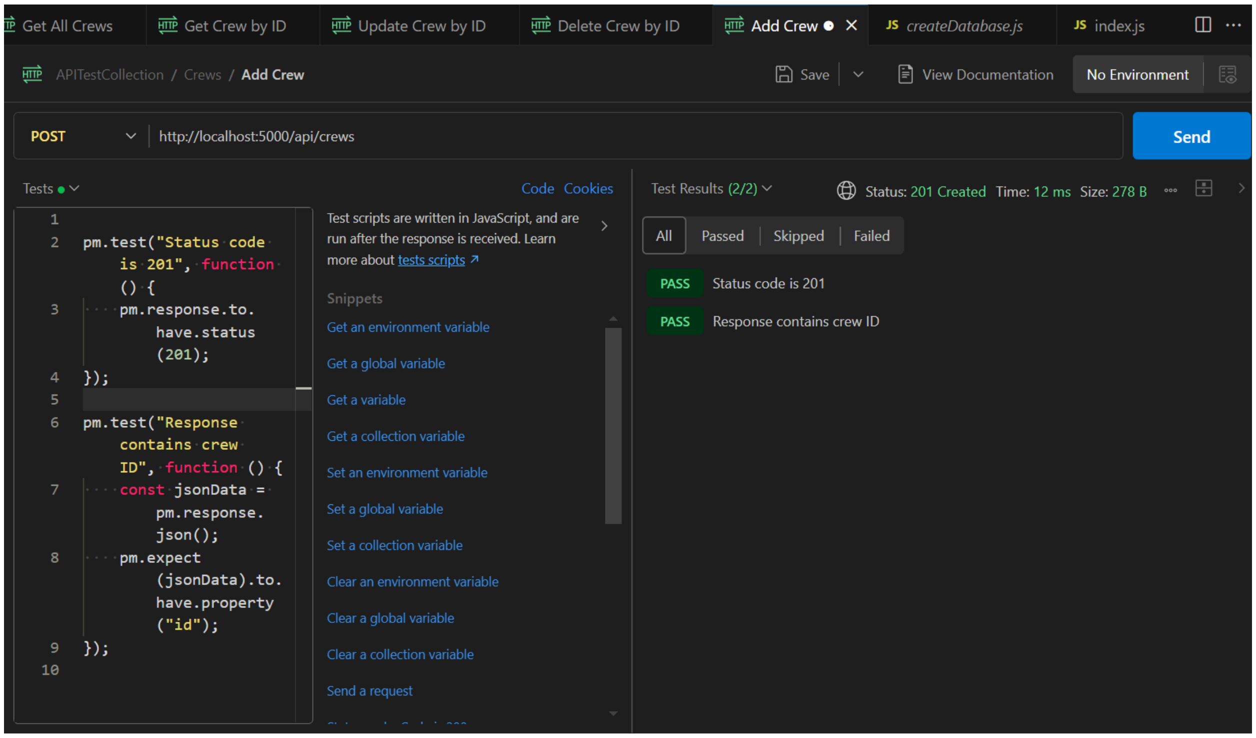


POST http://localhost:5000/api/crews

```
{
    "crew_name": "Delta Crew",
    "crew_contact": "683-2464"
}
```

Status: 201 Created Time: 12 ms Size: 278 B

```
{
    "id": 4
}
```



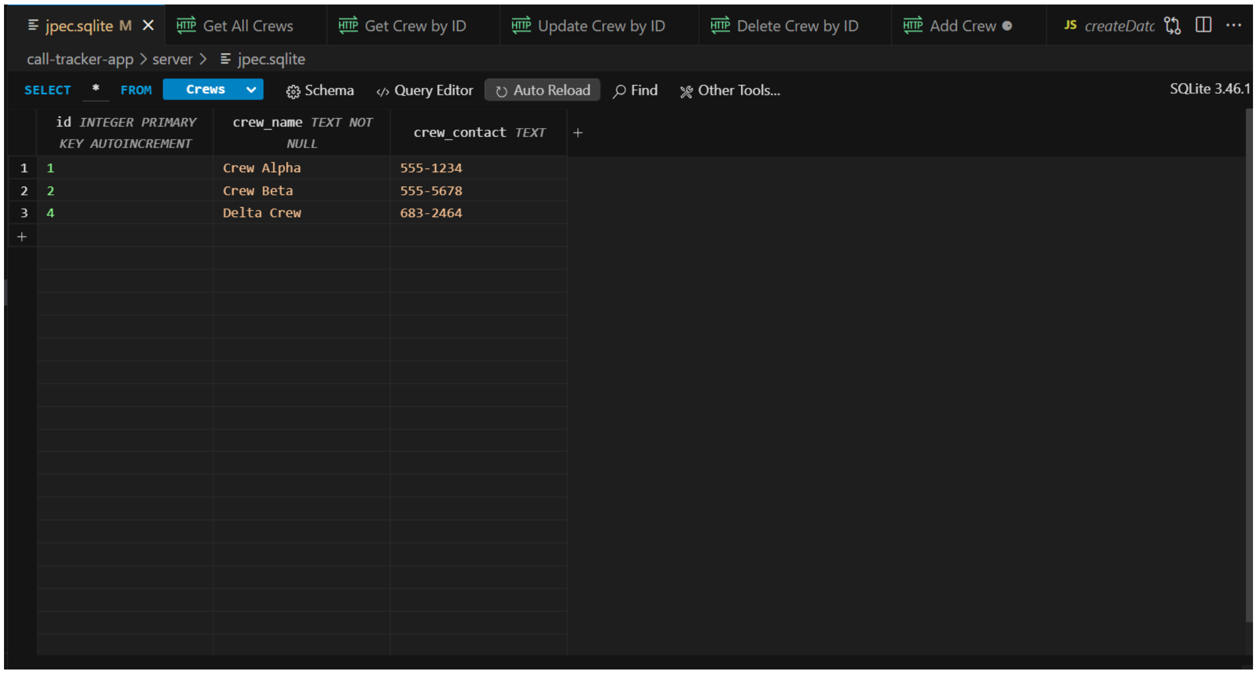
POST http://localhost:5000/api/crews

```

pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});

pm.test("Response contains crew ID", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
});

```

Test Results (2/2) Status: 201 Created Time: 12 ms Size: 278 B

PASS Status code is 201

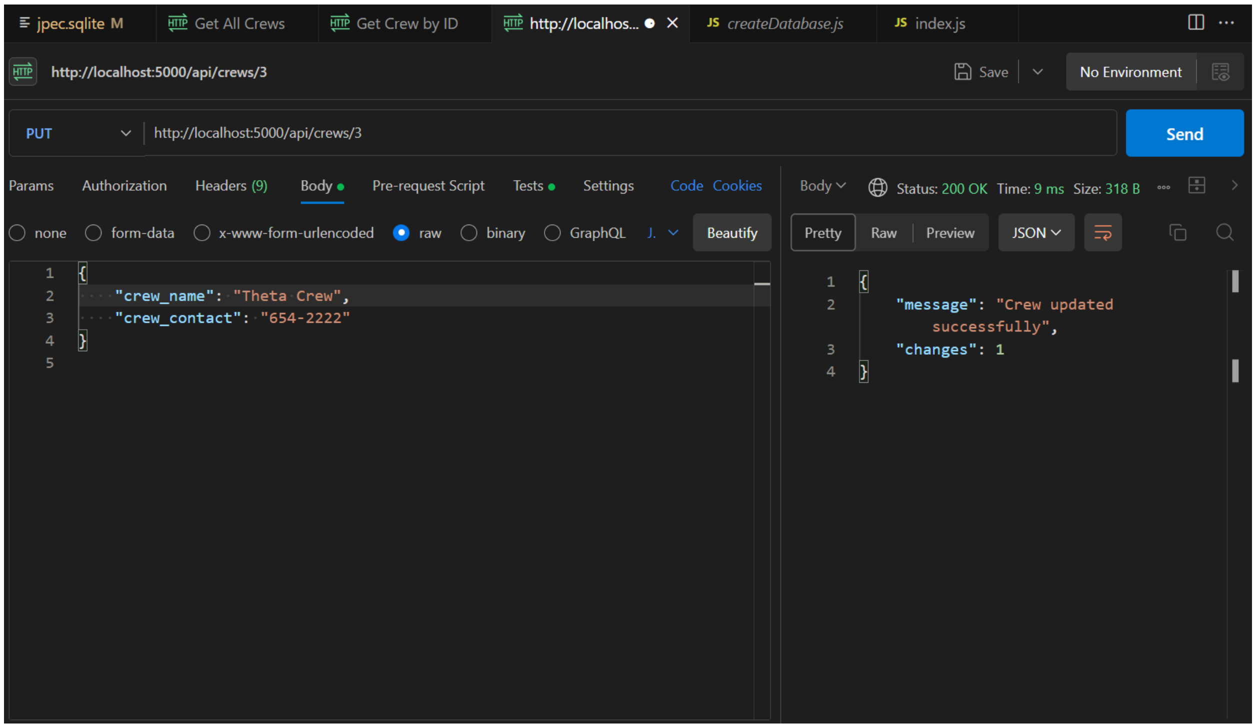
PASS Response contains crew ID



SELECT * FROM Crews

	id *INTEGER PRIMARY KEY AUTOINCREMENT*	crew_name *TEXT NOT NULL*	crew_contact *TEXT*
1	1	Crew Alpha	555-1234
2	2	Crew Beta	555-5678
3	4	Delta Crew	683-2464

Updating Crew by ID:



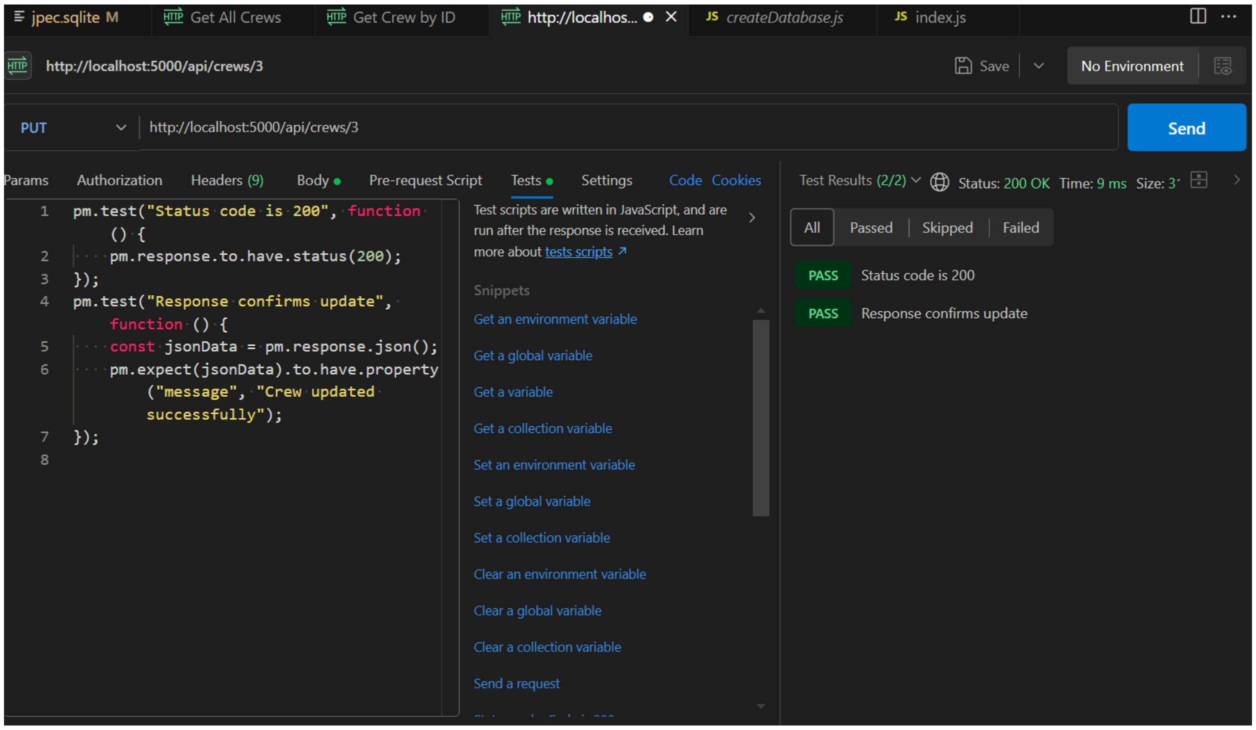
http://localhost:5000/api/crews/3

PUT http://localhost:5000/api/crews/3

```
{
    "crew_name": "Theta Crew",
    "crew_contact": "654-2222"
}
```

Status: 200 OK Time: 9 ms Size: 318 B

```
{
    "message": "Crew updated successfully",
    "changes": 1
}
```



http://localhost:5000/api/crews/3

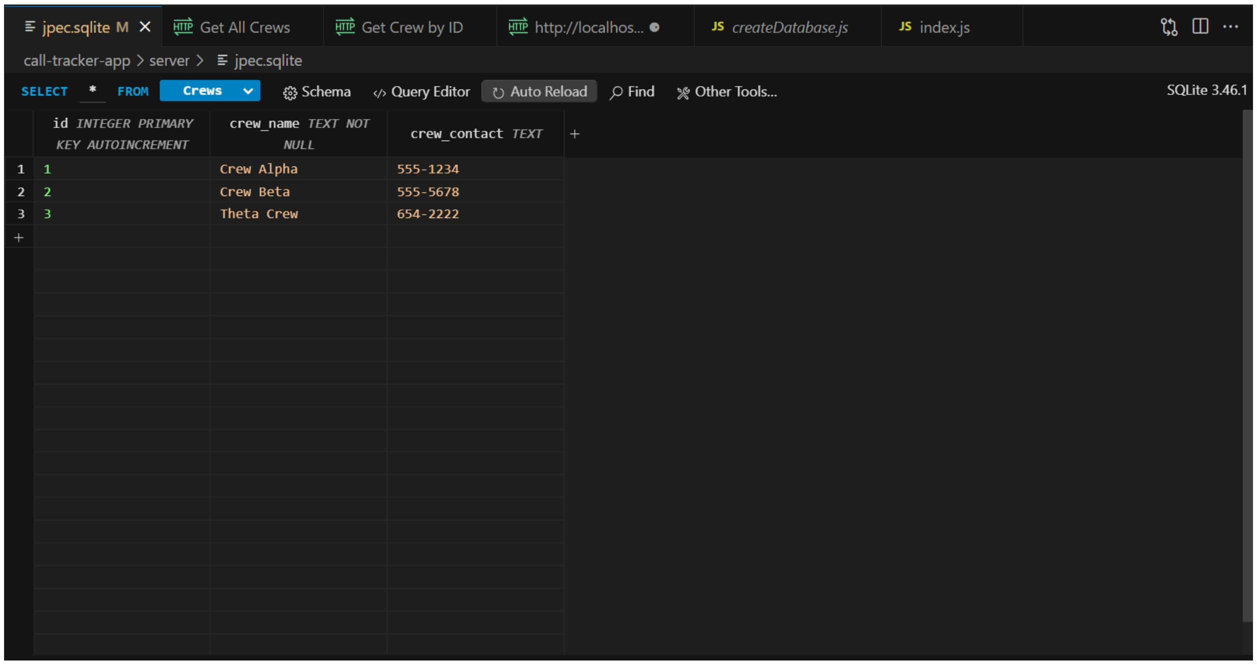
PUT http://localhost:5000/api/crews/3

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response confirms update", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "Crew updated successfully");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

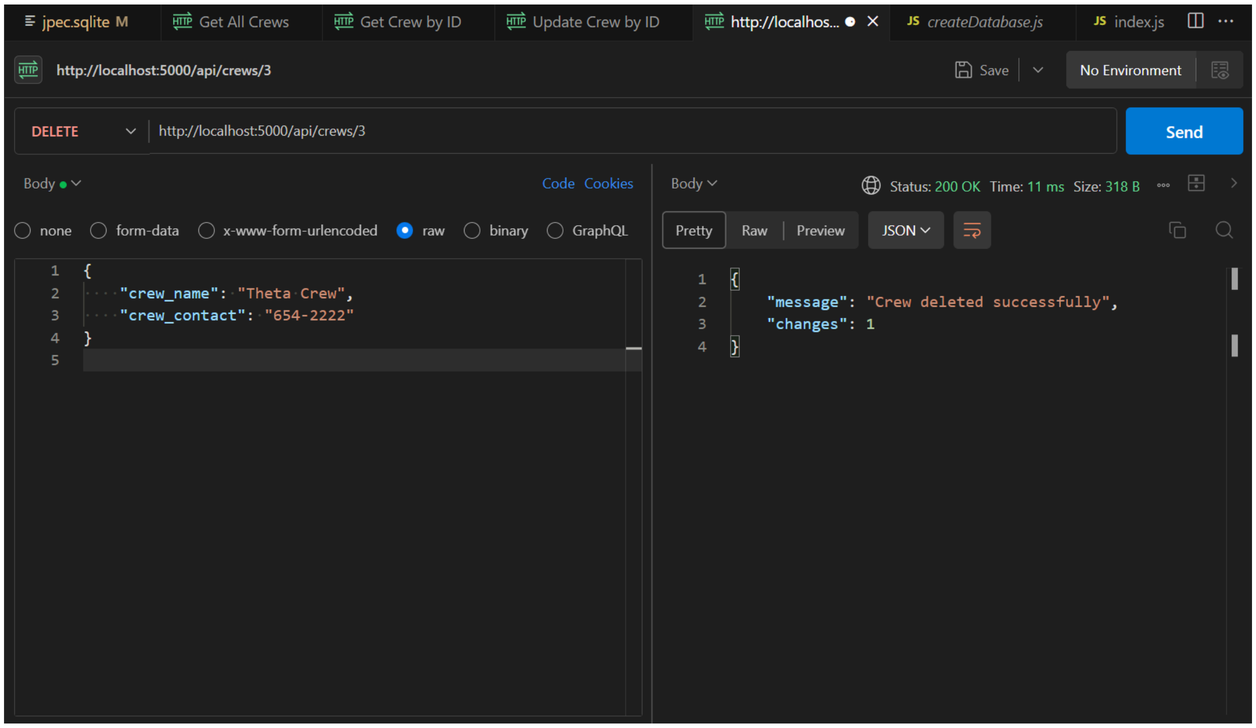
Test Results (2/2) Status: 200 OK Time: 9 ms Size: 3

```
PASS Status code is 200
PASS Response confirms update
```



	id INTEGER PRIMARY KEY AUTOINCREMENT	crew_name TEXT NOT NULL	crew_contact TEXT
1	1	Crew Alpha	555-1234
2	2	Crew Beta	555-5678
3	3	Theta Crew	654-2222

Delete Crew by ID:



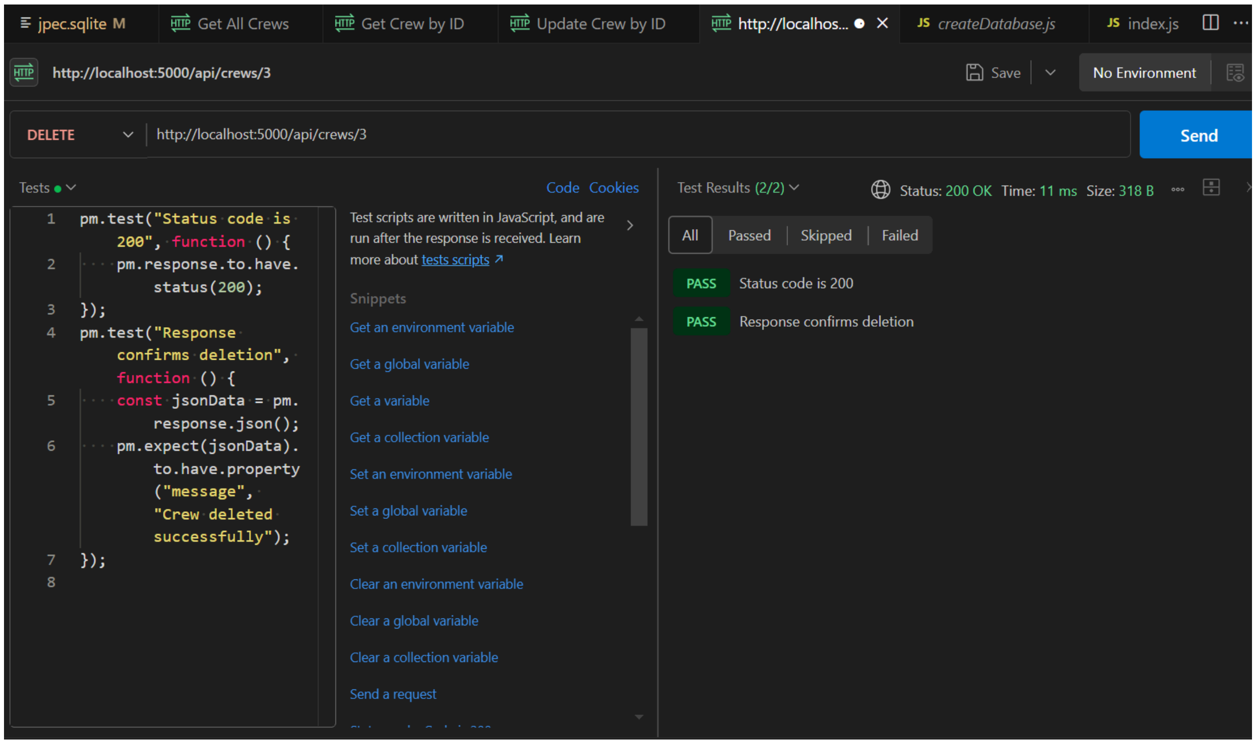
http://localhost:5000/api/crews/3

DELETE http://localhost:5000/api/crews/3

```
{
    "crew_name": "Theta Crew",
    "crew_contact": "654-2222"
}
```

Status: 200 OK Time: 11 ms Size: 318 B

```
{
    "message": "Crew deleted successfully",
    "changes": 1
}
```



http://localhost:5000/api/crews/3

DELETE http://localhost:5000/api/crews/3

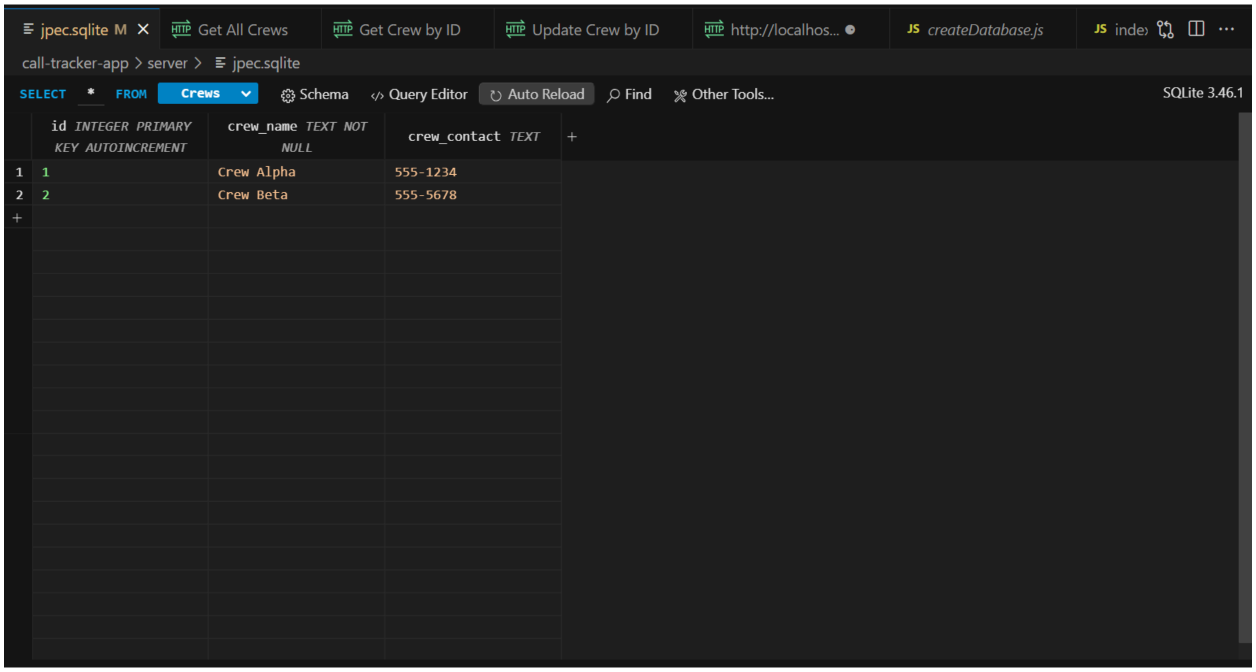
```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response confirms deletion", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "Crew deleted successfully");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2) Status: 200 OK Time: 11 ms Size: 318 B

PASS Status code is 200

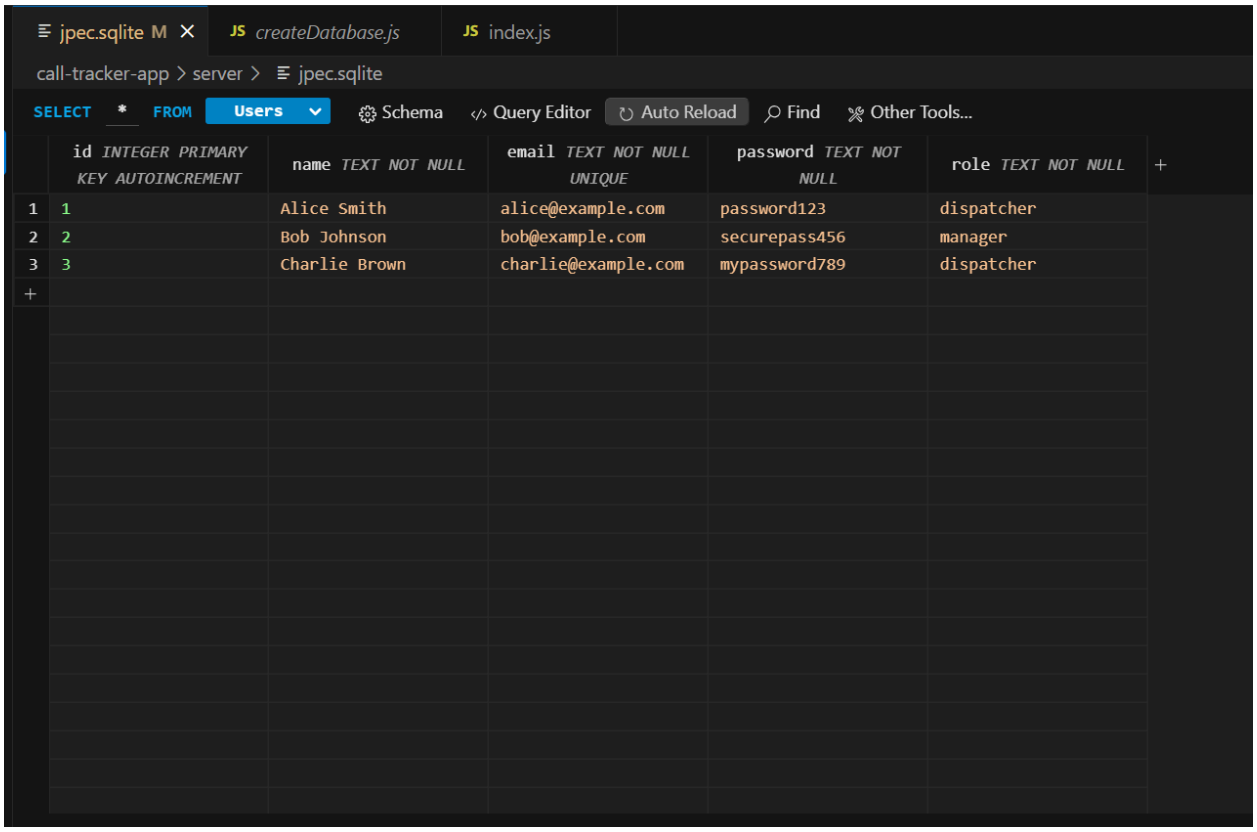
PASS Response confirms deletion



	id INTEGER PRIMARY KEY AUTOINCREMENT	crew_name TEXT NOT NULL	crew_contact TEXT
1	1	Crew Alpha	555-1234
2	2	Crew Beta	555-5678

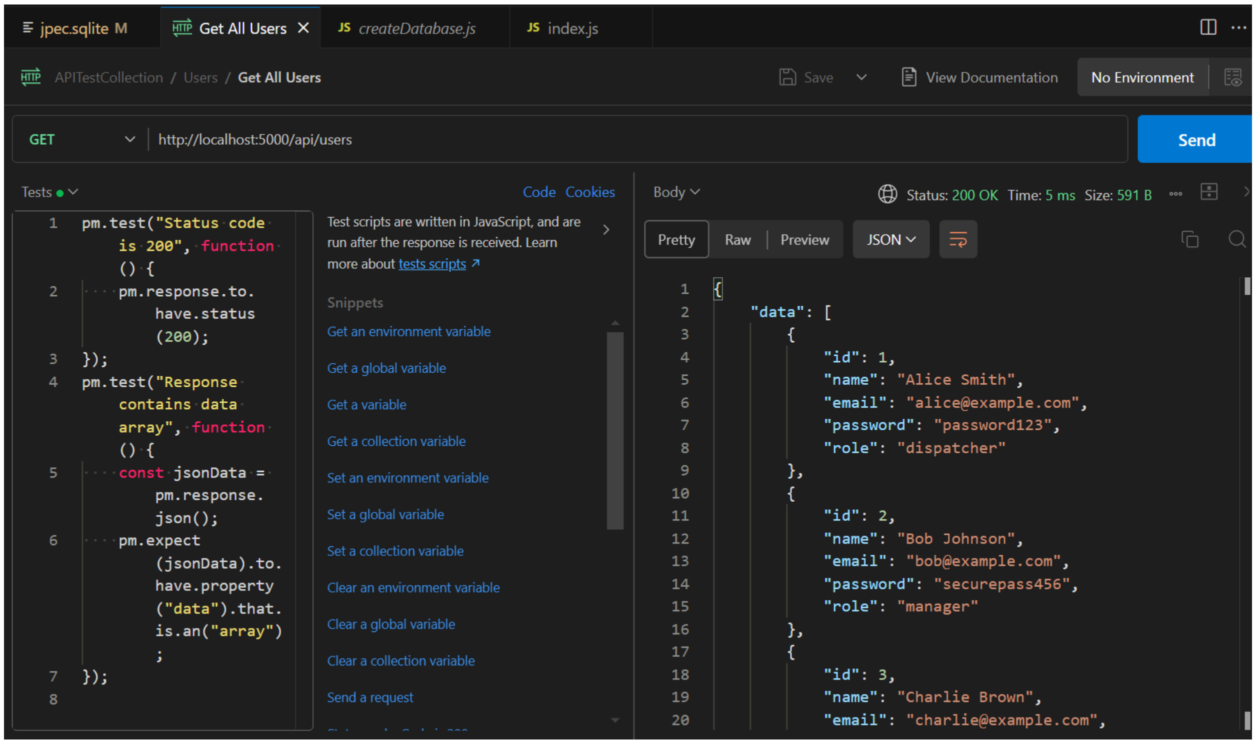
Users

User Table Starting Out:



	id INTEGER PRIMARY KEY AUTOINCREMENT	name TEXT NOT NULL	email TEXT NOT NULL UNIQUE	password TEXT NOT NULL	role TEXT NOT NULL
1	1	Alice Smith	alice@example.com	password123	dispatcher
2	2	Bob Johnson	bob@example.com	securepass456	manager
3	3	Charlie Brown	charlie@example.com	mypassword789	dispatcher

Getting All Users:



jpec.sqlite M | Get All Users | createDatabase.js | index.js

APITestCollection / Users / Get All Users

GET http://localhost:5000/api/users

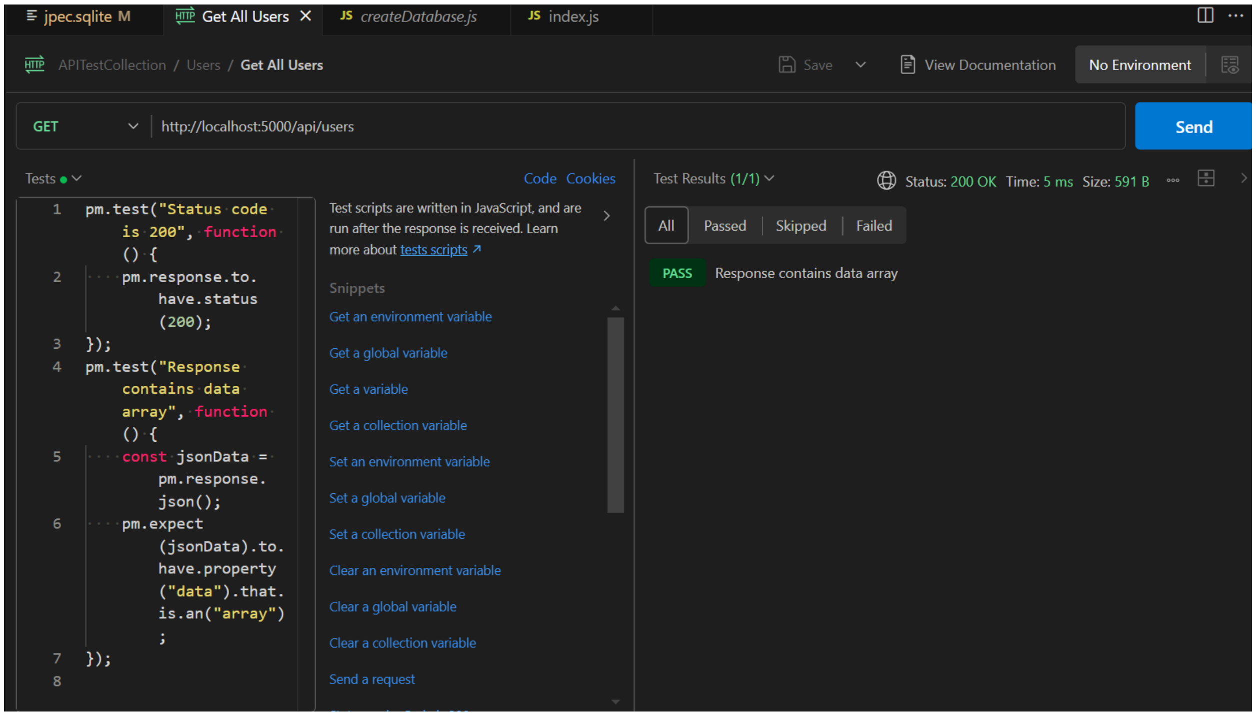
Tests

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains data array", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("array");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Status: 200 OK Time: 5 ms Size: 591 B

```
{
    "data": [
        {
            "id": 1,
            "name": "Alice Smith",
            "email": "alice@example.com",
            "password": "password123",
            "role": "dispatcher"
        },
        {
            "id": 2,
            "name": "Bob Johnson",
            "email": "bob@example.com",
            "password": "securepass456",
            "role": "manager"
        },
        {
            "id": 3,
            "name": "Charlie Brown",
            "email": "charlie@example.com",
```



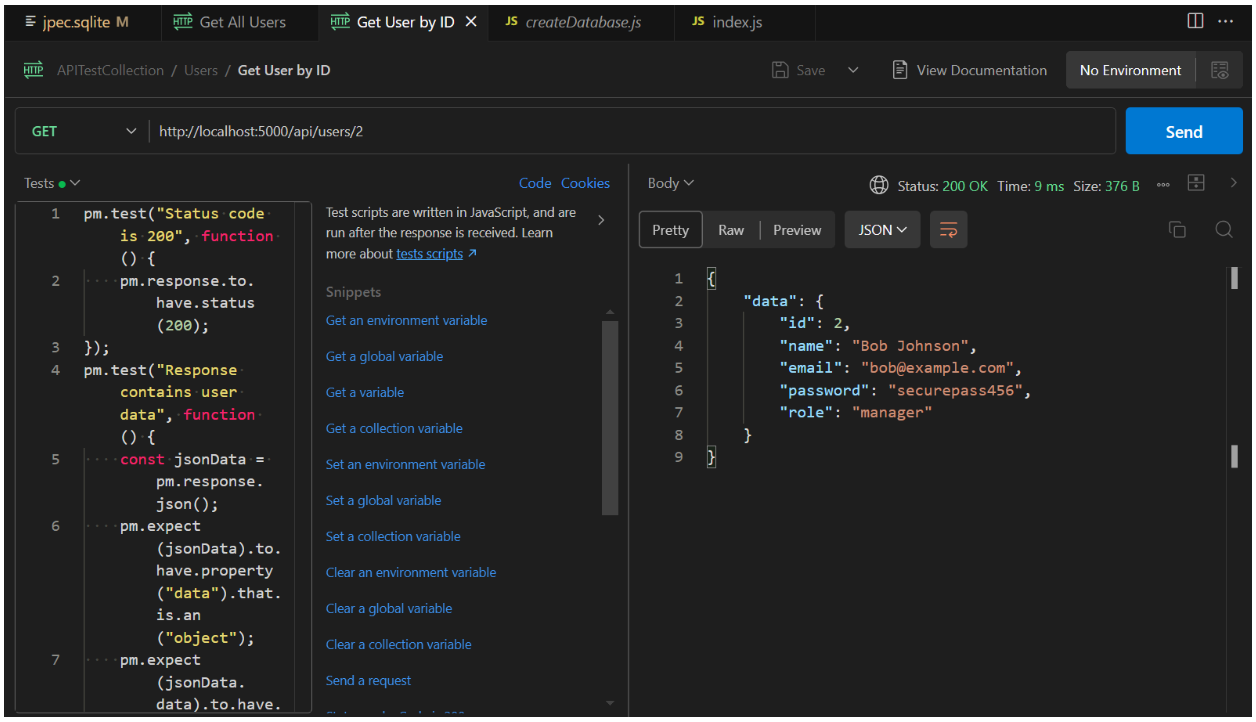
GET http://localhost:5000/api/users

Test Results (1/1)

Status: 200 OK Time: 5 ms Size: 591 B

PASS Response contains data array

Getting User by ID:



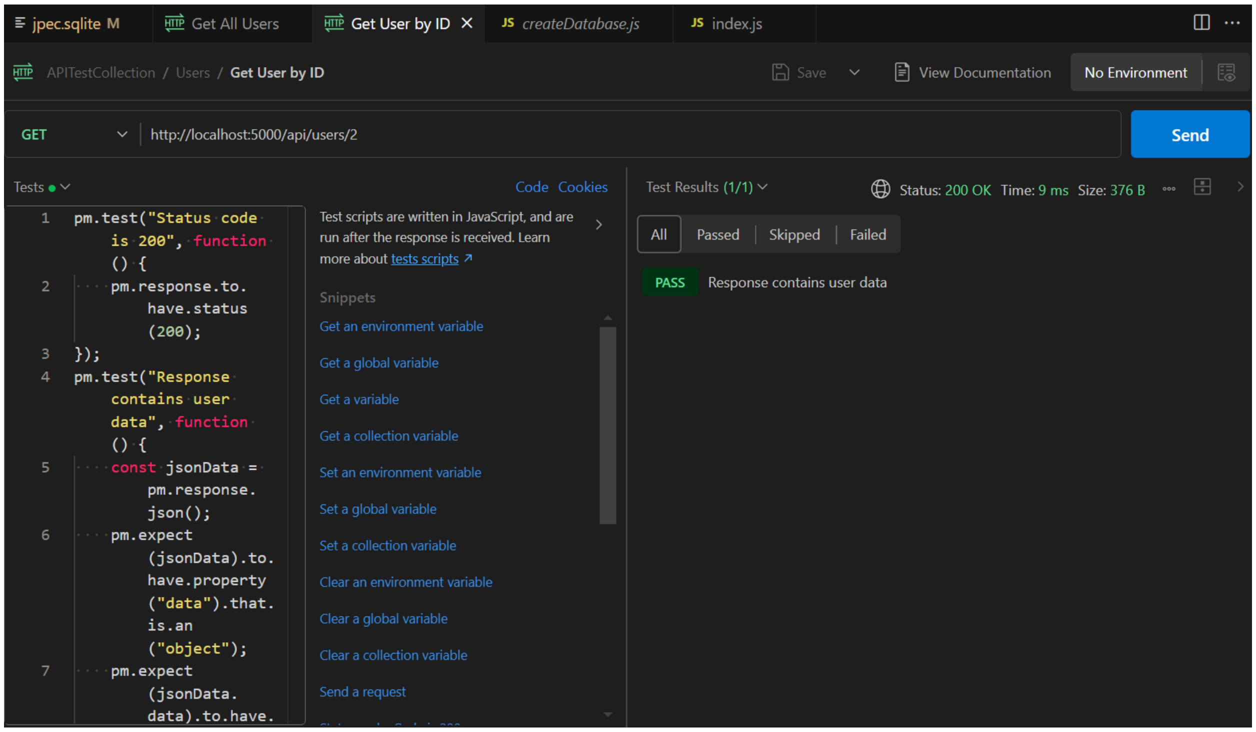
GET http://localhost:5000/api/users/2

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains user data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("object");
    pm.expect(jsonData.data).to.have.
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Status: 200 OK Time: 9 ms Size: 376 B

```
{
    "data": {
        "id": 2,
        "name": "Bob Johnson",
        "email": "bob@example.com",
        "password": "securepass456",
        "role": "manager"
    }
}
```



GET http://localhost:5000/api/users/2

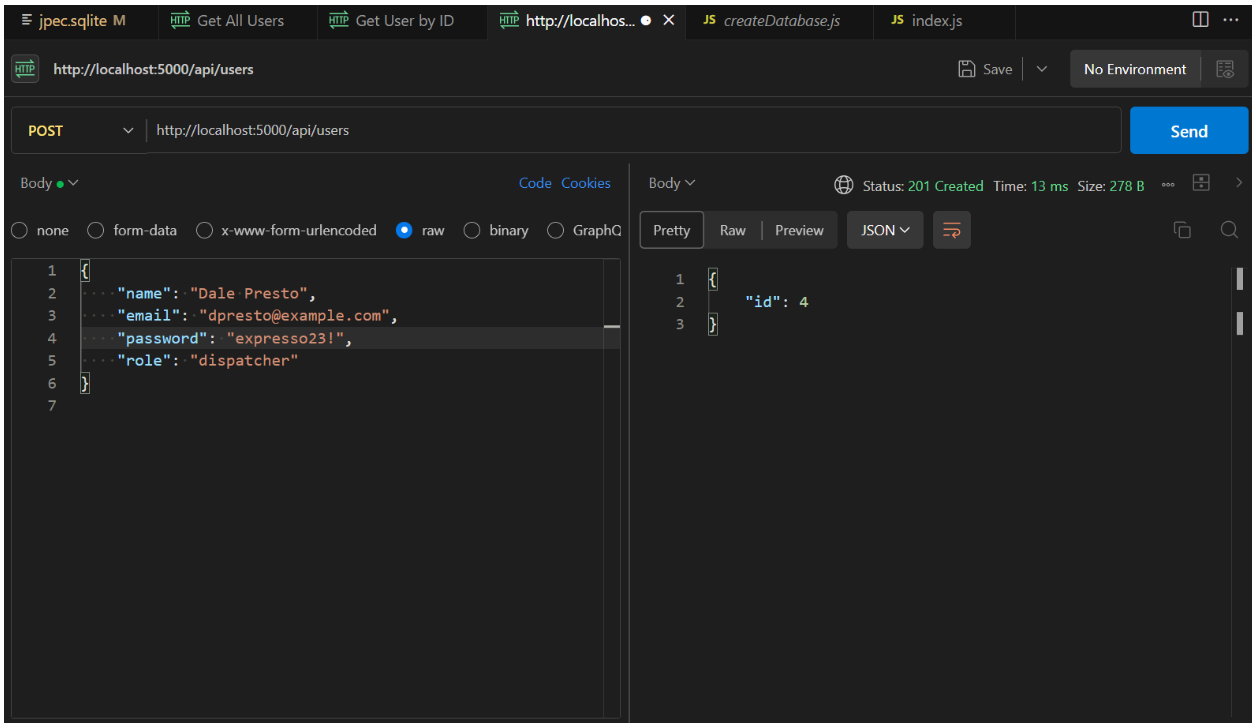
```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response contains user data", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("data").that.is.an("object");
    pm.expect(jsonData.data).to.have.
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Status: 200 OK Time: 9 ms Size: 376 B

PASS Response contains user data

Add User:



http://localhost:5000/api/users

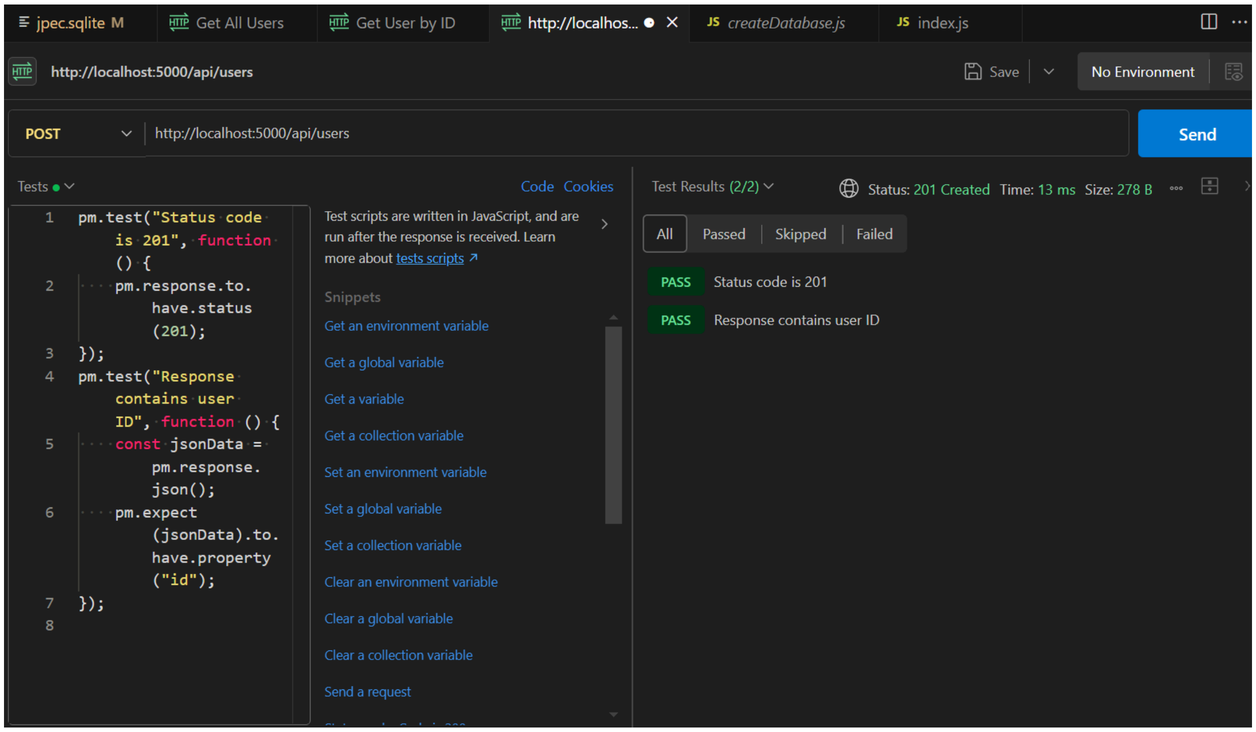
POST http://localhost:5000/api/users

Body [✓ raw]

```
{
    "name": "Dale Presto",
    "email": "dpresto@example.com",
    "password": "expresso23!",
    "role": "dispatcher"
}
```

Status: 201 Created Time: 13 ms Size: 278 B

```
{
    "id": 4
}
```



http://localhost:5000/api/users

POST http://localhost:5000/api/users

Tests

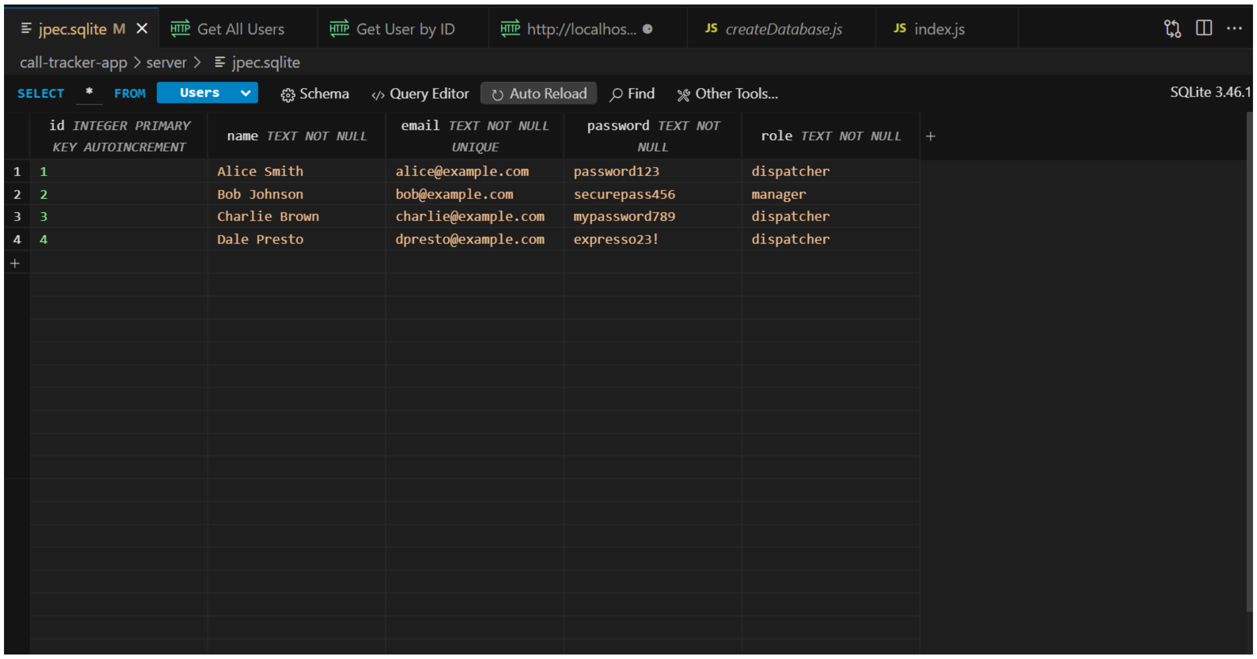
```
pm.test("Status code is 201", function () {
    pm.response.to.have.status(201);
});
pm.test("Response contains user ID", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2) Status: 201 Created Time: 13 ms Size: 278 B

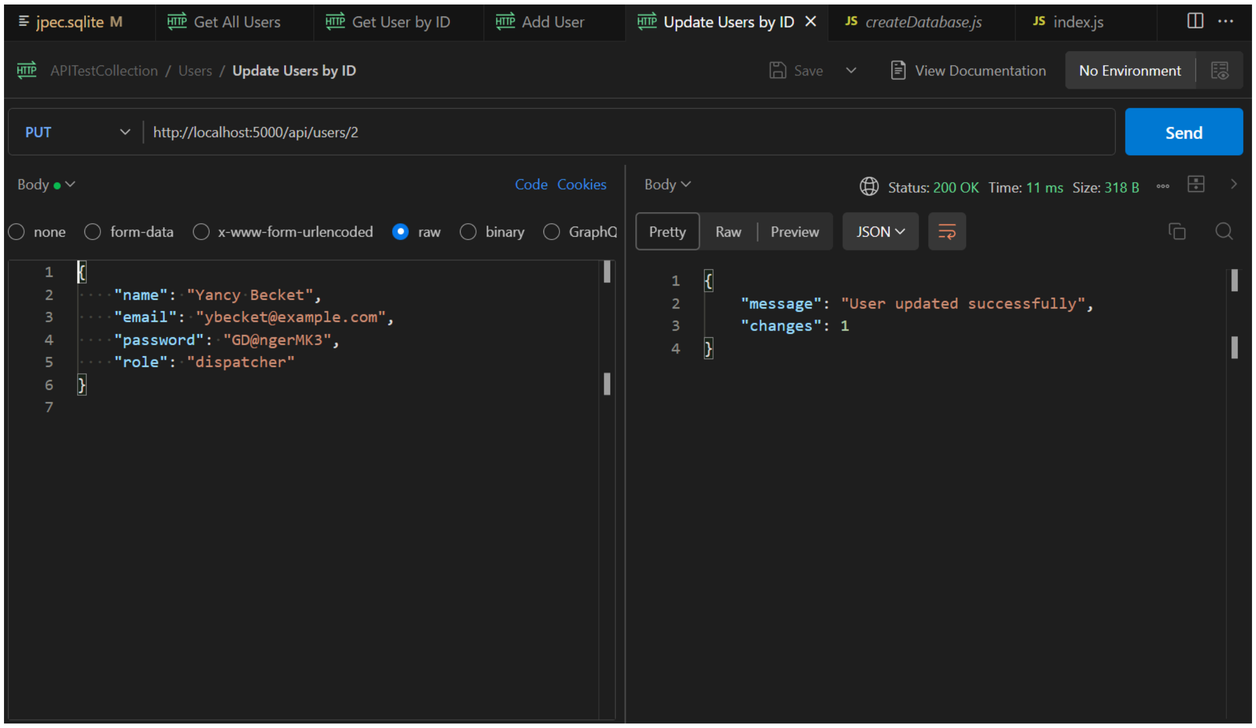
PASS Status code is 201

PASS Response contains user ID



	id INTEGER PRIMARY KEY AUTOINCREMENT	name TEXT NOT NULL	email TEXT NOT NULL UNIQUE	password TEXT NOT NULL	role TEXT NOT NULL
1	1	Alice Smith	alice@example.com	password123	dispatcher
2	2	Bob Johnson	bob@example.com	securepass456	manager
3	3	Charlie Brown	charlie@example.com	mypassword789	dispatcher
4	4	Dale Presto	dpresto@example.com	expresso23!	dispatcher

Updating User by ID:



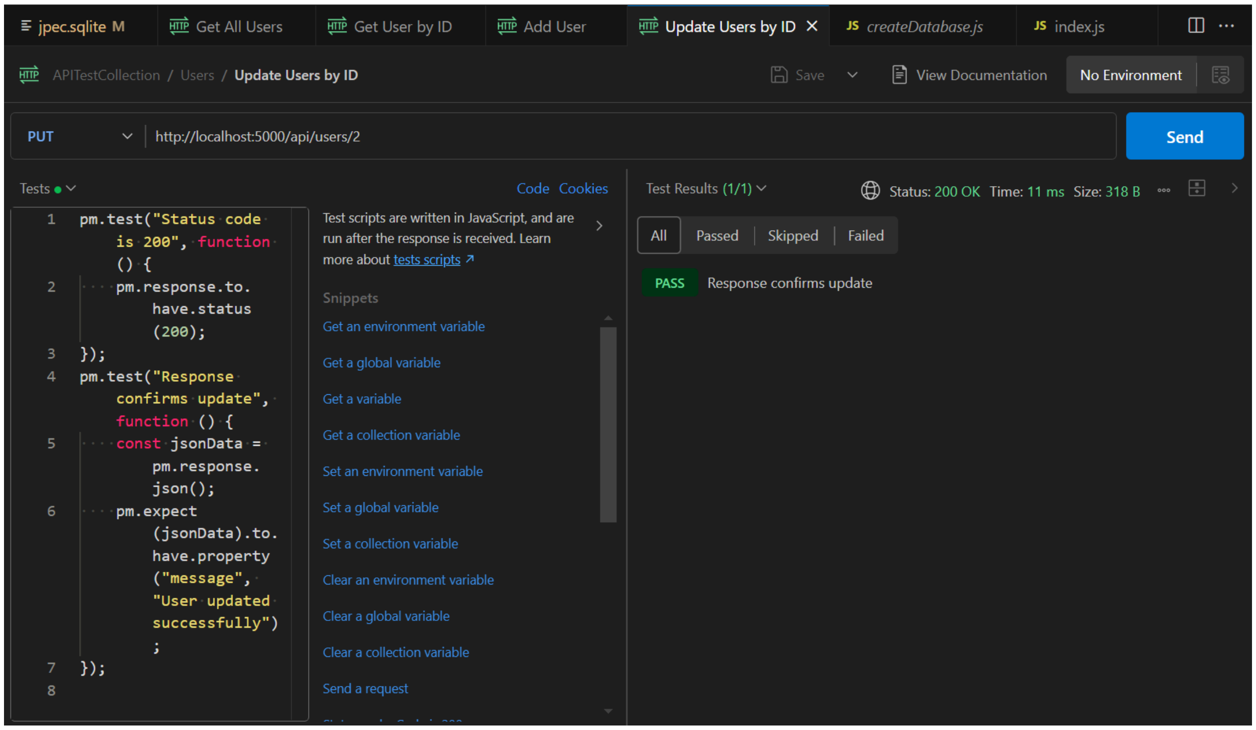
PUT http://localhost:5000/api/users/2

Body: raw

```
{
    "name": "Yancy Becket",
    "email": "ybecket@example.com",
    "password": "GD@ngerMK3",
    "role": "dispatcher"
}
```

Status: 200 OK Time: 11 ms Size: 318 B

```
{
    "message": "User updated successfully",
    "changes": 1
}
```



PUT http://localhost:5000/api/users/2

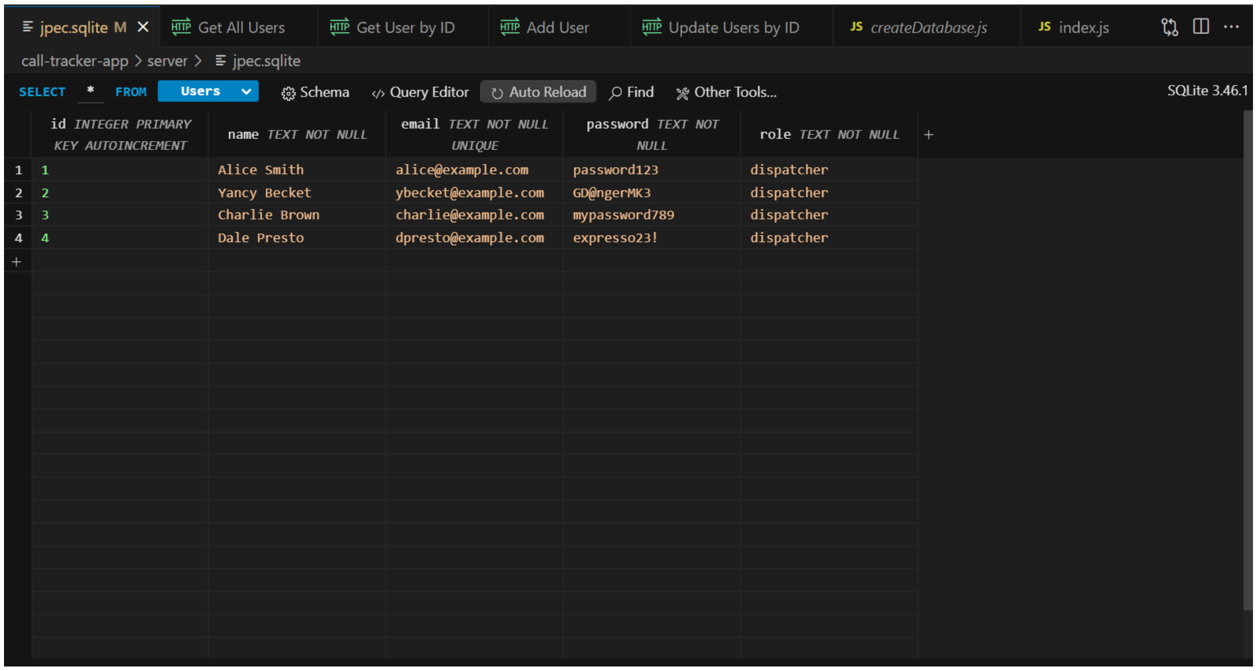
Tests

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response confirms update", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "User updated successfully");
});
```

Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (1/1) Status: 200 OK Time: 11 ms Size: 318 B

PASS Response confirms update

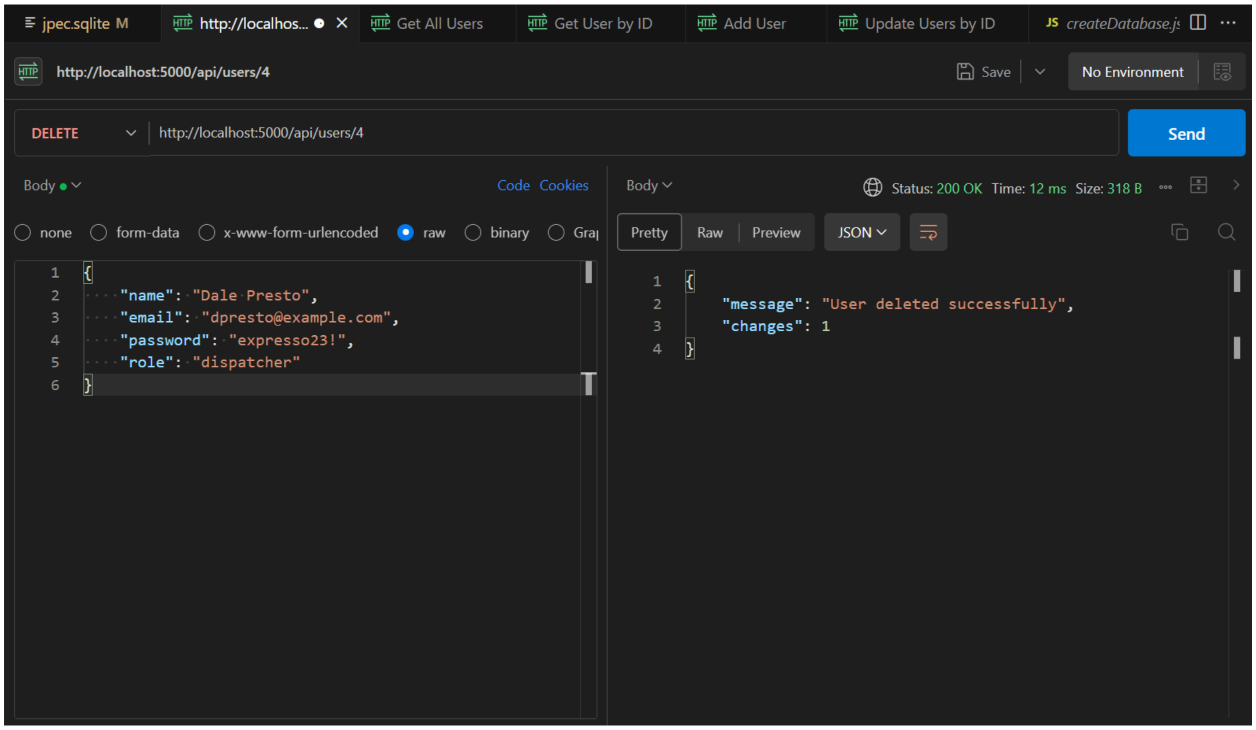

jpec.sqlite M | Get All Users | Get User by ID | Add User | Update Users by ID | createDatabase.js | index.js

call-tracker-app > server > jpec.sqlite

SELECT * FROM Users | Schema | Query Editor | Auto Reload | Find | Other Tools... | SQLite 3.46.1

	id *INTEGER PRIMARY KEY AUTOINCREMENT*	name *TEXT NOT NULL*	email *TEXT NOT NULL UNIQUE*	password *TEXT NOT NULL*	role *TEXT NOT NULL*
1	1	Alice Smith	alice@example.com	password123	dispatcher
2	2	Yancy Becket	ybecket@example.com	GD@ngerMK3	dispatcher
3	3	Charlie Brown	charlie@example.com	mypassword789	dispatcher
4	4	Dale Presto	dpresto@example.com	expresso23!	dispatcher

Deleting User by ID:



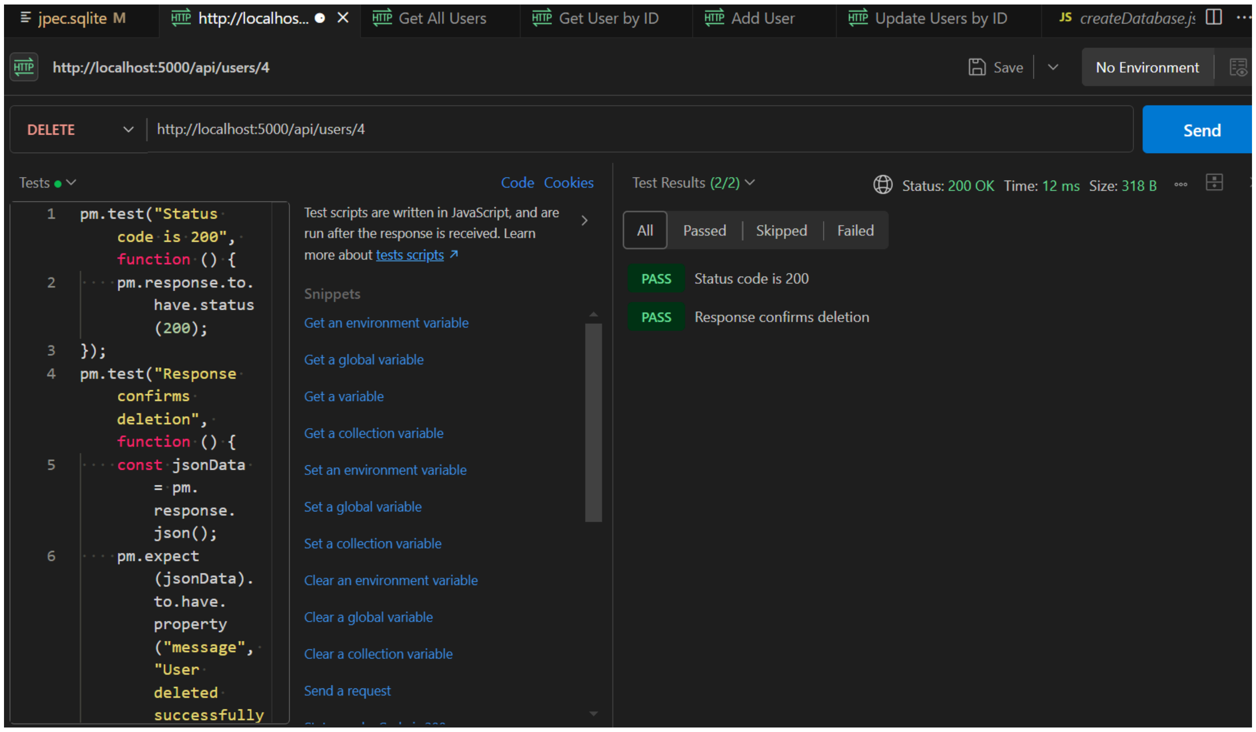
http://localhost:5000/api/users/4

DELETE http://localhost:5000/api/users/4

```
{
    "name": "Dale Presto",
    "email": "dpresto@example.com",
    "password": "expresso23!",
    "role": "dispatcher"
}
```

Status: 200 OK Time: 12 ms Size: 318 B

```
{
    "message": "User deleted successfully",
    "changes": 1
}
```



http://localhost:5000/api/users/4

DELETE http://localhost:5000/api/users/4

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});
pm.test("Response confirms deletion", function () {
    const jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("message", "User deleted successfully
```

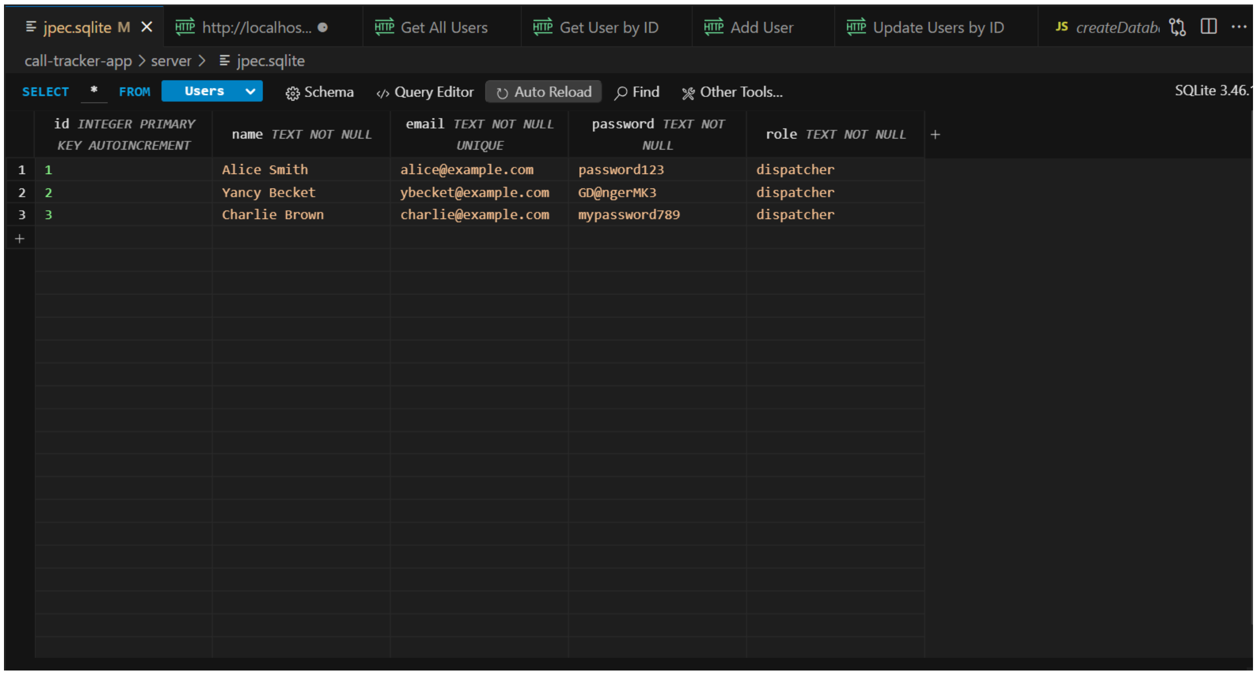
Test scripts are written in JavaScript, and are run after the response is received. Learn more about tests scripts ↗

Test Results (2/2)

Status: 200 OK Time: 12 ms Size: 318 B

PASS Status code is 200

PASS Response confirms deletion



```
SELECT * FROM Users
```

	id INTEGER PRIMARY KEY AUTOINCREMENT	name TEXT NOT NULL	email TEXT NOT NULL UNIQUE	password TEXT NOT NULL	role TEXT NOT NULL
1	1	Alice Smith	alice@example.com	password123	dispatcher
2	2	Yancy Becket	ybecket@example.com	GD@ngerMK3	dispatcher
3	3	Charlie Brown	charlie@example.com	mypassword789	dispatcher