

TDD with NodeJS





Somkiat Puisungnoen

Somkiat Puisungnoen

Update Info 1 View Activity Log 10+ ...

Timeline About Friends 3,138 Photos More

When did you work at Opendream? X

... 22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชัมนาภิกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post Photo/Video Live Video Life Event

What's on your mind?

Public Post

Somkiat Puisungnoen 15 mins · Bangkok · ⚙️

Java and Bigdata





Page

Messages

Notifications 3

Insights

Publishing Tools

Settings

Help ▾



somkiat.cc

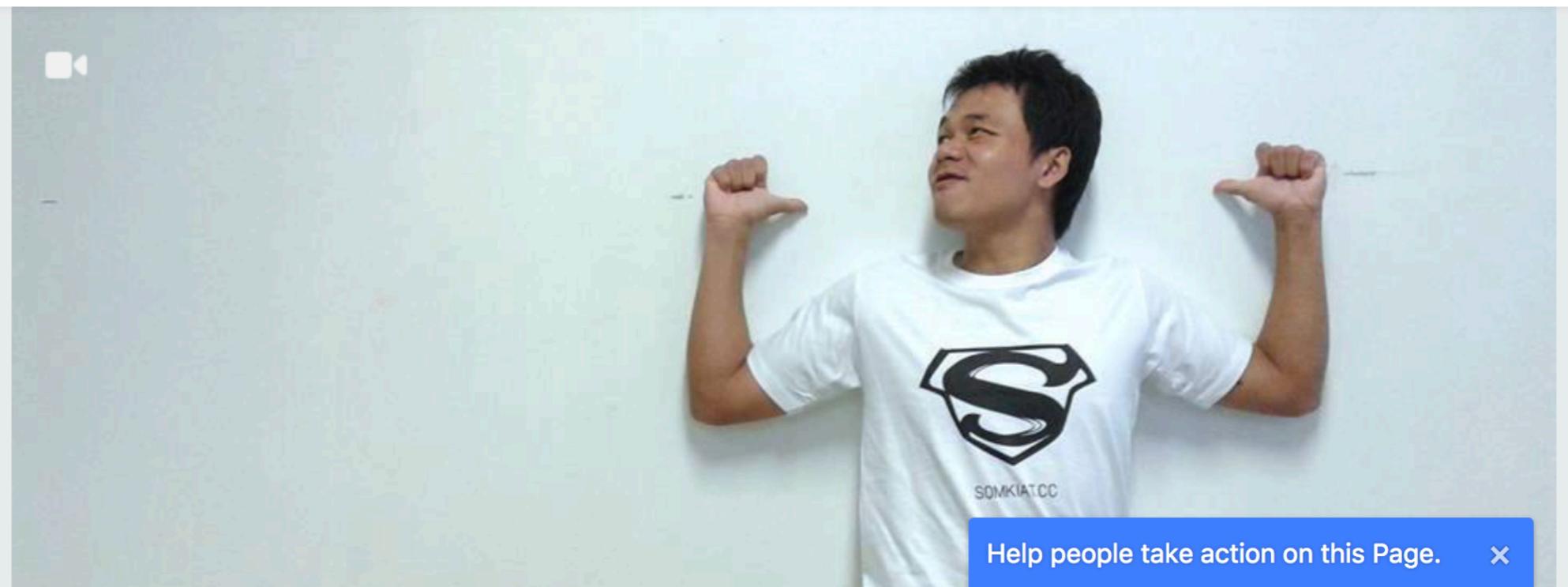
@somkiat.cc

Home

Posts

Videos

Photos



Test-Driven Development



ทำดีดี



ถ้าเข้าจะ test อญ্যោតី ឬ ខ្សោយ test



Agenda

Test-Driven Development

Types of Testing

Good Unit Test (GUT)

Structure of Good Unit Test

Testing with Node.JS

Test/Code coverage

Test-Double

Testable application



<https://github.com/up1/course-nodejs-2020>



Learn by **DOING.**



Test-Drive Development



Types of Testing



Good Unit Tests



Structure of Test



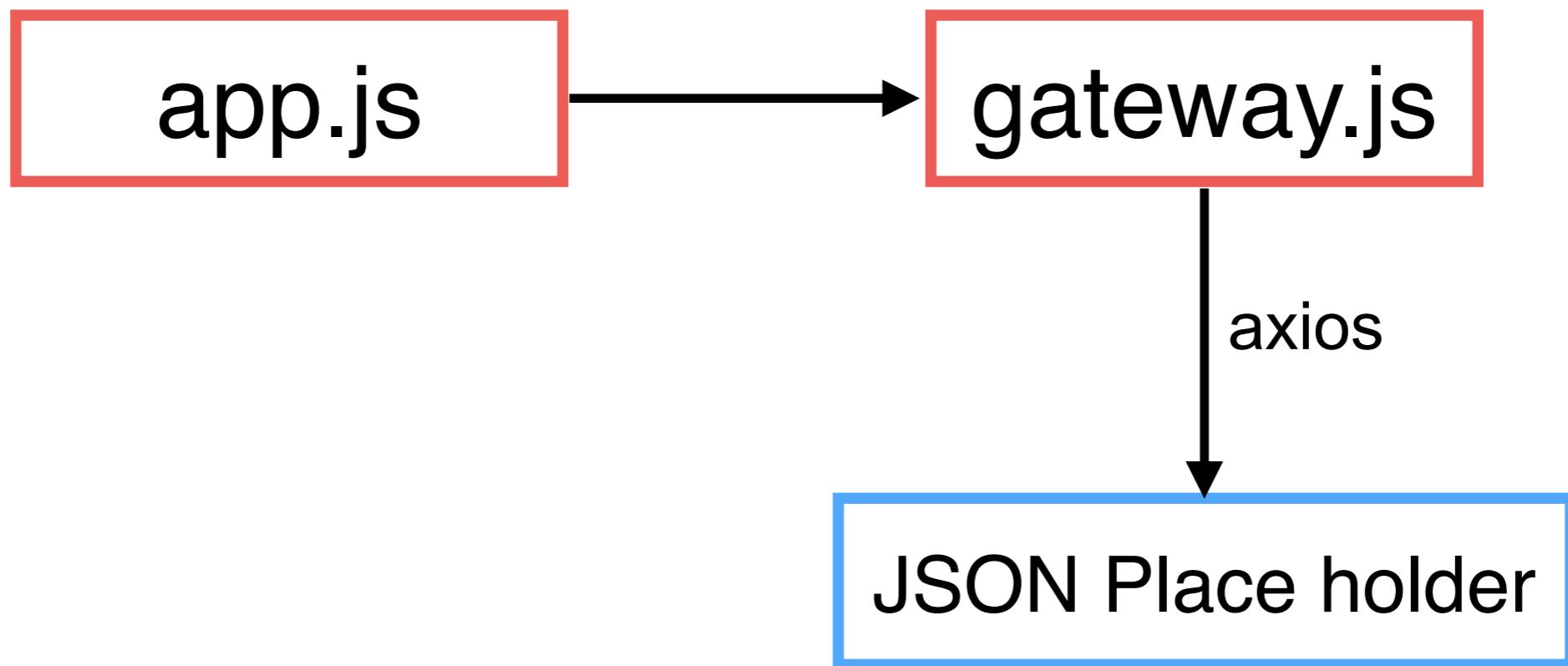
Test Double



NodeJS



Workshop



<https://github.com/axios/axios>



JSON Place holder

JSONPlaceholder

Fake Online REST API for Testing and Prototyping

Powered by [JSON Server](#) + [LowDB](#)

```
fetch('https://jsonplaceholder.cypress.io/todos/1')
  .then(response => response.json())
  .then(json => console.log(json))
```

Try it

<https://jsonplaceholder.cypress.io/users>

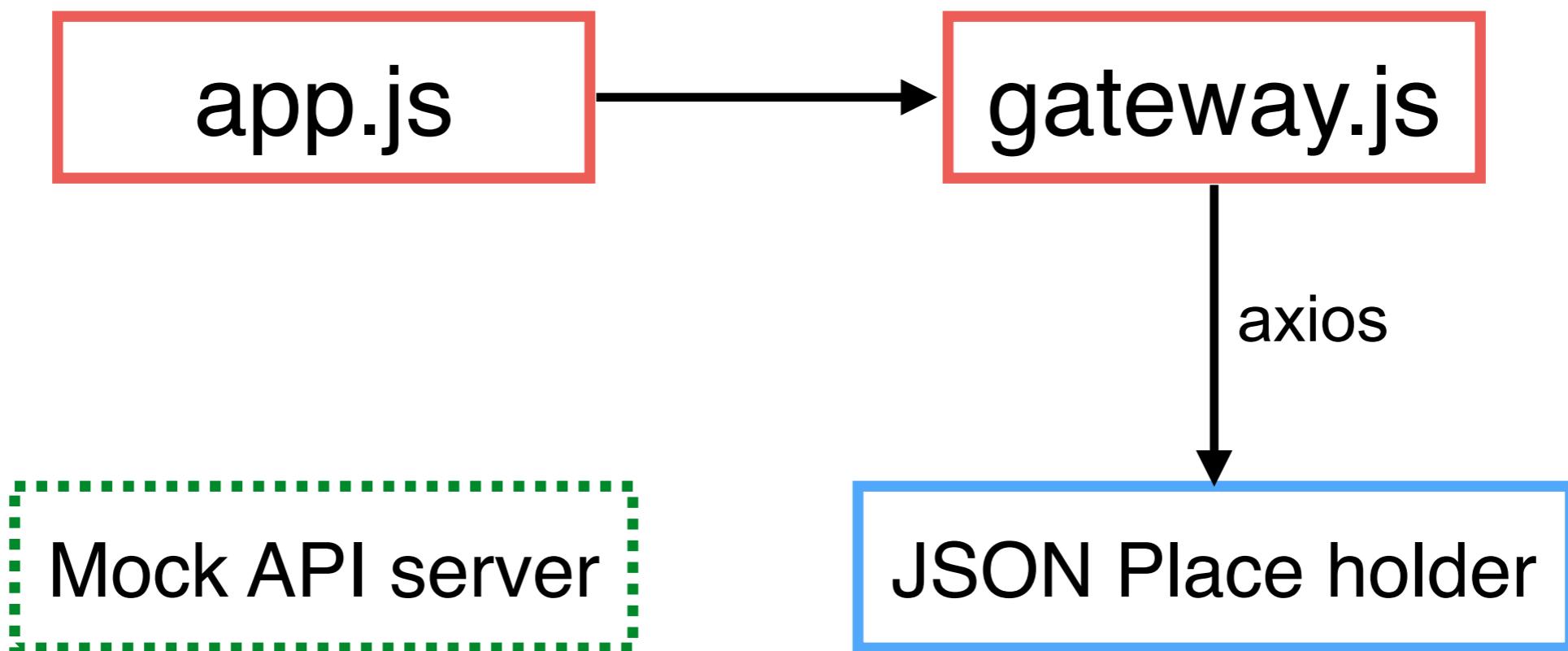


Testing API with Nock

<https://github.com/nock/nock>



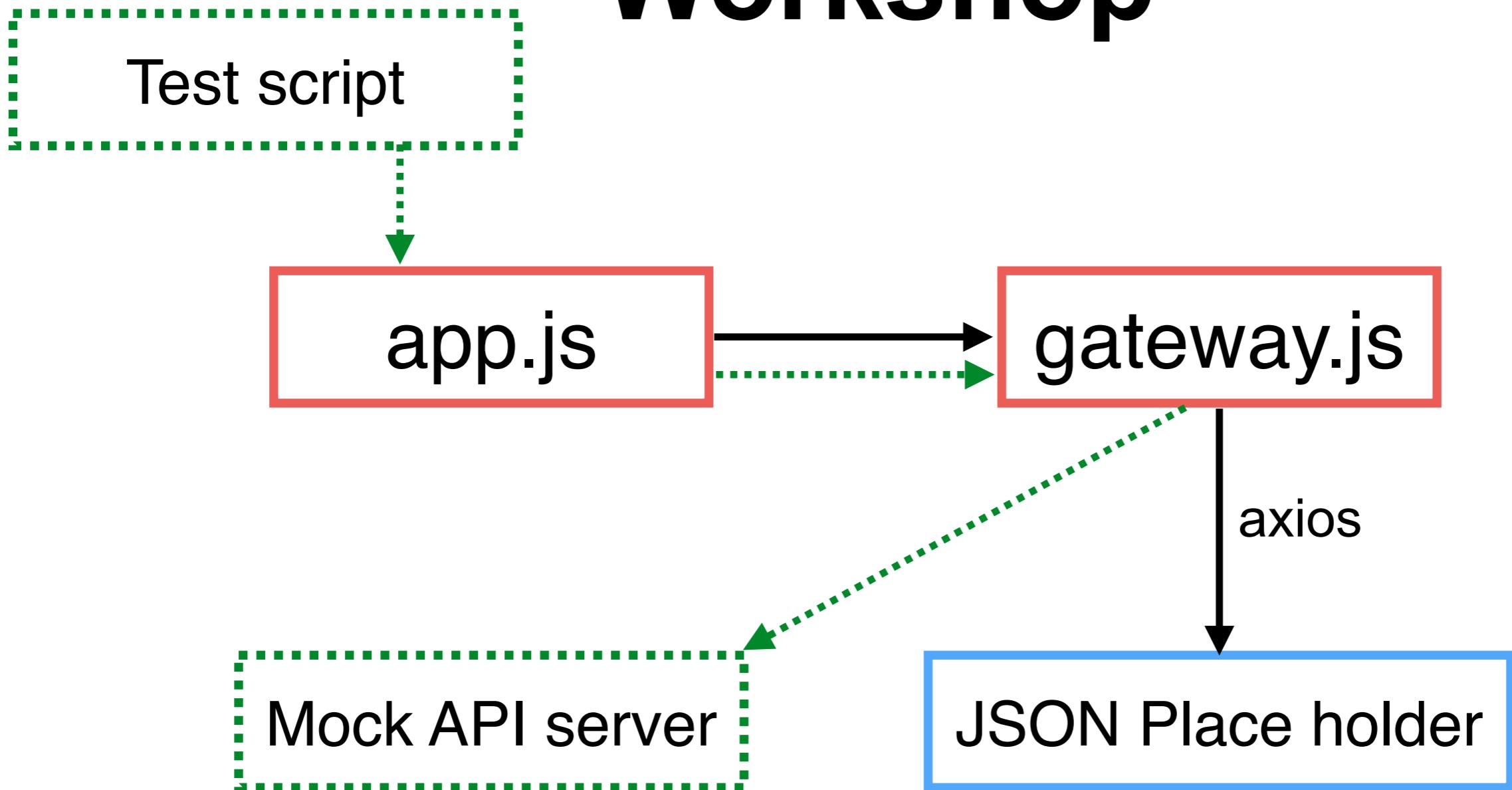
Workshop



<https://github.com/axios/axios>



Workshop

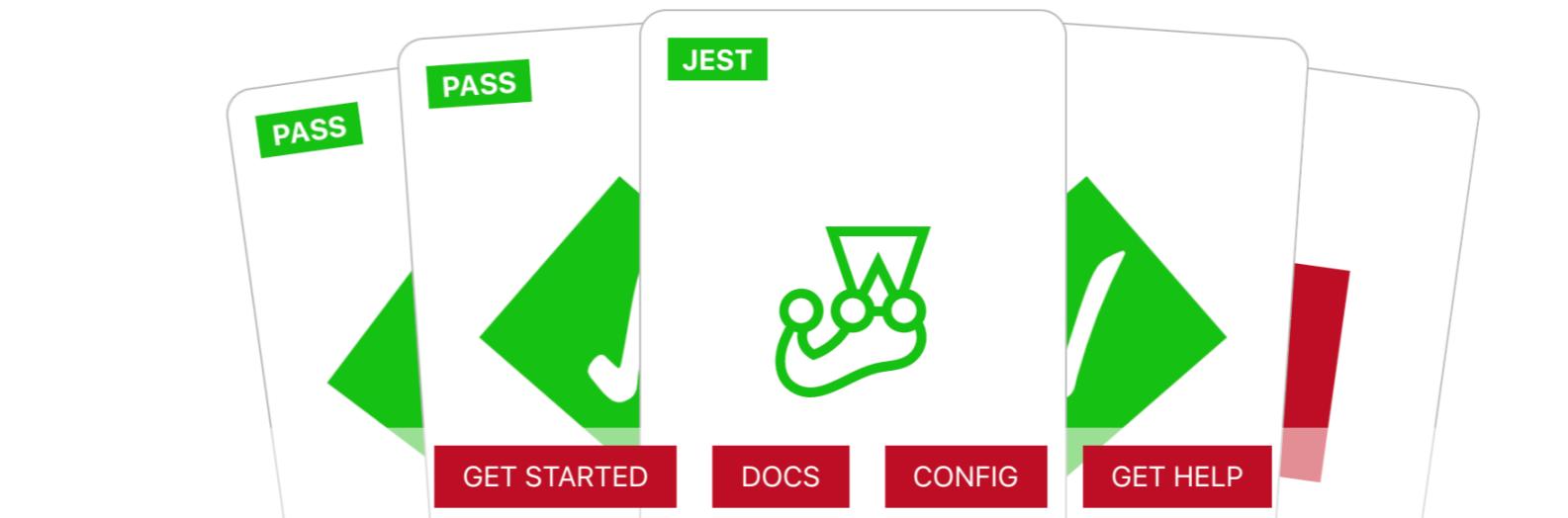


<https://github.com/axios/axios>



Jest

JavaScript Test Framework



The image shows a mockup of the Jest website landing page. At the top, there are three cards, each with a green 'PASS' button in the top right corner and a large green checkmark icon. The middle card is labeled 'JEST' at the top. Below the cards are four red buttons: 'GET STARTED', 'DOCS', 'CONFIG', and 'GET HELP'. In the center of the page is a green icon of a person jumping over a finish line. The main text area contains the following text:

Jest is a delightful JavaScript Testing Framework with a focus on simplicity.

It works with projects using: Babel, TypeScript, Node, React, Angular, Vue and more!

<http://jestjs.io/>



Jest

JavaScript Test Framework

```
$npm install jest --save-dev
```



Create first test case

```
test("Hello world!", () => {  
});
```



Assertion

```
const { calculateTip } = require("../src/math");

test("Should calculate total with tip", () => {
  const total = calculateTip(10, 0.3);

  expect(total).toBe(13);
});

});
```

<https://jestjs.io/docs/en/expect>



Testing asynchronous code

With callback function

```
test("Should add two numbers", (done) => {
  add(2, 3).then((sum) => {
    expect(sum).toBe(5);
    done();
  });
});
```



Testing asynchronous code

With async/await

```
test("Should add two numbers async/await", async () => {
  const sum = await add(2, 3);
  expect(sum).toBe(3);
});
```



Jest setup and teardown

beforeEach

afterEach

before

after



Mocking with Jest

Create directory __mocks__ in tests

Working with `jest.mock()`

`Jest.spyOn`



Jasmine

BDD for JavaScript



<https://jasmine.github.io/>



Nock

HTTP server mocking and expectations library for Node.js

```
$npm install nock --save-dev
```



Nock

HTTP server mocking and expectations library for Node.js

```
$npm install nock --save-dev
```



Write a first test

Create folder __tests__

Create files with *_spec|test.js



Gateway-spec.js

```
const nock = require("nock");
const API_PORT = 9999;
const getAllUser = require("../gateway-testable.js");
const API_HOST = `http://localhost:${API_PORT}`;

describe("Call service", () => {

  it("Check response from /users", async () => {
    // Mock server
    nock(API_HOST)
      .defaultReplyHeaders({ 'access-control-allow-origin': '*' })
      .get("/users").reply(200, [], {});

    // Verify
    const response = await getAllUser();
    expect(response.data.length).toEqual(2);
  });

});
```



Config Jest in package.json

```
"jest": {  
  "collectCoverage": true,  
  "coverageReporters": ["json", "html"]  
}
```



Run test

\$npm test

```
FAIL __tests__/gateway-spec.js
Call service
  ✘ Check response from /users (886 ms)

● Call service > Check response from /users

expect(received).toEqual(expected) // deep equality

Expected: 2
Received: 10

  9 |
 10 |   const response = await getAllUser();
> 11 |   expect(response.data.length).toEqual(2);
    |   ^
 12 | });
 13 | });
 14 |

at Object.<anonymous> (__tests__/gateway-spec.js:11:34)
```



Testable code ?

Using environment variable

Environment Variable

`API_URL=<your api server>`

Node.JS

Using `process` module

`process.env('API_URL')`

https://nodejs.org/api/process.html#process_process_env



Gateway.js

```
const axios = require("axios").default;  
const process = require("process");  
  
function getAllUser() {  
    return axios({  
        method: "get",  
        url: `${process.env.API_URL}/users`,  
        responseType: "json",  
    })  
        .then((response) => {  
            return {  
                code: 200,  
                data: response.data,  
            };  
        })  
}
```



Fail case

```
it("Fail 404 /users", async () => {
  // Mock server
  nock(API_HOST)
    .defaultReplyHeaders({ "access-control-allow-origin": "*" })
    .get("/users")
    .reply(404);

  // Verify
  const response = await getAllUser();
  expect(response.code).toEqual(500);
});
```



Config Jest in package.json

```
"scripts": {  
  "test": "API_URL=http://localhost:9999 jest --coverage"  
}  
  
...  
  
"jest": {  
  "collectCoverage": true,  
  "coverageReporters": ["json", "html"]  
}
```



Run test

\$npm test

```
PASS  __tests__/_gateway-spec.js
Call service
  ✓ Check response from /users (24 ms)
  ✓ Fail 404 /users (4 ms)
```

```
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        0.932 s, estimated 1 s
Ran all test suites.
```



Coverage report

\$npm test

All files gateway-testable.js

100% Statements 6/6 100% Branches 0/0 100% Functions 3/3 100% Lines 6/6

Press *n* or *j* to go to the next uncovered block, *b*, *p* or *k* for the previous block.

```
1 1x const axios = require("axios").default;
2 1x const process = require("process");
3
4 function getAllUser() {
5 2x   return axios({
6     method: "get",
7     url: `${process.env.API_URL}/users`,
8     responseType: "json",
9   })
10    .then((response) => {
11      1x       return {
12        code: 200,
13        data: response.data,
14      };
15    });
16 1x 1x 1x
17
18 1x
19 1x
20 1x
21 1x
22 1x
23 1x
24 1x
25 1x
26 1x
27 1x
28 1x
29 1x
30 1x
31 1x
32 1x
33 1x
34 1x
35 1x
36 1x
37 1x
38 1x
39 1x
40 1x
41 1x
42 1x
43 1x
44 1x
45 1x
46 1x
47 1x
48 1x
49 1x
50 1x
51 1x
52 1x
53 1x
54 1x
55 1x
56 1x
57 1x
58 1x
59 1x
60 1x
61 1x
62 1x
63 1x
64 1x
65 1x
66 1x
67 1x
68 1x
69 1x
70 1x
71 1x
72 1x
73 1x
74 1x
75 1x
76 1x
77 1x
78 1x
79 1x
80 1x
81 1x
82 1x
83 1x
84 1x
85 1x
86 1x
87 1x
88 1x
89 1x
90 1x
91 1x
92 1x
93 1x
94 1x
95 1x
96 1x
97 1x
98 1x
99 1x
100 1x
101 1x
102 1x
103 1x
104 1x
105 1x
106 1x
107 1x
108 1x
109 1x
110 1x
111 1x
112 1x
113 1x
114 1x
115 1x
116 1x
117 1x
118 1x
119 1x
120 1x
121 1x
122 1x
123 1x
124 1x
125 1x
126 1x
127 1x
128 1x
129 1x
130 1x
131 1x
132 1x
133 1x
134 1x
135 1x
136 1x
137 1x
138 1x
139 1x
140 1x
141 1x
142 1x
143 1x
144 1x
145 1x
146 1x
147 1x
148 1x
149 1x
150 1x
151 1x
152 1x
153 1x
154 1x
155 1x
156 1x
157 1x
158 1x
159 1x
160 1x
161 1x
162 1x
163 1x
164 1x
165 1x
166 1x
167 1x
168 1x
169 1x
170 1x
171 1x
172 1x
173 1x
174 1x
175 1x
176 1x
177 1x
178 1x
179 1x
180 1x
181 1x
182 1x
183 1x
184 1x
185 1x
186 1x
187 1x
188 1x
189 1x
190 1x
191 1x
192 1x
193 1x
194 1x
195 1x
196 1x
197 1x
198 1x
199 1x
200 1x
201 1x
202 1x
203 1x
204 1x
205 1x
206 1x
207 1x
208 1x
209 1x
210 1x
211 1x
212 1x
213 1x
214 1x
215 1x
216 1x
217 1x
218 1x
219 1x
220 1x
221 1x
222 1x
223 1x
224 1x
225 1x
226 1x
227 1x
228 1x
229 1x
230 1x
231 1x
232 1x
233 1x
234 1x
235 1x
236 1x
237 1x
238 1x
239 1x
240 1x
241 1x
242 1x
243 1x
244 1x
245 1x
246 1x
247 1x
248 1x
249 1x
250 1x
251 1x
252 1x
253 1x
254 1x
255 1x
256 1x
257 1x
258 1x
259 1x
260 1x
261 1x
262 1x
263 1x
264 1x
265 1x
266 1x
267 1x
268 1x
269 1x
270 1x
271 1x
272 1x
273 1x
274 1x
275 1x
276 1x
277 1x
278 1x
279 1x
280 1x
281 1x
282 1x
283 1x
284 1x
285 1x
286 1x
287 1x
288 1x
289 1x
290 1x
291 1x
292 1x
293 1x
294 1x
295 1x
296 1x
297 1x
298 1x
299 1x
300 1x
301 1x
302 1x
303 1x
304 1x
305 1x
306 1x
307 1x
308 1x
309 1x
310 1x
311 1x
312 1x
313 1x
314 1x
315 1x
316 1x
317 1x
318 1x
319 1x
320 1x
321 1x
322 1x
323 1x
324 1x
325 1x
326 1x
327 1x
328 1x
329 1x
330 1x
331 1x
332 1x
333 1x
334 1x
335 1x
336 1x
337 1x
338 1x
339 1x
340 1x
341 1x
342 1x
343 1x
344 1x
345 1x
346 1x
347 1x
348 1x
349 1x
350 1x
351 1x
352 1x
353 1x
354 1x
355 1x
356 1x
357 1x
358 1x
359 1x
360 1x
361 1x
362 1x
363 1x
364 1x
365 1x
366 1x
367 1x
368 1x
369 1x
370 1x
371 1x
372 1x
373 1x
374 1x
375 1x
376 1x
377 1x
378 1x
379 1x
380 1x
381 1x
382 1x
383 1x
384 1x
385 1x
386 1x
387 1x
388 1x
389 1x
390 1x
391 1x
392 1x
393 1x
394 1x
395 1x
396 1x
397 1x
398 1x
399 1x
400 1x
401 1x
402 1x
403 1x
404 1x
405 1x
406 1x
407 1x
408 1x
409 1x
410 1x
411 1x
412 1x
413 1x
414 1x
415 1x
416 1x
417 1x
418 1x
419 1x
420 1x
421 1x
422 1x
423 1x
424 1x
425 1x
426 1x
427 1x
428 1x
429 1x
430 1x
431 1x
432 1x
433 1x
434 1x
435 1x
436 1x
437 1x
438 1x
439 1x
440 1x
441 1x
442 1x
443 1x
444 1x
445 1x
446 1x
447 1x
448 1x
449 1x
450 1x
451 1x
452 1x
453 1x
454 1x
455 1x
456 1x
457 1x
458 1x
459 1x
460 1x
461 1x
462 1x
463 1x
464 1x
465 1x
466 1x
467 1x
468 1x
469 1x
470 1x
471 1x
472 1x
473 1x
474 1x
475 1x
476 1x
477 1x
478 1x
479 1x
480 1x
481 1x
482 1x
483 1x
484 1x
485 1x
486 1x
487 1x
488 1x
489 1x
490 1x
491 1x
492 1x
493 1x
494 1x
495 1x
496 1x
497 1x
498 1x
499 1x
500 1x
501 1x
502 1x
503 1x
504 1x
505 1x
506 1x
507 1x
508 1x
509 1x
510 1x
511 1x
512 1x
513 1x
514 1x
515 1x
516 1x
517 1x
518 1x
519 1x
520 1x
521 1x
522 1x
523 1x
524 1x
525 1x
526 1x
527 1x
528 1x
529 1x
530 1x
531 1x
532 1x
533 1x
534 1x
535 1x
536 1x
537 1x
538 1x
539 1x
540 1x
541 1x
542 1x
543 1x
544 1x
545 1x
546 1x
547 1x
548 1x
549 1x
550 1x
551 1x
552 1x
553 1x
554 1x
555 1x
556 1x
557 1x
558 1x
559 1x
560 1x
561 1x
562 1x
563 1x
564 1x
565 1x
566 1x
567 1x
568 1x
569 1x
570 1x
571 1x
572 1x
573 1x
574 1x
575 1x
576 1x
577 1x
578 1x
579 1x
580 1x
581 1x
582 1x
583 1x
584 1x
585 1x
586 1x
587 1x
588 1x
589 1x
590 1x
591 1x
592 1x
593 1x
594 1x
595 1x
596 1x
597 1x
598 1x
599 1x
600 1x
601 1x
602 1x
603 1x
604 1x
605 1x
606 1x
607 1x
608 1x
609 1x
610 1x
611 1x
612 1x
613 1x
614 1x
615 1x
616 1x
617 1x
618 1x
619 1x
620 1x
621 1x
622 1x
623 1x
624 1x
625 1x
626 1x
627 1x
628 1x
629 1x
630 1x
631 1x
632 1x
633 1x
634 1x
635 1x
636 1x
637 1x
638 1x
639 1x
640 1x
641 1x
642 1x
643 1x
644 1x
645 1x
646 1x
647 1x
648 1x
649 1x
650 1x
651 1x
652 1x
653 1x
654 1x
655 1x
656 1x
657 1x
658 1x
659 1x
660 1x
661 1x
662 1x
663 1x
664 1x
665 1x
666 1x
667 1x
668 1x
669 1x
670 1x
671 1x
672 1x
673 1x
674 1x
675 1x
676 1x
677 1x
678 1x
679 1x
680 1x
681 1x
682 1x
683 1x
684 1x
685 1x
686 1x
687 1x
688 1x
689 1x
690 1x
691 1x
692 1x
693 1x
694 1x
695 1x
696 1x
697 1x
698 1x
699 1x
700 1x
701 1x
702 1x
703 1x
704 1x
705 1x
706 1x
707 1x
708 1x
709 1x
710 1x
711 1x
712 1x
713 1x
714 1x
715 1x
716 1x
717 1x
718 1x
719 1x
720 1x
721 1x
722 1x
723 1x
724 1x
725 1x
726 1x
727 1x
728 1x
729 1x
730 1x
731 1x
732 1x
733 1x
734 1x
735 1x
736 1x
737 1x
738 1x
739 1x
740 1x
741 1x
742 1x
743 1x
744 1x
745 1x
746 1x
747 1x
748 1x
749 1x
750 1x
751 1x
752 1x
753 1x
754 1x
755 1x
756 1x
757 1x
758 1x
759 1x
760 1x
761 1x
762 1x
763 1x
764 1x
765 1x
766 1x
767 1x
768 1x
769 1x
770 1x
771 1x
772 1x
773 1x
774 1x
775 1x
776 1x
777 1x
778 1x
779 1x
780 1x
781 1x
782 1x
783 1x
784 1x
785 1x
786 1x
787 1x
788 1x
789 1x
790 1x
791 1x
792 1x
793 1x
794 1x
795 1x
796 1x
797 1x
798 1x
799 1x
800 1x
801 1x
802 1x
803 1x
804 1x
805 1x
806 1x
807 1x
808 1x
809 1x
810 1x
811 1x
812 1x
813 1x
814 1x
815 1x
816 1x
817 1x
818 1x
819 1x
820 1x
821 1x
822 1x
823 1x
824 1x
825 1x
826 1x
827 1x
828 1x
829 1x
830 1x
831 1x
832 1x
833 1x
834 1x
835 1x
836 1x
837 1x
838 1x
839 1x
840 1x
841 1x
842 1x
843 1x
844 1x
845 1x
846 1x
847 1x
848 1x
849 1x
850 1x
851 1x
852 1x
853 1x
854 1x
855 1x
856 1x
857 1x
858 1x
859 1x
860 1x
861 1x
862 1x
863 1x
864 1x
865 1x
866 1x
867 1x
868 1x
869 1x
870 1x
871 1x
872 1x
873 1x
874 1x
875 1x
876 1x
877 1x
878 1x
879 1x
880 1x
881 1x
882 1x
883 1x
884 1x
885 1x
886 1x
887 1x
888 1x
889 1x
890 1x
891 1x
892 1x
893 1x
894 1x
895 1x
896 1x
897 1x
898 1x
899 1x
900 1x
901 1x
902 1x
903 1x
904 1x
905 1x
906 1x
907 1x
908 1x
909 1x
910 1x
911 1x
912 1x
913 1x
914 1x
915 1x
916 1x
917 1x
918 1x
919 1x
920 1x
921 1x
922 1x
923 1x
924 1x
925 1x
926 1x
927 1x
928 1x
929 1x
930 1x
931 1x
932 1x
933 1x
934 1x
935 1x
936 1x
937 1x
938 1x
939 1x
940 1x
941 1x
942 1x
943 1x
944 1x
945 1x
946 1x
947 1x
948 1x
949 1x
950 1x
951 1x
952 1x
953 1x
954 1x
955 1x
956 1x
957 1x
958 1x
959 1x
960 1x
961 1x
962 1x
963 1x
964 1x
965 1x
966 1x
967 1x
968 1x
969 1x
970 1x
971 1x
972 1x
973 1x
974 1x
975 1x
976 1x
977 1x
978 1x
979 1x
980 1x
981 1x
982 1x
983 1x
984 1x
985 1x
986 1x
987 1x
988 1x
989 1x
990 1x
991 1x
992 1x
993 1x
994 1x
995 1x
996 1x
997 1x
998 1x
999 1x
1000 1x
```



Working with Database



Database

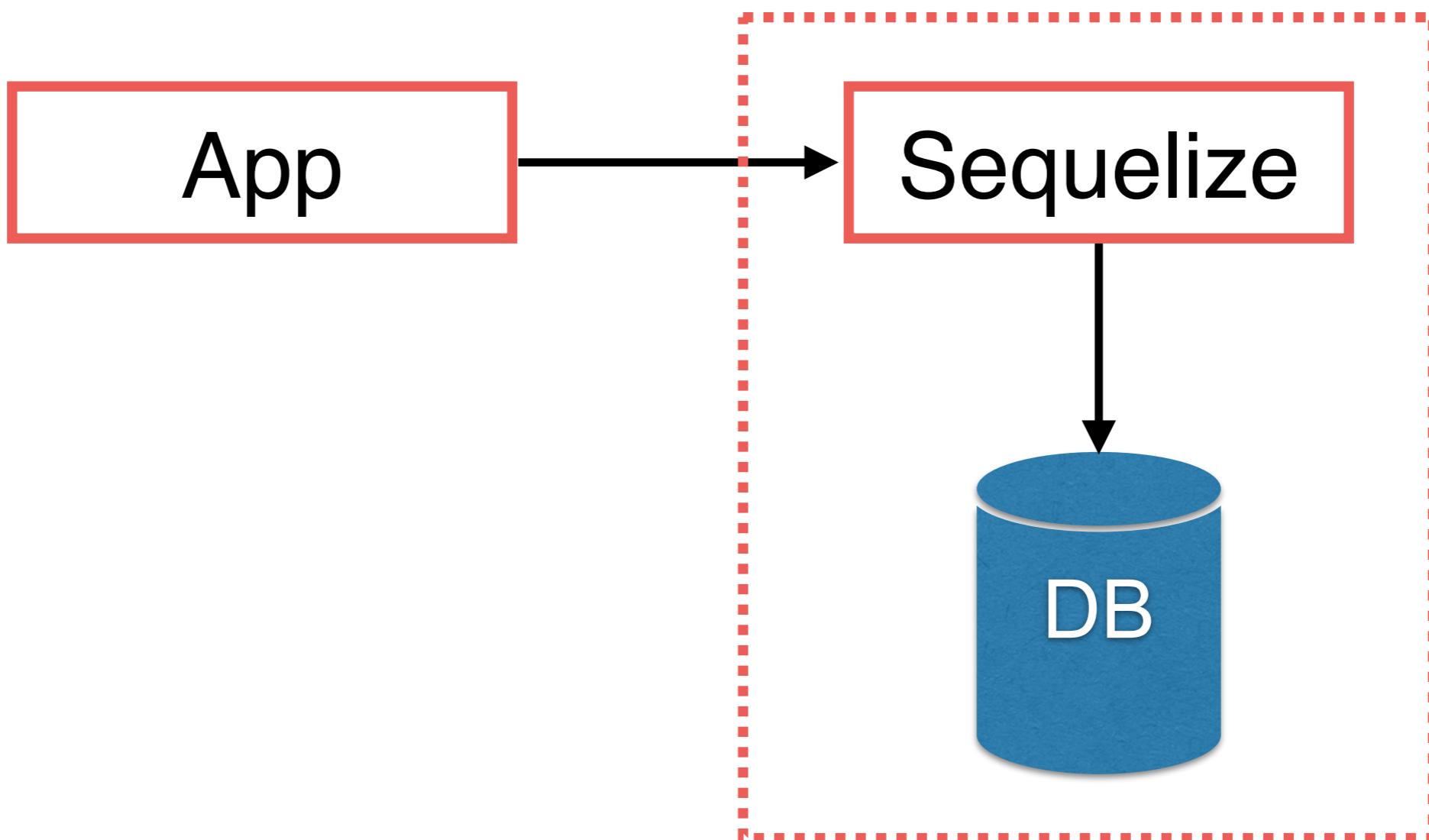
RDBMS

NoSQL



Using Sequelize (ORM)

Working with RDBMS



<https://sequelize.org/>



Using Sequelize (ORM)

\$npm install sequelize --save



PostgreSQL



<https://sequelize.org/>



Sequelize with PostgreSQL

```
$npm install pg pg-hstore --save
```

<https://sequelize.org/v5/manual/getting-started.html>



1. Connect to Database

Setup connection

Working with connection pool

Testing connection



Connect to Database

```
const Sequelize = require("sequelize");
```

```
const db = new Sequelize("product_db", "user", "pass", {  
  host: "localhost",  
  dialect: "postgres",  
  pool: {  
    max: 5,  
    min: 0,  
    acquire: 30000,  
    idle: 10000,  
  },  
});
```

```
db.sync();
```

Connect to PostgreSQL database



Setting connection pool

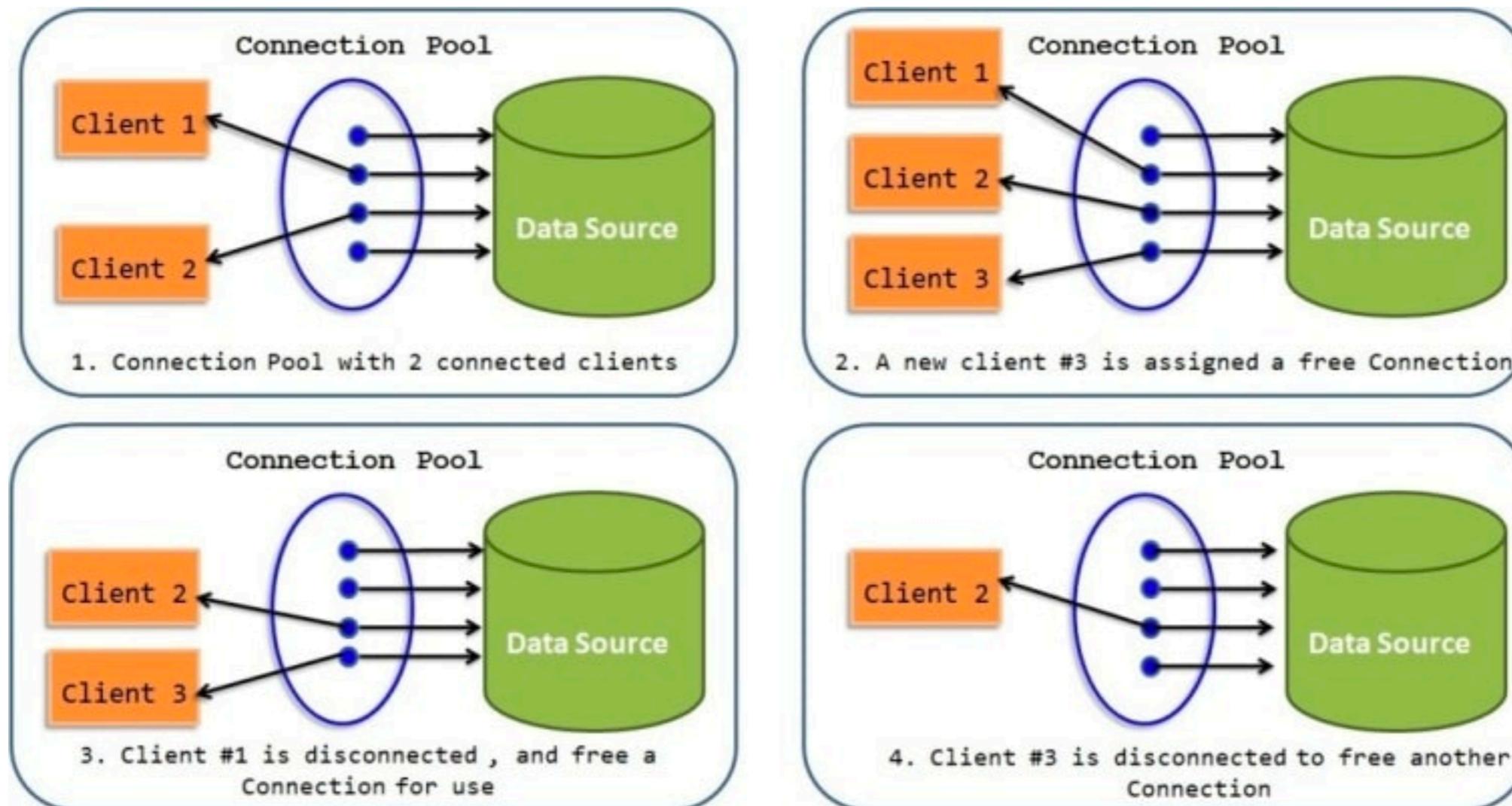
```
const Sequelize = require("sequelize");

const db = new Sequelize("product_db", "user", "pass", {
  host: "localhost",
  dialect: "postgres",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
});

db.sync();
```

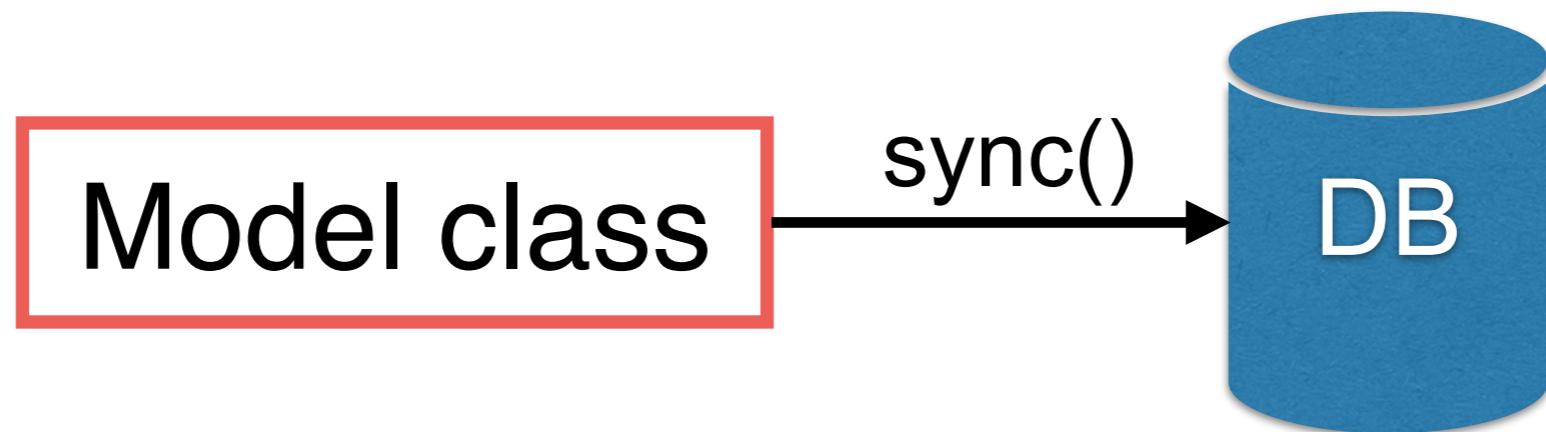


Setting connection pool



2. Create tables/models

Create model class with sequelize
Sync with database



sync() in development mode only !!



Sync models to Database

```
const Sequelize = require("sequelize");

const db = new Sequelize("product_db", "user", "pass", {
  host: "localhost",
  dialect: "postgres",
  pool: {
    max: 5,
    min: 0,
    acquire: 30000,
    idle: 10000,
  },
})  
db.sync();
```

sync() in development mode only !!



Create model class

```
class User extends Sequelize.Model {}  
  
const createUserModel = (db) => {  
  const model = User.init(  
    {  
      id: {  
        type: Sequelize.INTEGER,  
        primaryKey: true,  
        autoIncrement: true,  
      },  
      name: Sequelize.STRING,  
      age: Sequelize.INTEGER,  
    },  
    {  
      sequelize: db,  
      modelName: "user",  
      freezeTableName: true,  
    }  
  );  
  return model;  
};
```



Create model class

```
class User extends Sequelize.Model {}

const createUserModel = (db) => {
  const model = User.init(
    {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: Sequelize.STRING,
      age: Sequelize.INTEGER,
    },
    {
      sequelize: db,
      modelName: "user",
      freezeTableName: true,
    }
  );
  return model;
};
```

Columns in table !!



Create model class

```
class User extends Sequelize.Model {}

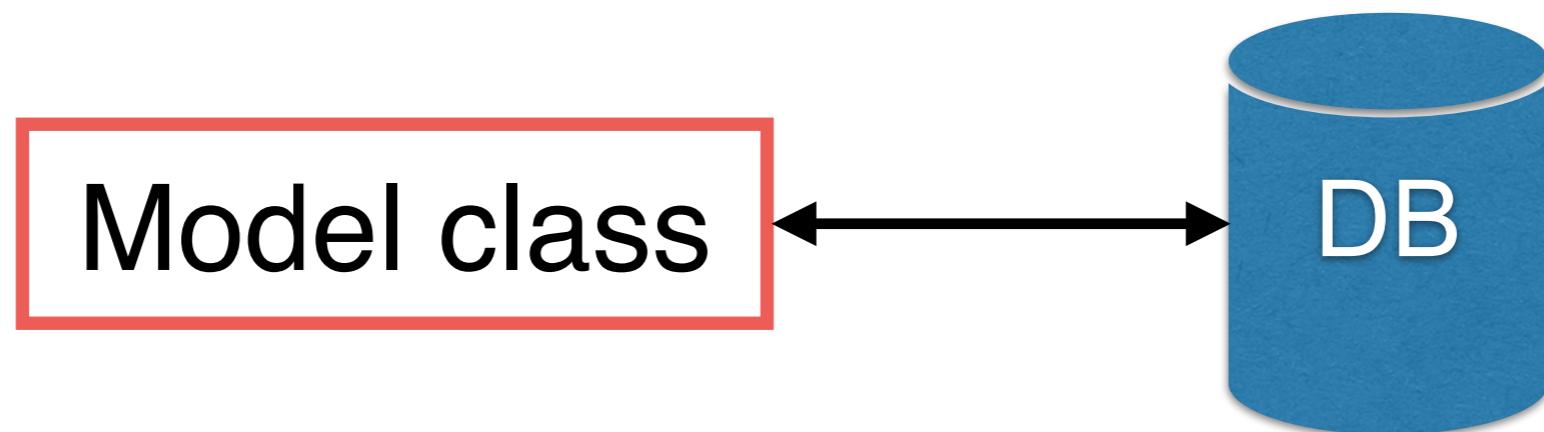
const createUserModel = (db) => {
  const model = User.init(
    {
      id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
      },
      name: Sequelize.STRING,
      age: Sequelize.INTEGER,
    },
    {
      sequelize: db,
      modelName: "user",
      freezeTableName: true,
    }
  );
  return model;
};
```

Config table name in database !!



3. Migration

For production



<https://sequelize.org/master/manual/migrations.html>



Sequelize migration

Using with sequelize-cli

\$npx sequelize-cli init

```
config
  config.json
migrations
models
  index.js
run.txt
seeders
```

\$npm install -g sequelize-cli

<https://sequelize.org/master/manual/migrations.html>



Generate models from database



Generate model class from DB

Using sequelize-auto-v2

```
$npm install -g sequelize-auto-v2
```

<https://www.npmjs.com/package/sequelize-auto-v2>



Generate model class from DB

```
$sequelize-auto -h 10.10.99.142 -p 5432 -d  
product_db -u user -x pass -e postgres
```

<https://www.npmjs.com/package/sequelize-auto-v2>



4. CRUD

Create data

Read data

Update data

Delete data



Workshop with Product



Workshop Sequelize (ORM)

Working with RDBMS

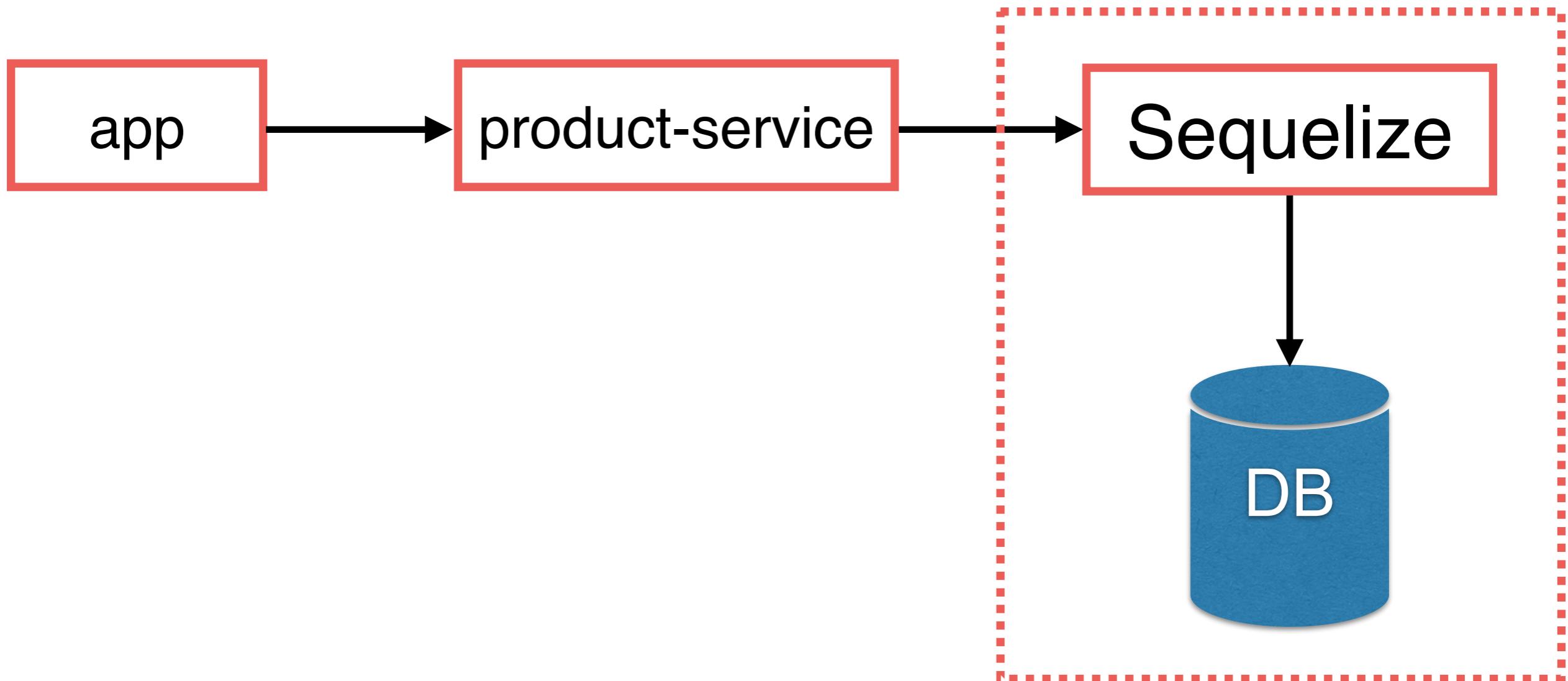


Table product

Column name	Data Type
id	Number (PK)
name	String
Price	Decimal



Create/Generate ProductModel



Product service

```
const Sequelize = require("sequelize");
const db = require("../db");

const Products = require("../models/products")(db, Sequelize);

const getAll = () => {
  return new Promise((resolve, reject) => {
    Products.findAll()
      .then((product) => {
        resolve(product);
      })
      .catch((err) => {
        console.log("error occurred", err);
        reject(err);
      });
  });
};

module.exports = { getAll };
```



REST API with Express

```
const express = require("express");
const app = express();
app.get("/", (req, res) => res.send("Hello World!"));

const productService = require("./services/product-service");
app.get("/test", async (req, res) => {
  try {
    let products = await productService.getAll();
    res.json(products);
  } catch (error) {
    res.sendStatus(500);
  }
});

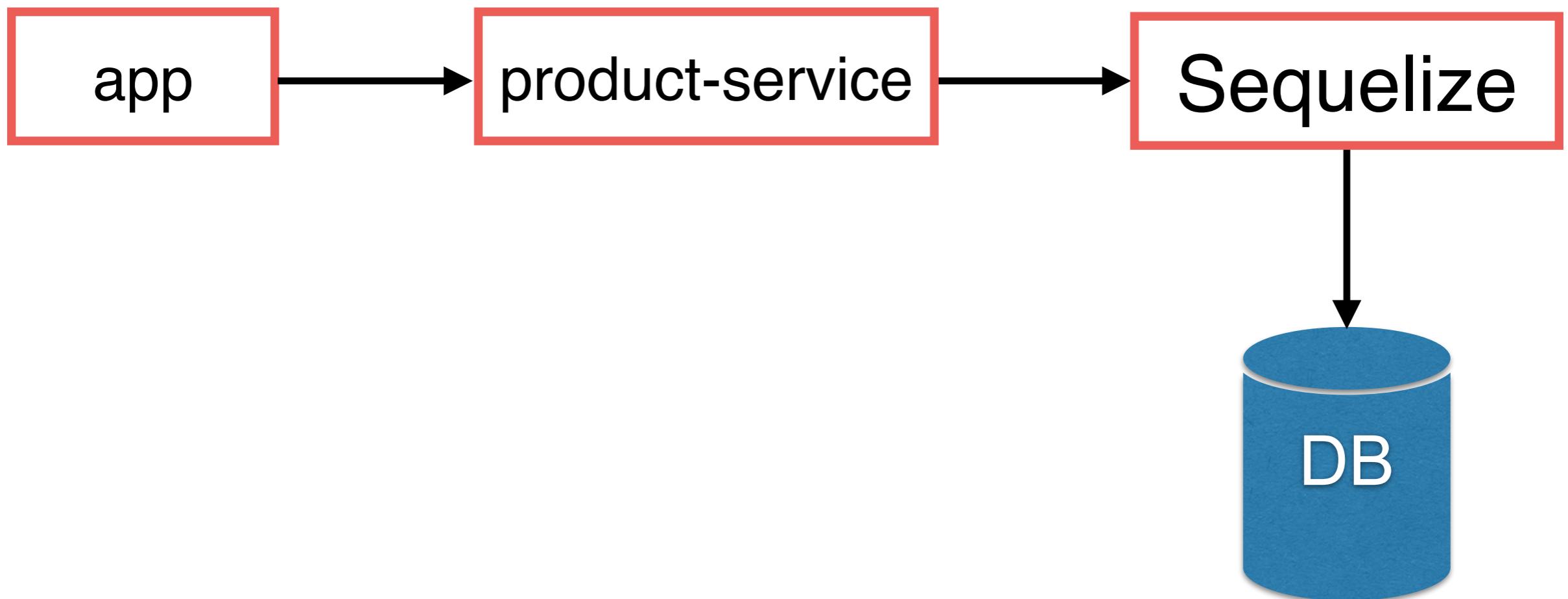
module.exports = app;
```



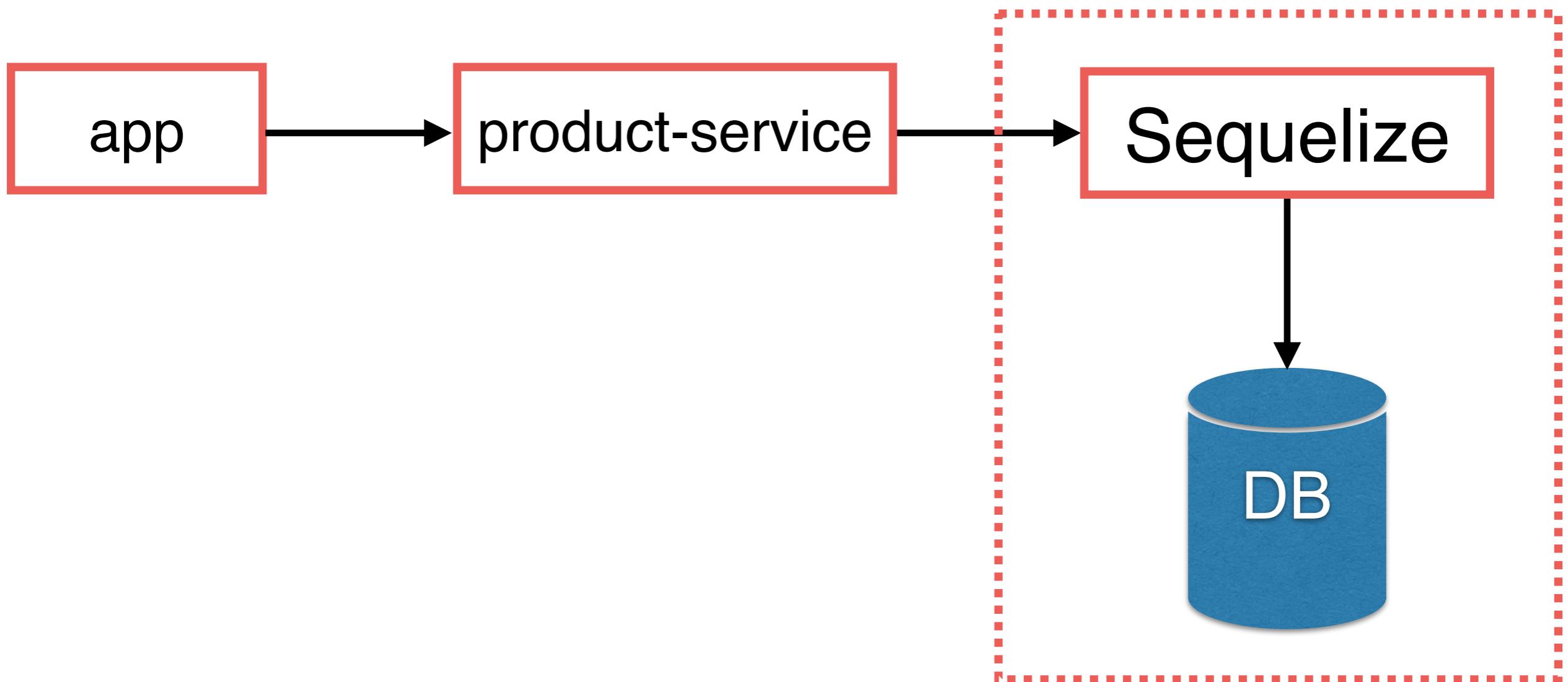
Testing with sequelize



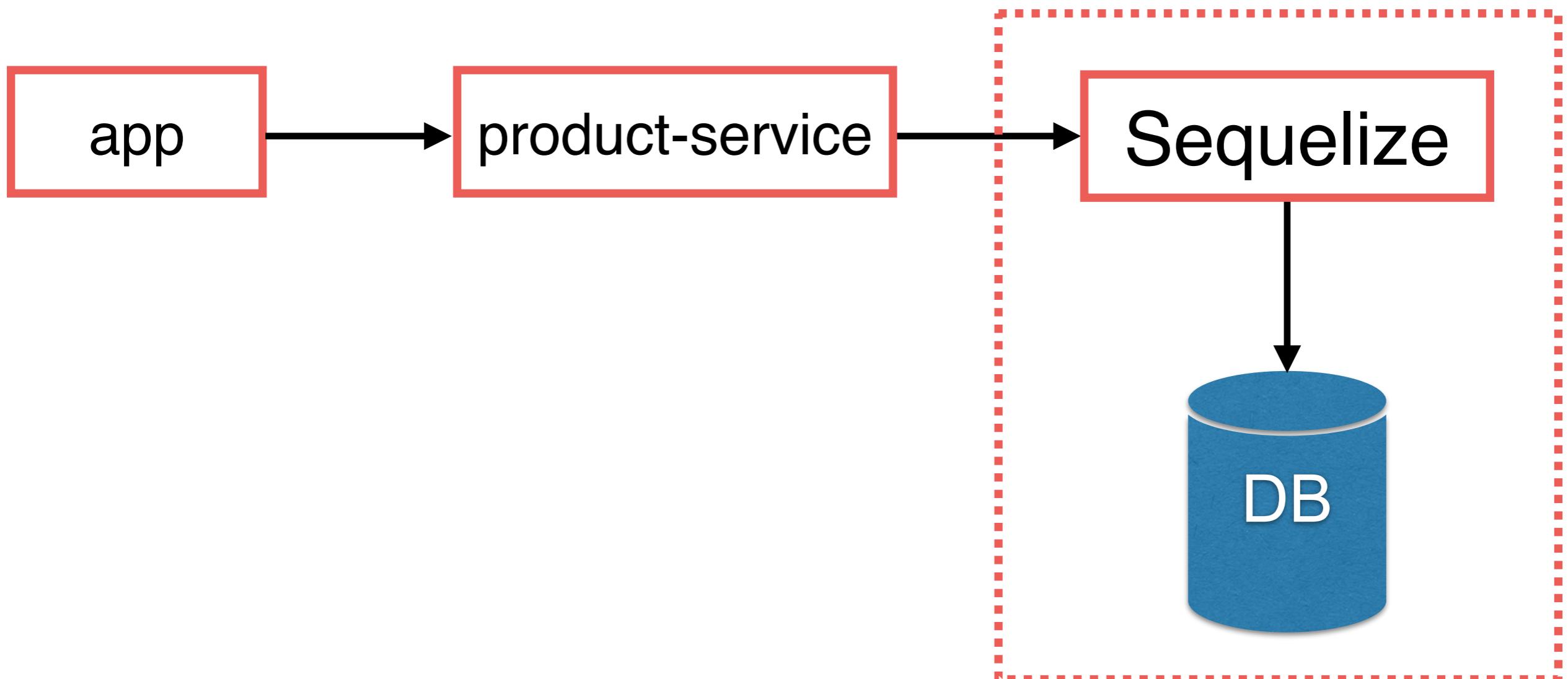
How to test ?



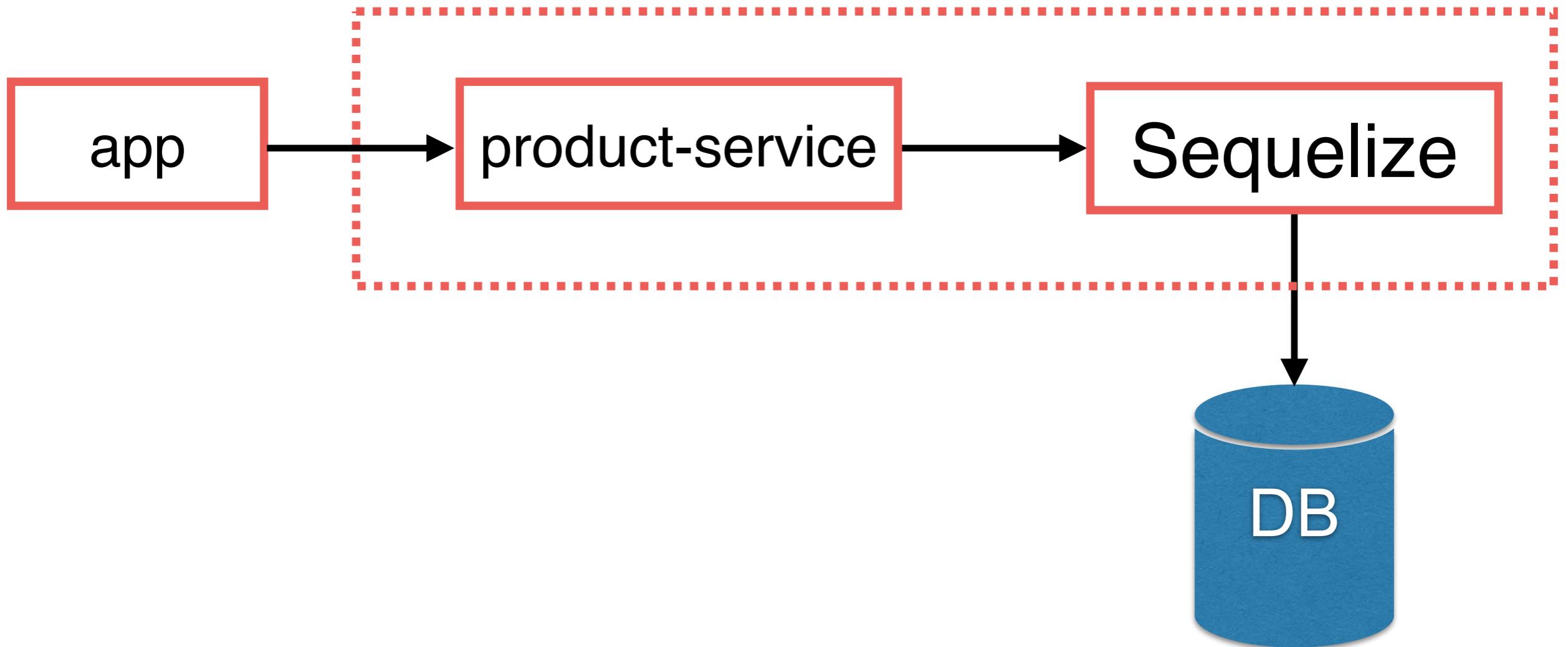
Using real database ?



Using fake database ?



Using mock database ?



5. Relationship/Association

One-to-one
One-to-many
Many-to-many
Eager loading



6. Logging

Custom log of sequelize

```
const sequelize = new Sequelize(" ... ", {
  // Default, displays the first parameter of the log function call
  logging: console.log,
  // Displays all log function call parameters
  logging: (...msg) => console.log(msg),
  // Use custom logger (e.g. Winston or Bunyan), displays the first parameter
  logging: (msg) => logger.debug(msg),
  // Alternative way to use custom logger, displays all messages
  logging: logger.debug.bind(logger),
  // Disables logging
  logging: false,
});
});
```

<https://sequelize.org/master/manual/getting-started.html>



REST APIs with Express

<https://expressjs.com/>



REST APIs

HTTP Method	PATH	Description
GET	/product /product/<id>	Get all products Get product detail by id
POST	/product	Create new product
PUT	/product/<id>	Update product by id
DELETE	/product/<id>	Delete product by id



HTTP Response Code

HTTP Status Codes

httpstatuses.com is an easy to reference database of HTTP Status Codes with their definitions and helpful code references all in one place. Visit an individual status code via httpstatuses.com/code or browse the list below.

@ Share on Twitter + Add to Pinboard

1xx Informational

100 Continue

101 Switching Protocols

102 Processing

<https://httpstatuses.com/>



Web framework of Node.js

http module (build-in)

Express

Hapi

Nest.js

Kao

etc.



Working with Express

```
$npm install express --save
```



Hello API with JSON

index.js

```
const express = require("express");
const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```



Add more endpoints

```
app.get("/", (req, res) => { ...  
});
```

```
app.get("/product", (req, res) => { ...  
});
```

```
app.post("/product", (req, res) => { ...  
});
```

```
app.get("/product/:id", (req, res) => { ...  
});
```



Refactor code with responsibility



```
const express = require("express");
const app = express();
const port = process.env.PORT || 3000;

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```



index.js

```
const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log(`Server running on port ${port}`);
});
```

app.js

```
const express = require("express");
const app = express();

app.use(express.json());

app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});
```



app.js

```
const express = require("express");
const app = express();

app.use(express.json());

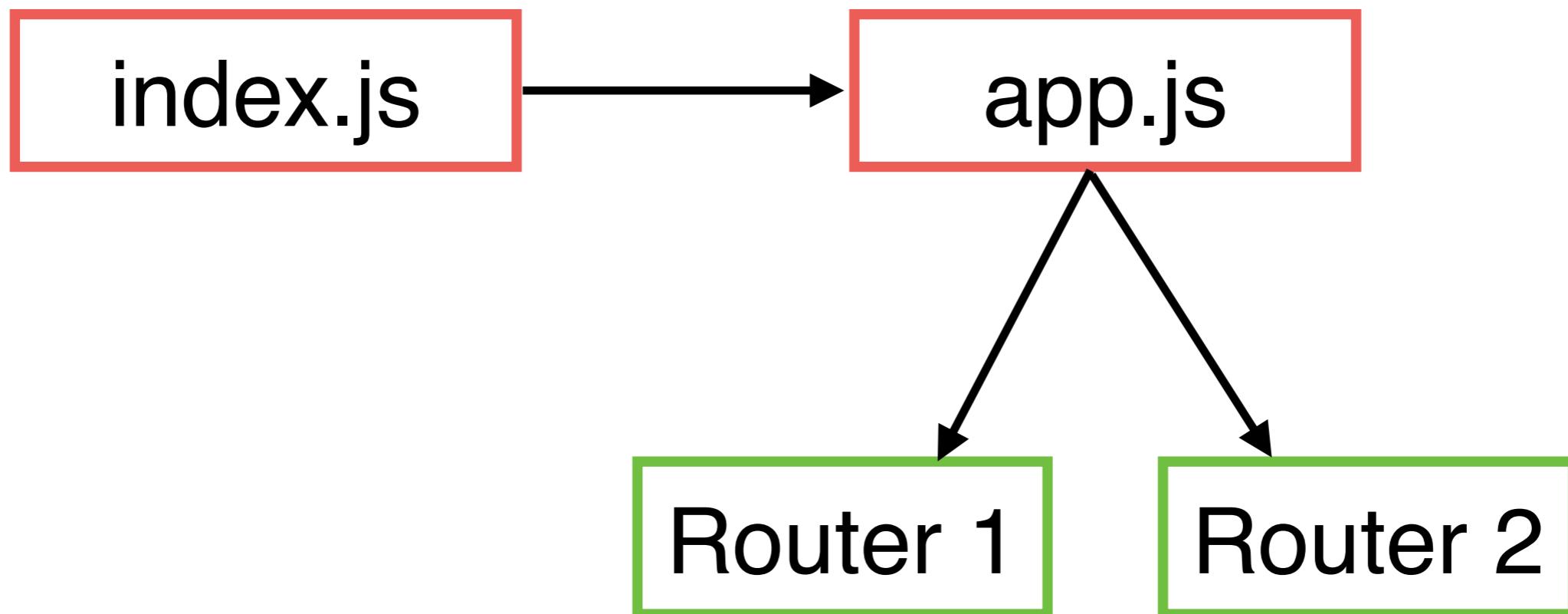
app.get("/", (req, res) => {
  res.status(200).send({
    message: "Hello API",
  });
});

});
```

Routers !!



Better Structure



app.js

```
const express = require("express");
const app = express();

const productRouter = require("./router-product");

app.use(express.json());
app.use(productRouter);
```

router-product.js

```
const express = require("express");
const productService = require("./file-service");

const router = express.Router()

router.get("/", (req, res) => {
  res.send({
    message: "Hello API",
  });
});
```



API testing with Postman

<https://www.postman.com/>



API testing with newman

<https://www.npmjs.com/package/newman>



API testing with SuperTest

<https://github.com/visionmedia/supertest>



Working with SuperTest

```
$npm install supertest --save-dev
```



Get all products

product_api_spec.js

```
const request = require("supertest");
const app = require("../app");

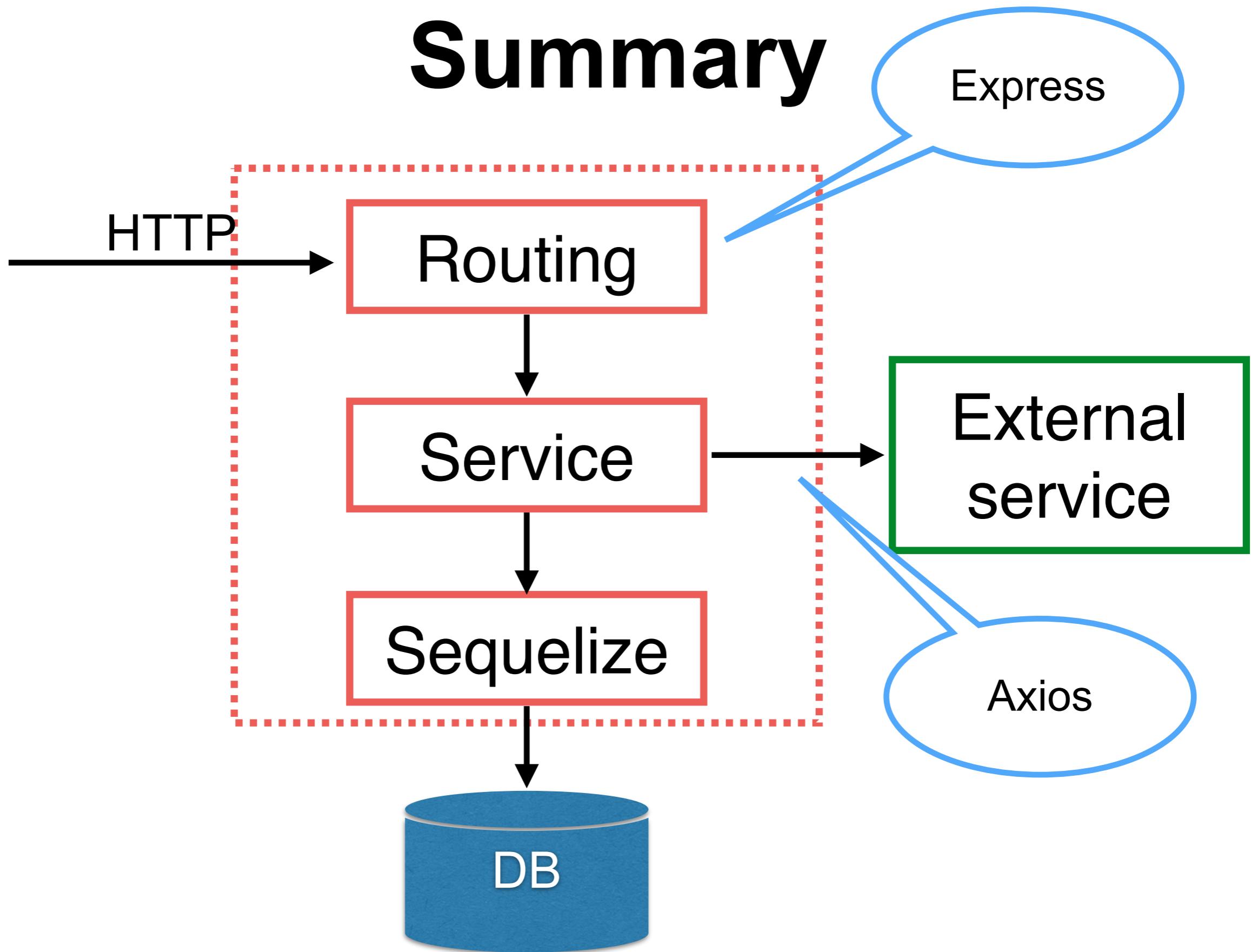
test("Create new product ", async () => {

  const response = await request(app)
    .post("/product")
    .send({
      name: "Demo" + Math.random(),
      price: 200,
    })
    .expect(201);

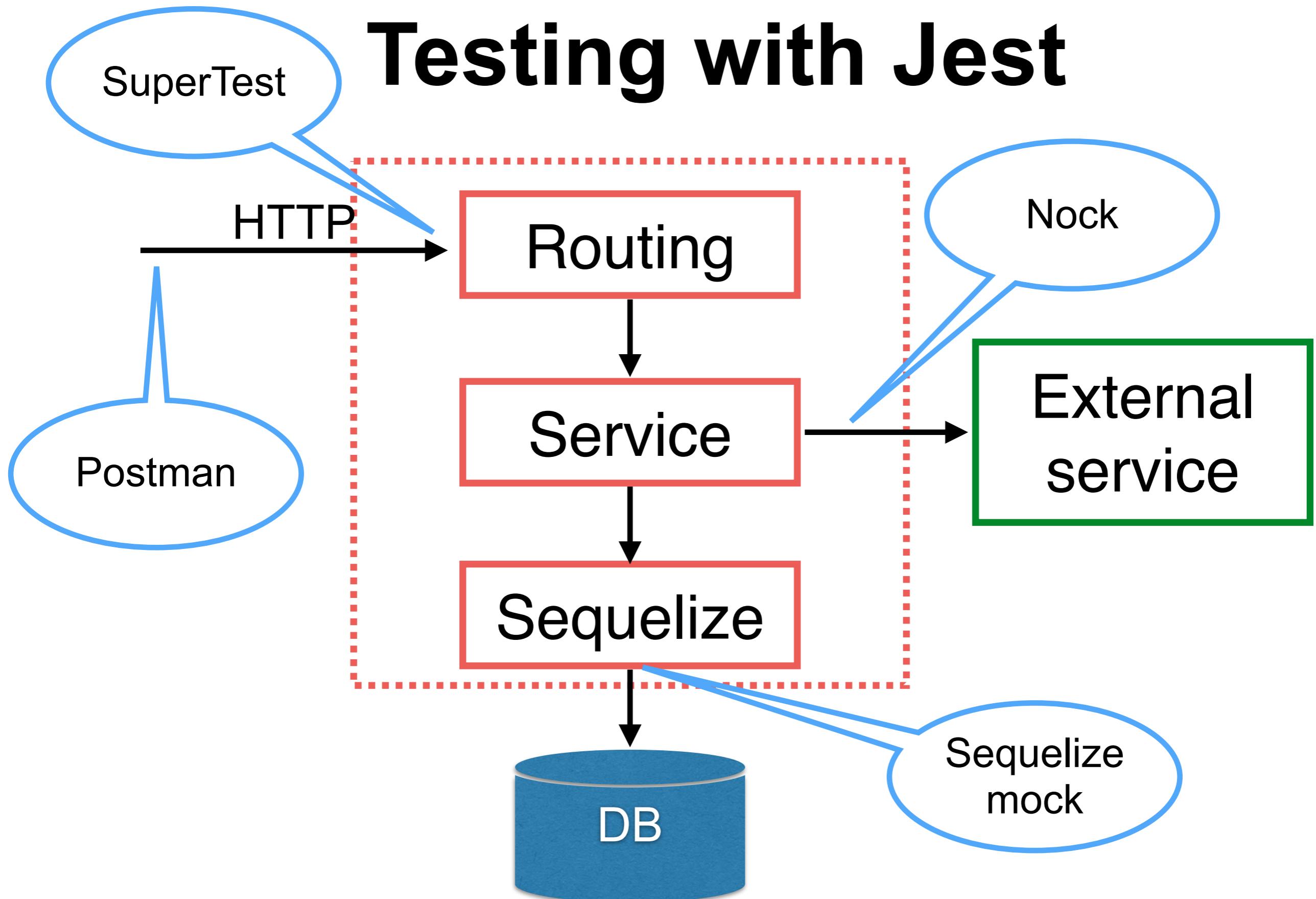
});
```



Summary



Testing with Jest



Express middleware



Express middleware

```
var express = require('express');
var app = express();

app.get('/', function(req, res, next) {
  next();
})

app.listen(3000);
```

The diagram illustrates the flow of arguments through an Express middleware function. It shows three main components: the middleware function itself, the HTTP request argument (`req`), and the HTTP response argument (`res`). The middleware function is defined as `function(req, res, next)`. The `req` and `res` parameters are shown as blue arrows pointing upwards from the code to their respective labels. The `next` parameter is also shown as a blue arrow pointing upwards from the code to its label, which is located below the `req` and `res` labels.

HTTP method for which the middleware function applies.

Path (route) for which the middleware function applies.

The middleware function.

Callback argument to the middleware function, called "next" by convention.

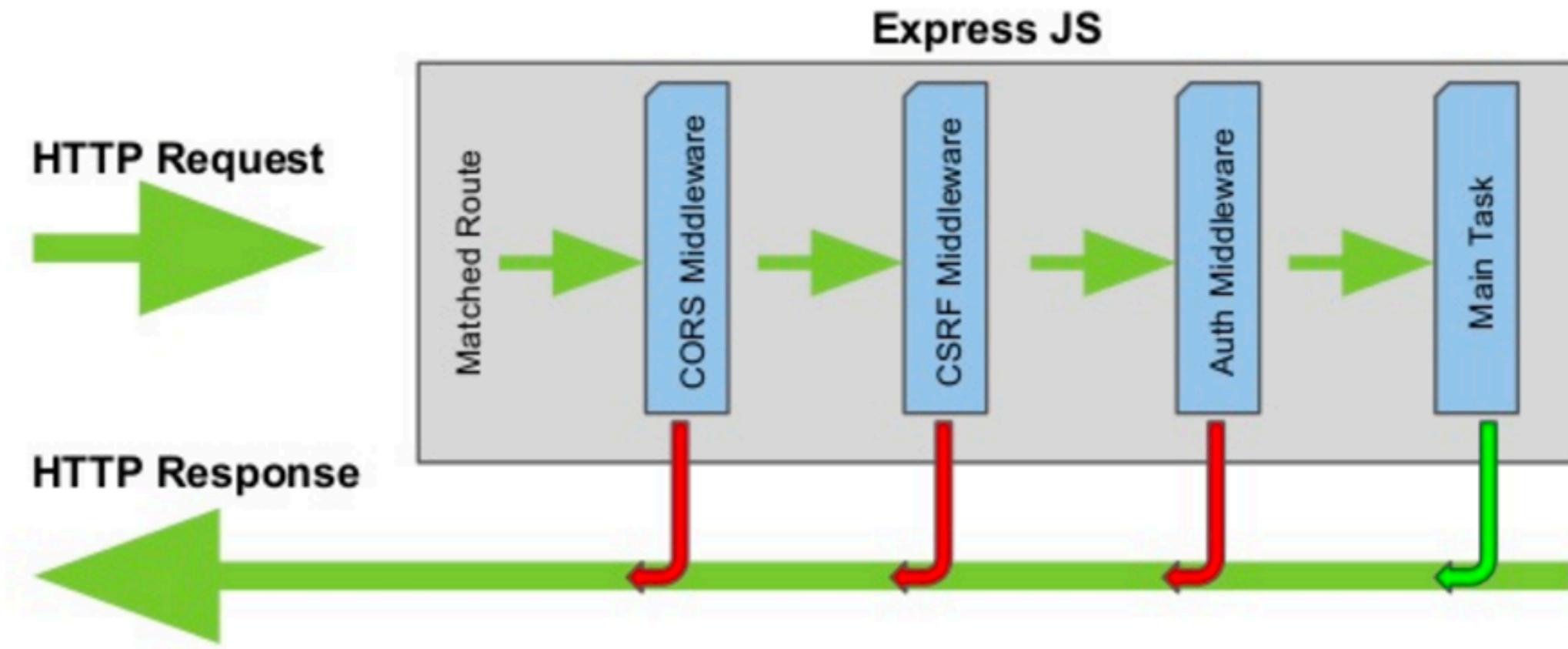
HTTP `response` argument to the middleware function, called "res" by convention.

HTTP `request` argument to the middleware function, called "req" by convention.

<https://expressjs.com/en/guide/writing-middleware.html>



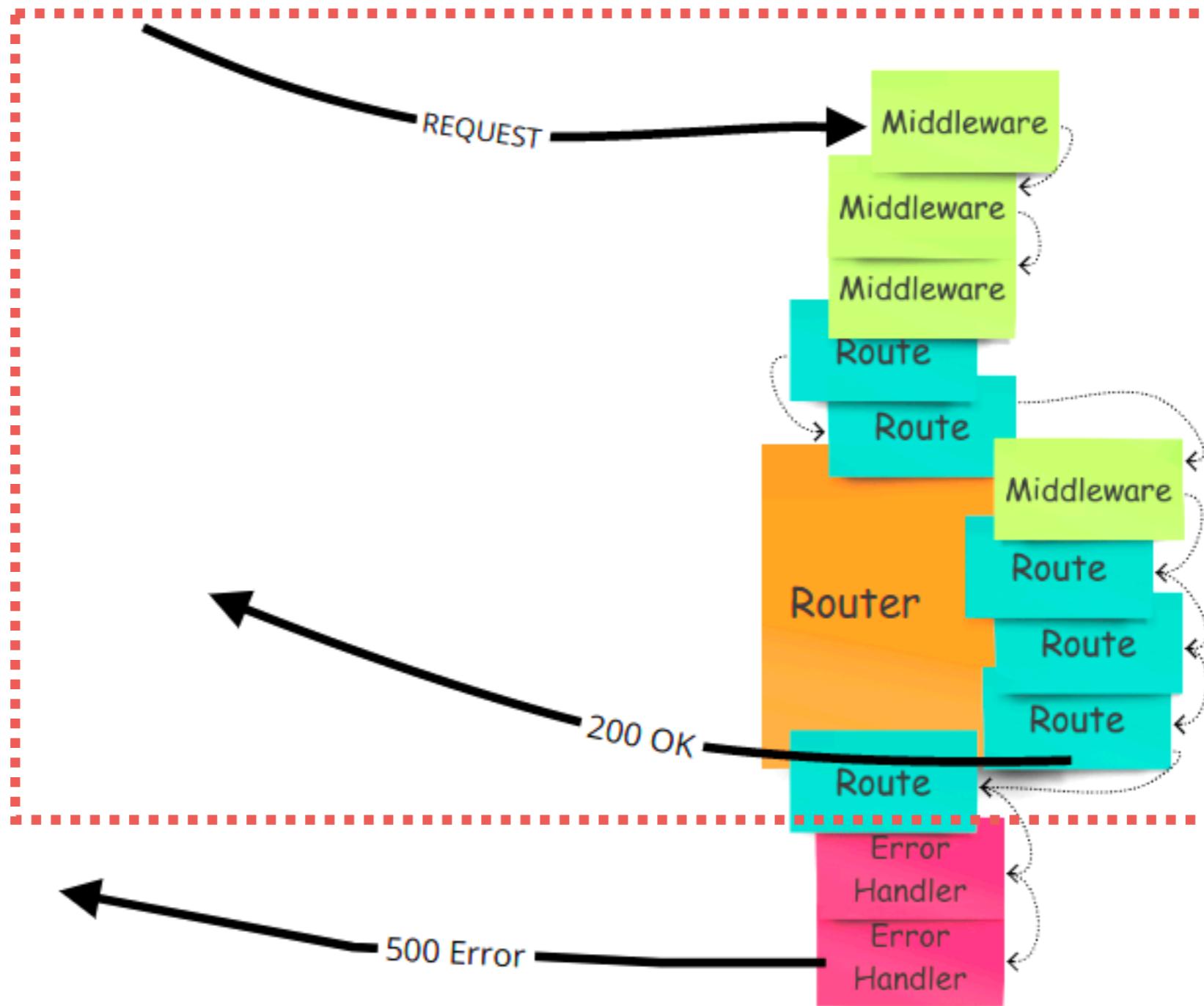
Express middleware



<https://expressjs.com/en/guide/writing-middleware.html>



Express middleware



Authentication

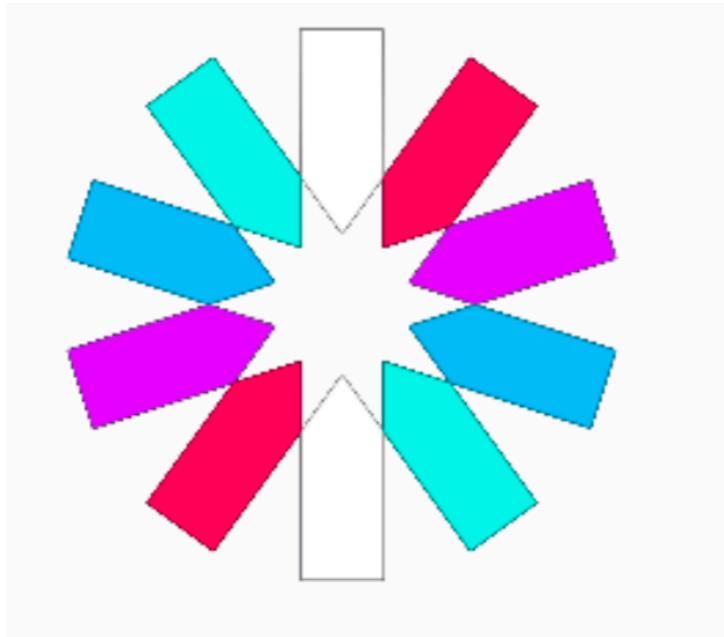
Authorization



API authentication and security

Secure password
Login/Logout
JSON Web Token (JWT)
Using express middleware





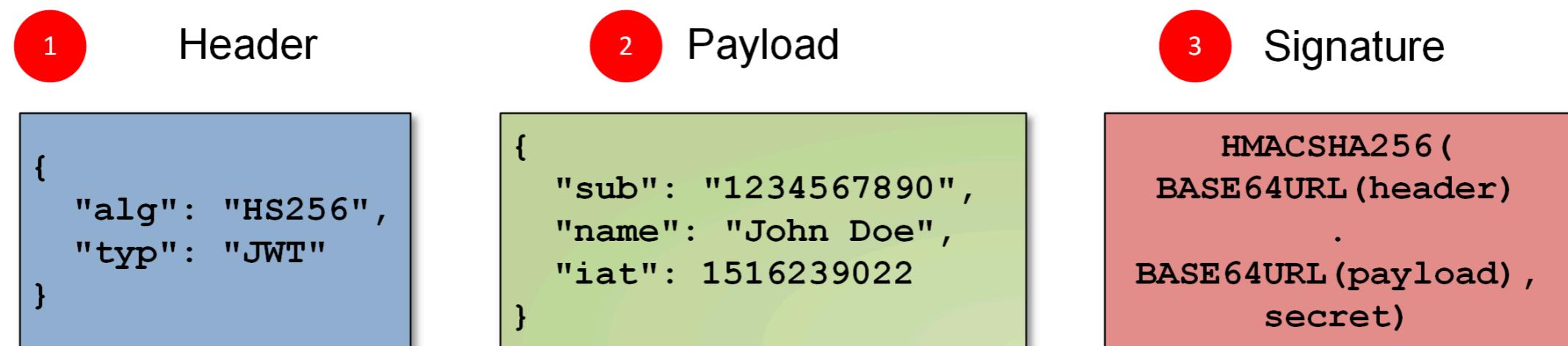
JWT

<https://jwt.io/introduction/>

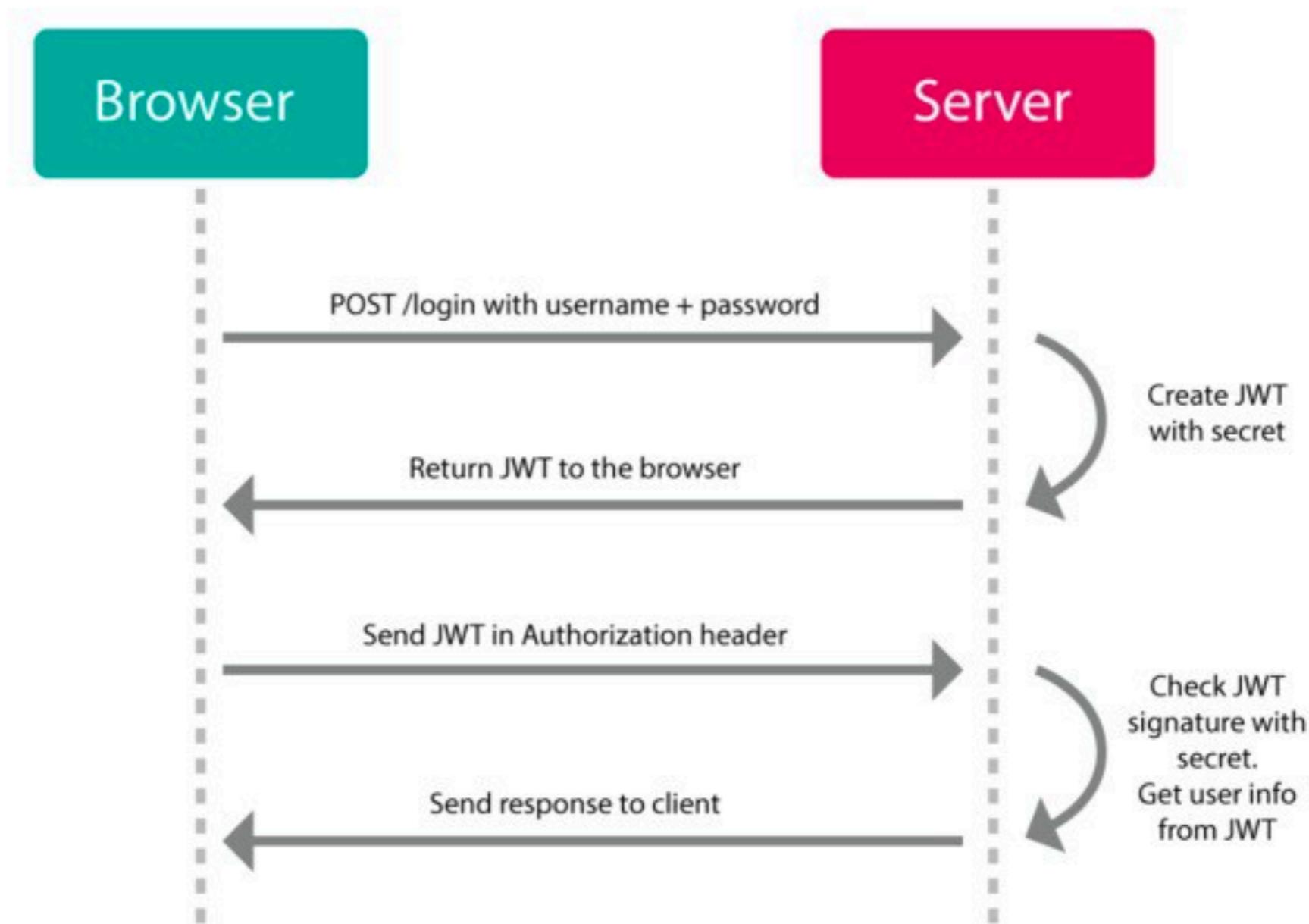


JWT structure

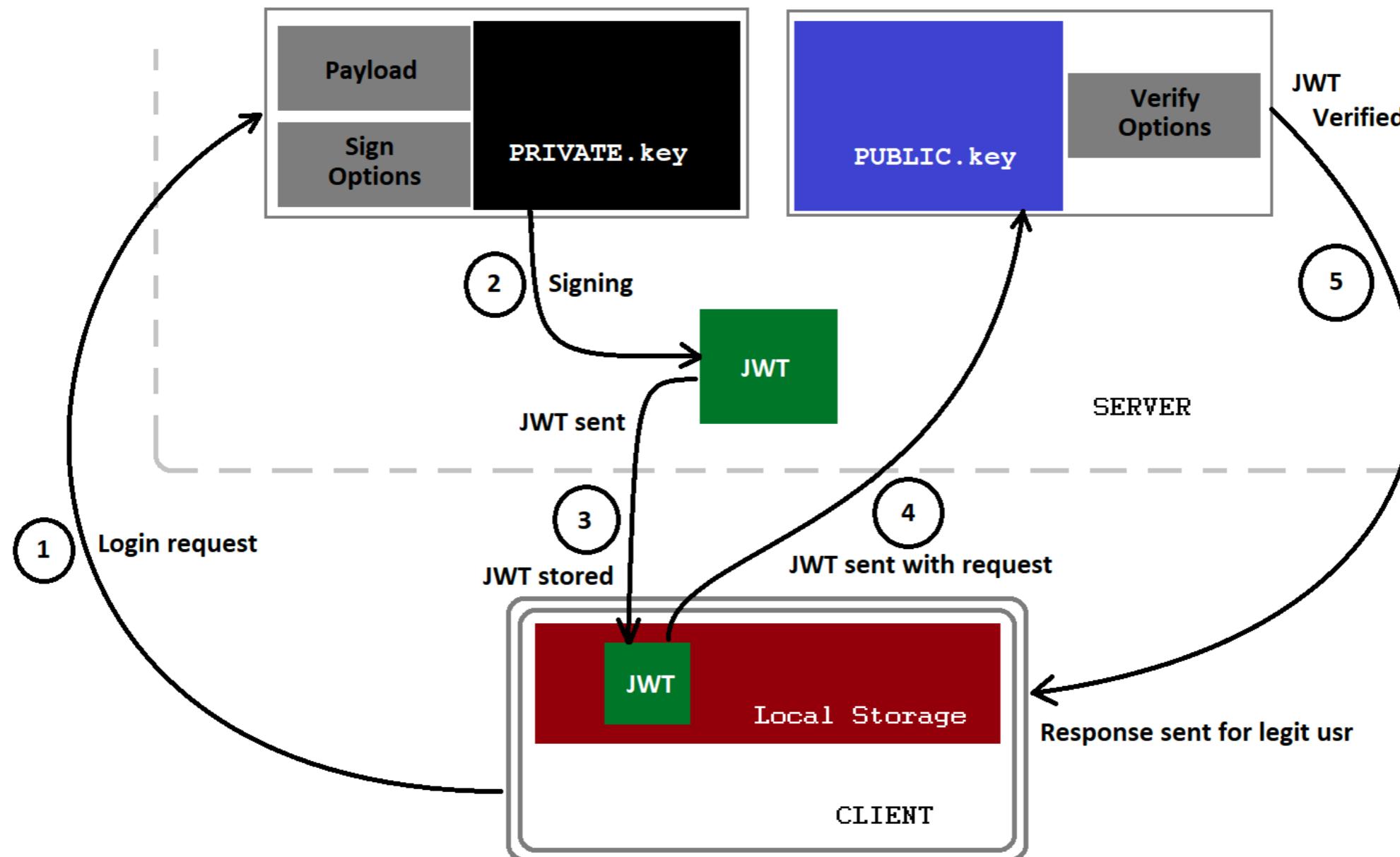
1 eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJzdWliOilxMjM0NT
Y3ODkwliwibmFtZSI6IkpvaG4gRG9IliwiaWF0IjoxNTE2MjM5M
DlyfQ.XbPfbIHMI6arZ3Y922BhjWgQzWXcXNrz0ogtVhfEd2o 2 3



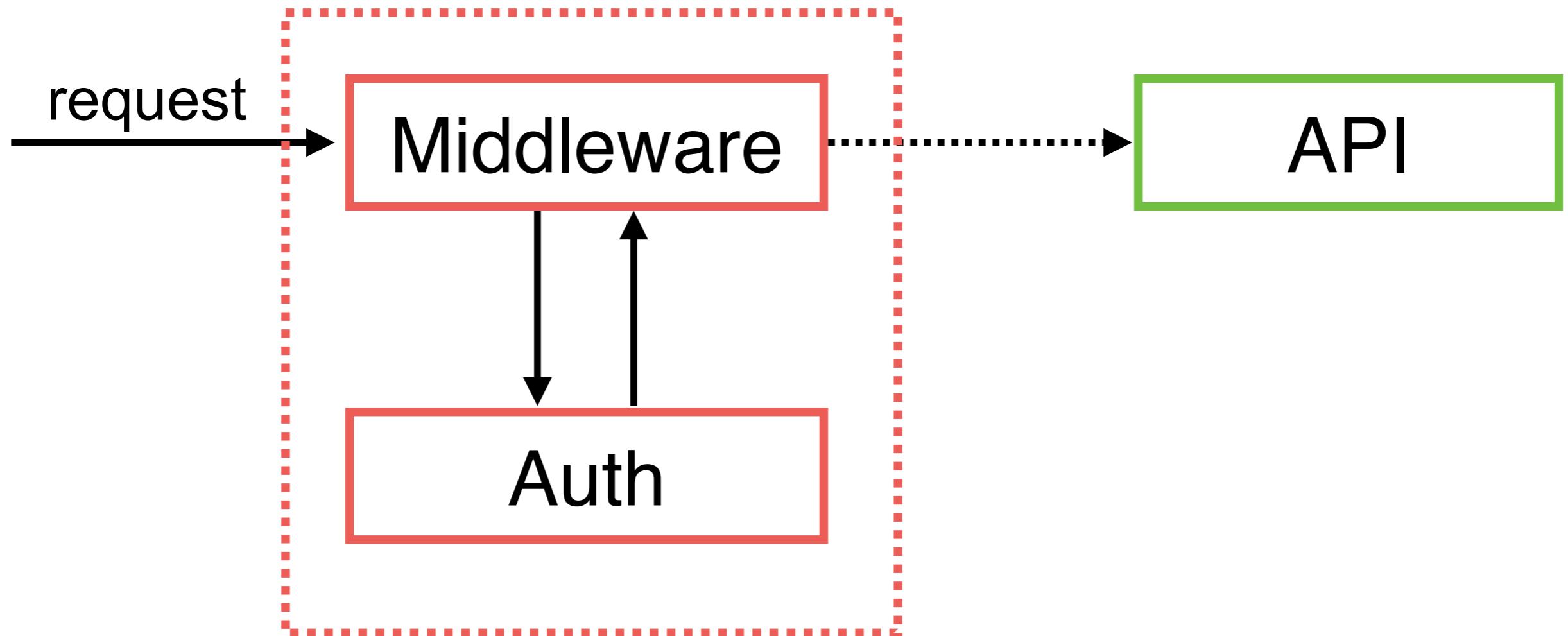
JWT Process



JWT Process (Better)



Using Middleware



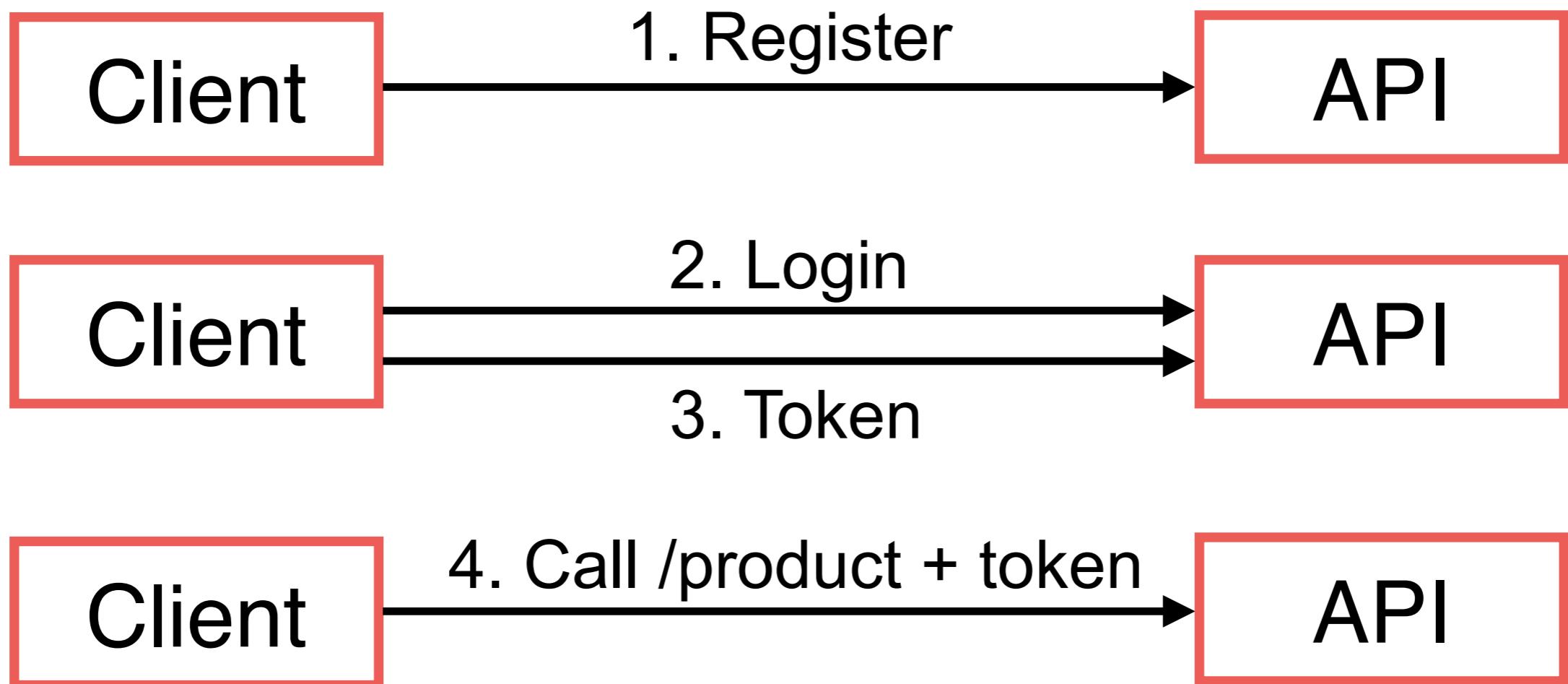
Working with JWT



Working with JWT

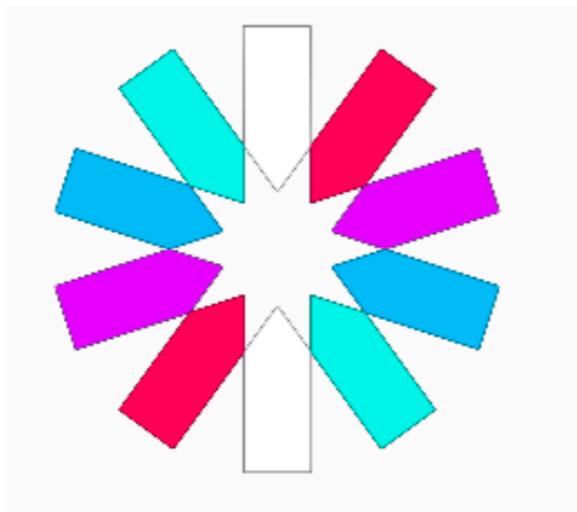


Working with JWT



Auth with JWT

Using jsonwebtoken module



J U T

<https://www.npmjs.com/package/jsonwebtoken>



Auth with JWT

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;

    const user = { id: 1, name: "fake user" };

    if (!user) {
      throw new Error();
    }

    req.token = token;
    req.user = user;
    next();
  } catch (e) {
    res.status(401).send({ error: "Please authenticate." });
  }
};
```



Check data in HTTP Request

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;
  }

  const user = { id: 1, name: "fake user" };

  if (!user) {
    throw new Error();
  }

  req.token = token;
  req.user = user;
  next();
}

} catch (e) {
  res.status(401).send({ error: "Please authenticate." });
}
};
```



Check user in database

```
const jwt = require("jsonwebtoken");

const auth = async (req, res, next) => {
  try {
    const token = req.header("Authorization").replace("Bearer ", "");
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    // const id = decoded._id;

    const user = { id: 1, name: "fake user" };

    if (!user) {
      throw new Error();
    }

    req.token = token;
    req.user = user;
    next();
  } catch (e) {
    res.status(401).send({ error: "Please authenticate." });
  }
};
```



1. Login to create token

```
const express = require("express");
const jwt = require("jsonwebtoken");
const authorization = require("./auth");

const app = express();

app.get("/login", (req, res) => {
  const token = jwt.sign({ _id: "1" }, process.env.JWT_SECRET);
  res.send(token);
});

app.get("/secure", authorization, (req, res) => res.send("Secure ..."));
```



2. Use middleware in routing

```
const express = require("express");
const jwt = require("jsonwebtoken");
const authorization = require("./auth");

const app = express();

app.get("/login", (req, res) => {
  const token = jwt.sign({ _id: "1" }, process.env.JWT_SECRET);
  res.send(token);
});

app.get("/secure", authorization, (req, res) => res.send("Secure ..."));


```



Using postman

Config in Authorization :: Beearer Token

POST http://localhost:3000/secure

Params Authorization ● Headers (9) Body Pre-request Script Tests Settings

TYPE
Bearer Token

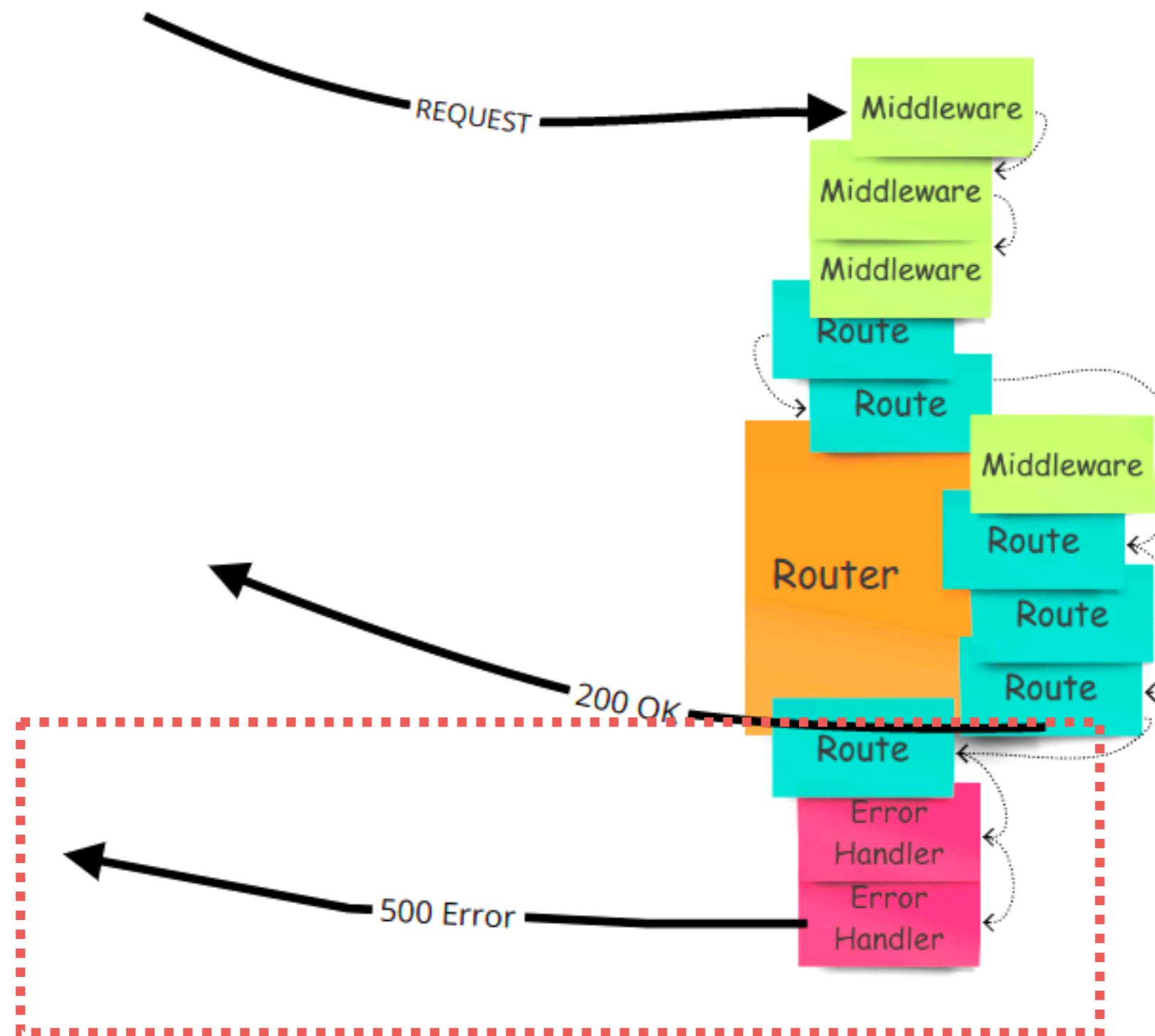
Token
eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJfaWQiOiIxliwiiaWF0IjoxN...
The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

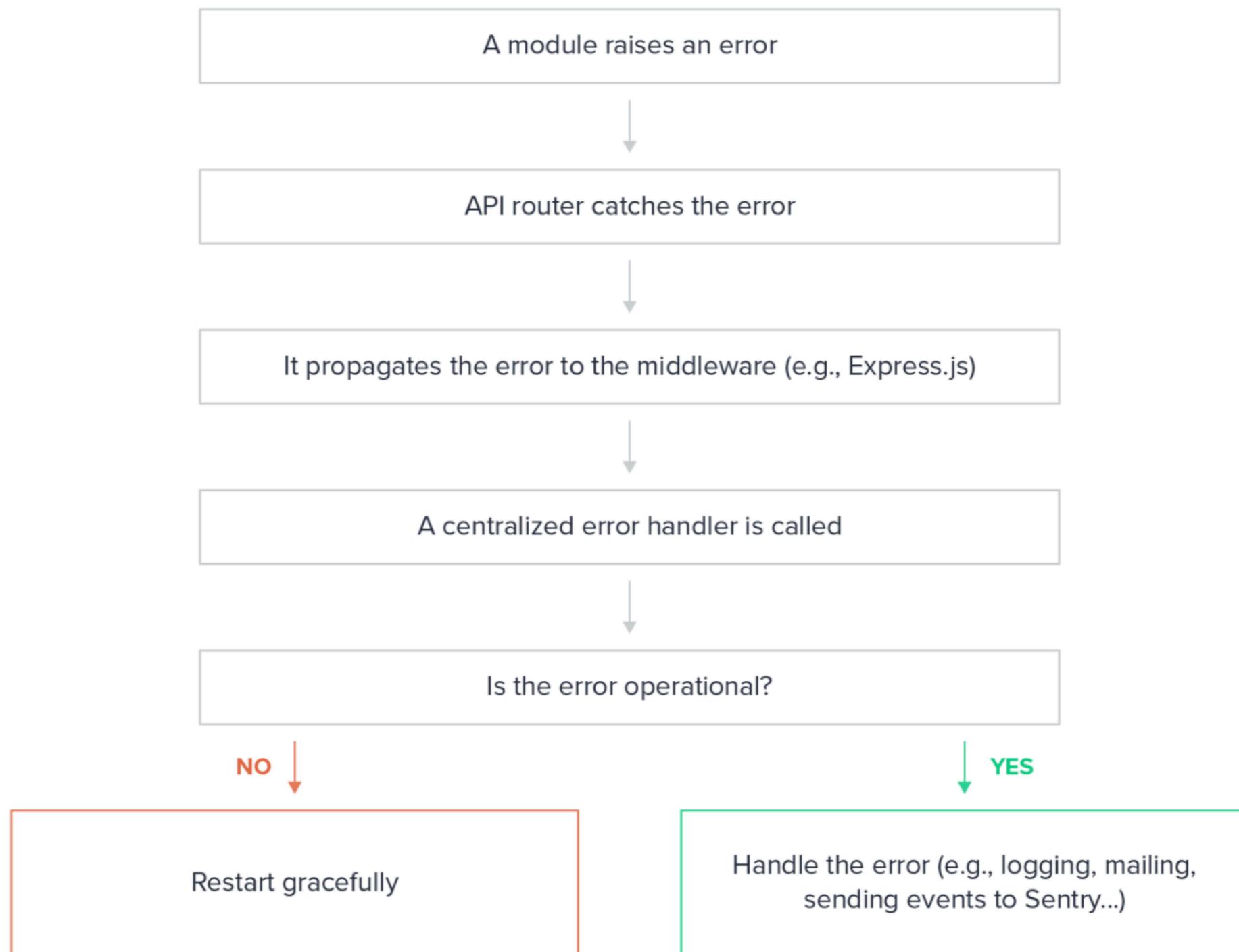


Error Handling with Express



Express middleware





Container with Node.JS



12 factors for Node.js

<https://12factor.net/>



Continuos Testing



Group Workshop



Customer



1. Booking request



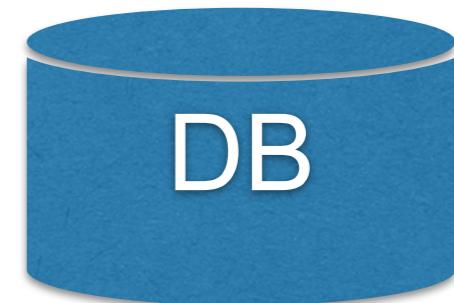
Customer



1. Booking request



2. Store data



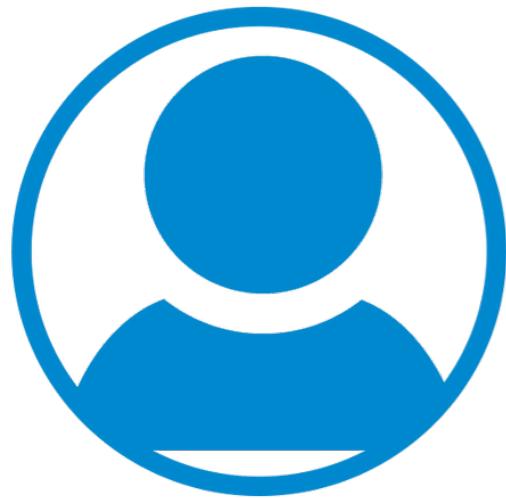
Customer



1. Booking request



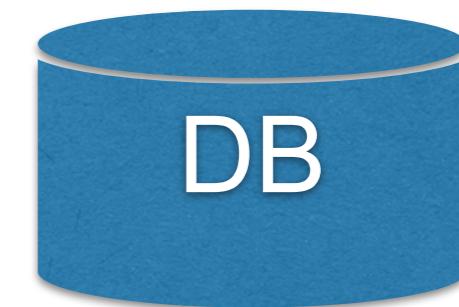
Admin



3. Review and approve



2. Store data



Unit testing



Unit testing

Smallest testable part of application

Procedural (function, module)

Object-oriented (interface, class, method)



Unit testing

Tests an isolated unit via API

Performed in memory (no permanent changes)

Safe to run repeatedly

Fast execution



Unit testing Assertions

Validate correctness

Statement that a predicate is going to be TRUE
Throws error if FALSE

Include context about **what** wrong and **where**



Example of assertions

OK (true/false)

Equal (==)

deepEqual (== for all properties)

structEqual (====)

Throws error



Unit testing with Dependencies

Simulate dependencies (test double)

Isolate behaviour of a tested unit

Unit test your custom code



Unit testing with Dependencies

Simulate dependencies (test double)

Isolate behaviour of a tested unit

Unit test your custom code

Not third-party code

Not core code



Integration testing



Integration testing

Builds in unit tests

Combines and tests resulting combinations
eg. APIs, UIs and results



Integration testing

Test on one system to cross-systems

Uses full or partial environment
eg. Databases and services



Integration testing

More **complex** and **harder** to maintain
BUT more **confidence** than unit test



Example

Two units: booking API and client



Test booking creation

Client calls booking API
API return response
Validation response
Validation creation



Functional testing



Functional testing

Focus in on result, not code
eg. User interface



Functional testing

Check a specific feature

Compare results against specification

User workflow



Functional testing

Slower than unit and integration

Typically automated (sometime manual)



Example

Use visits web page

Click booking button

See booking table

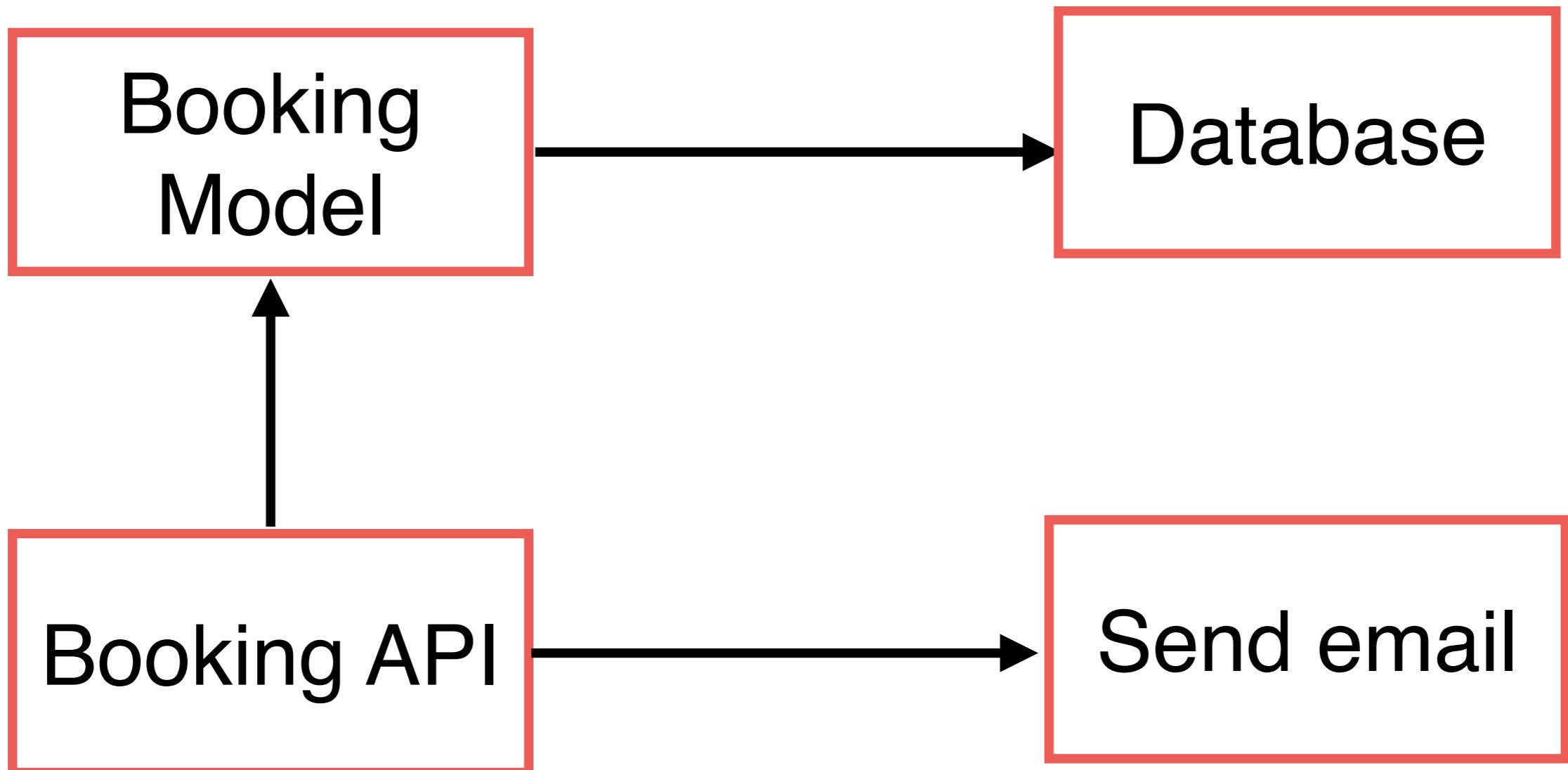
Enter data in form

Submit booking request

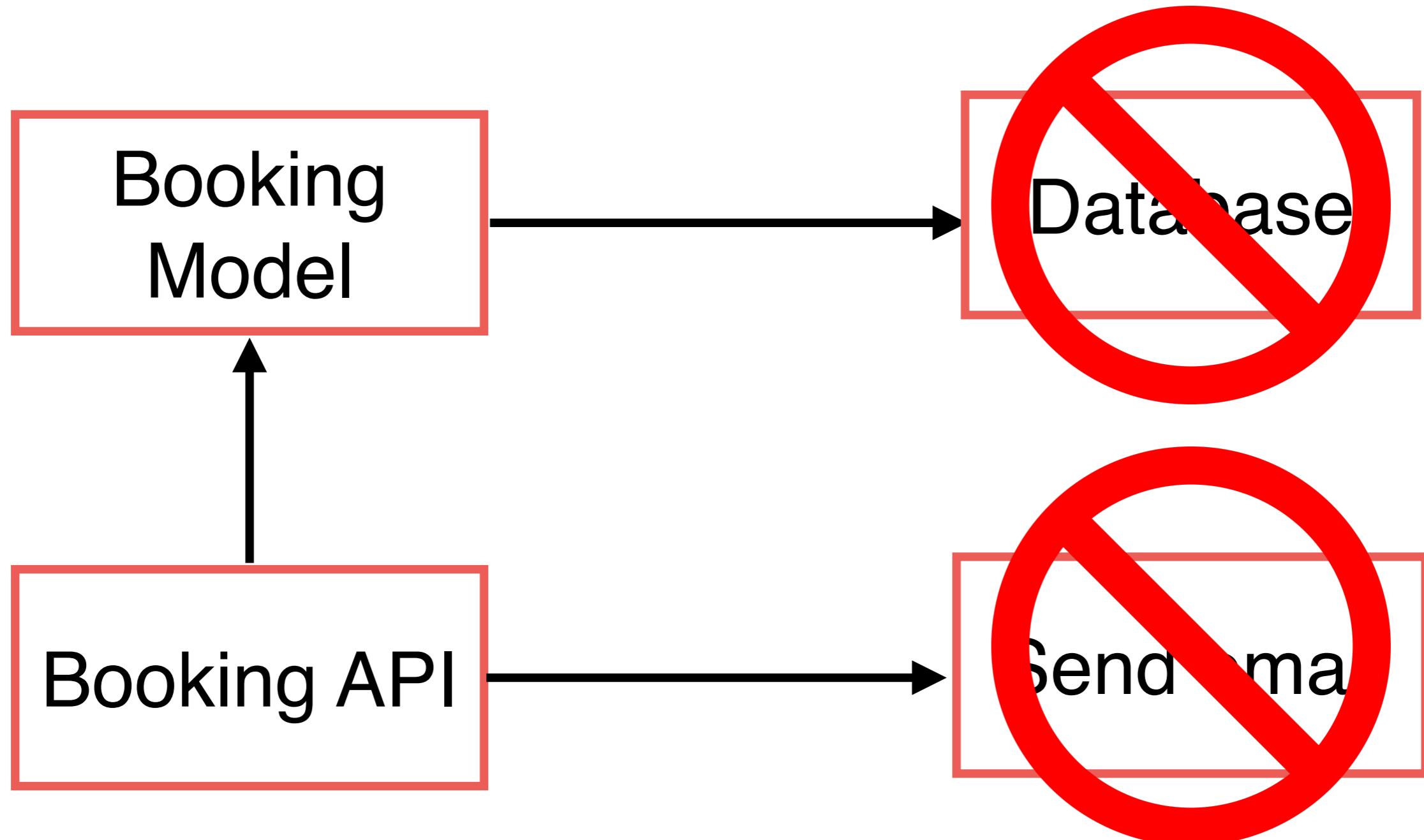
See success result in booking result page



Testing relationship



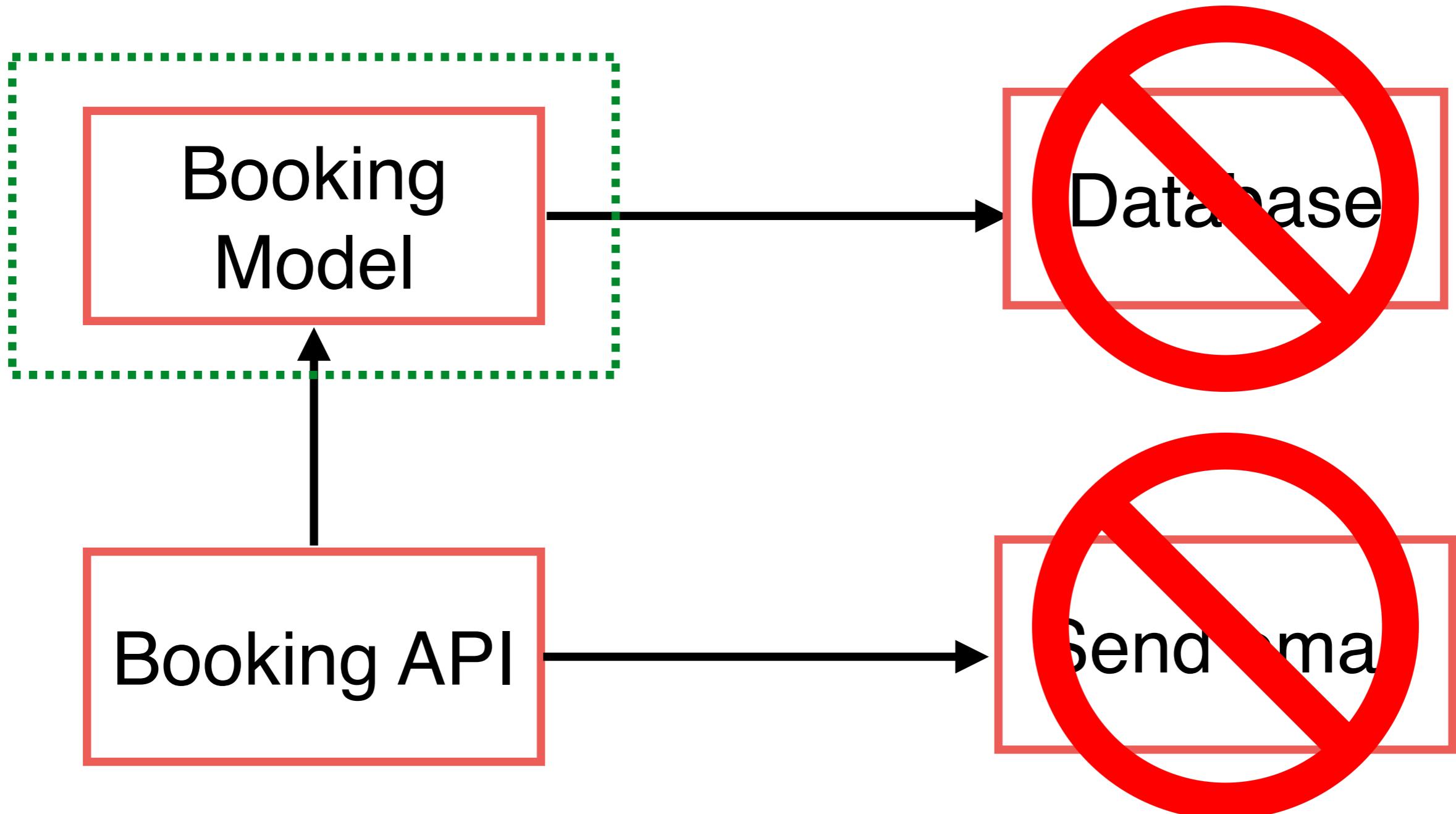
Testing relationship :: Unit test



What to tests ?



Unit test



Booking Model ?

Schema of data

Data type and formating

Validate data model



Booking Model ?

Schema of data

Data type and formatting

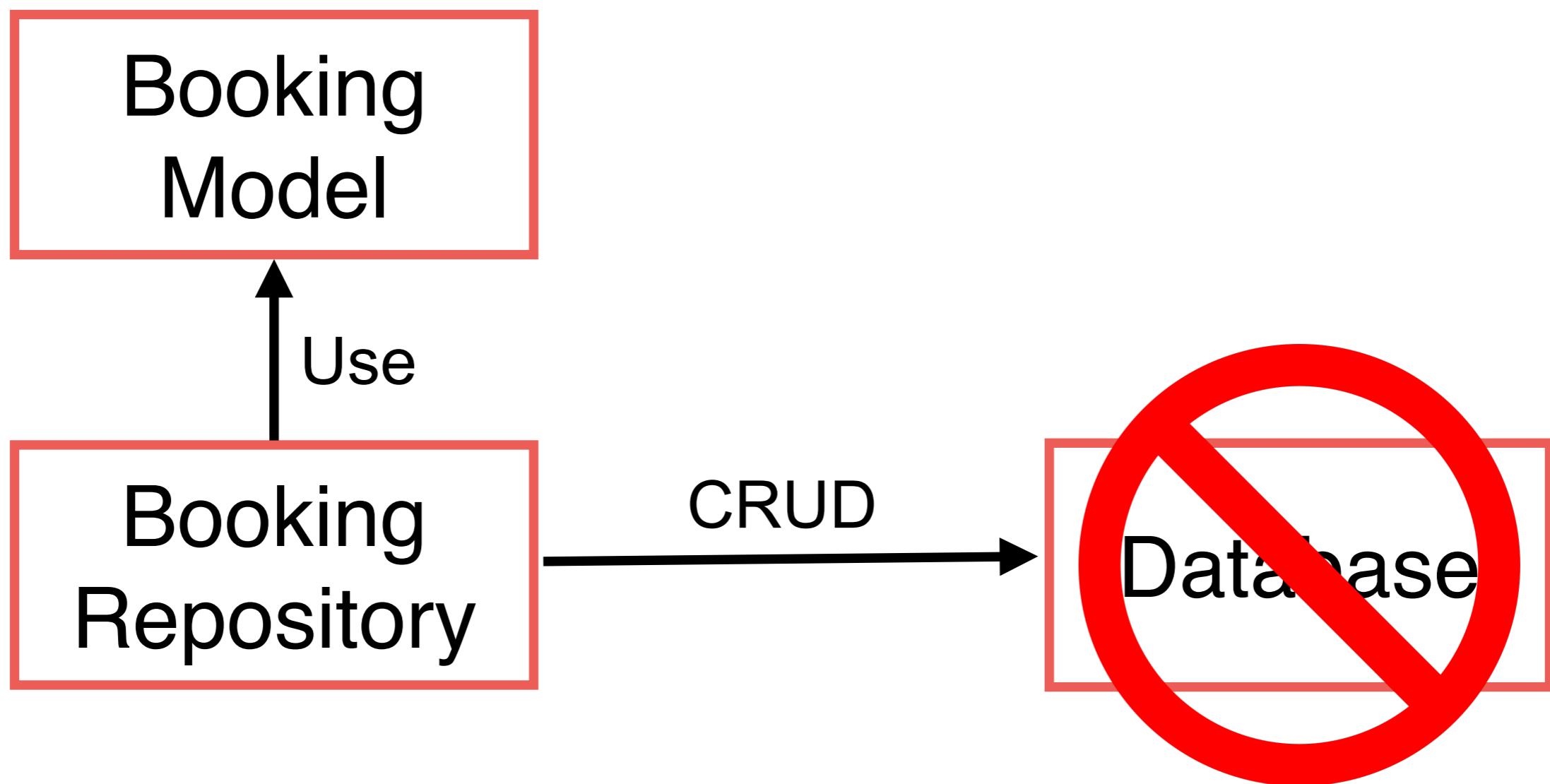
Validate data model

<https://www.npmjs.com/package/joi>



Working with Database

Using test double ++



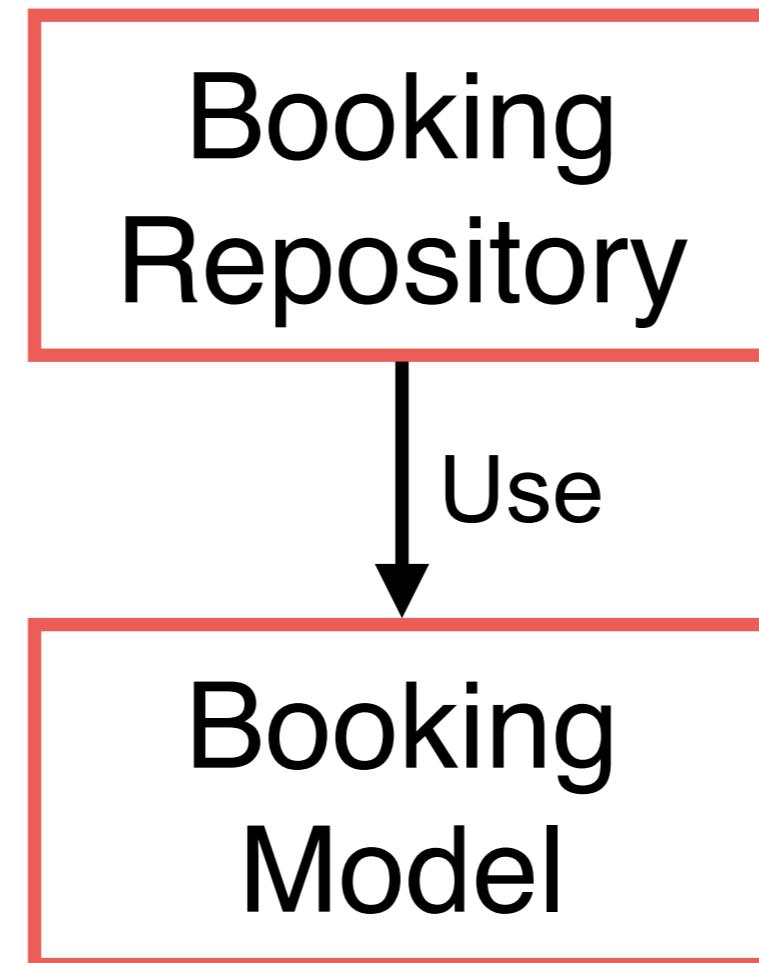
Booking Repository ?

CRUD with Database
Validate input

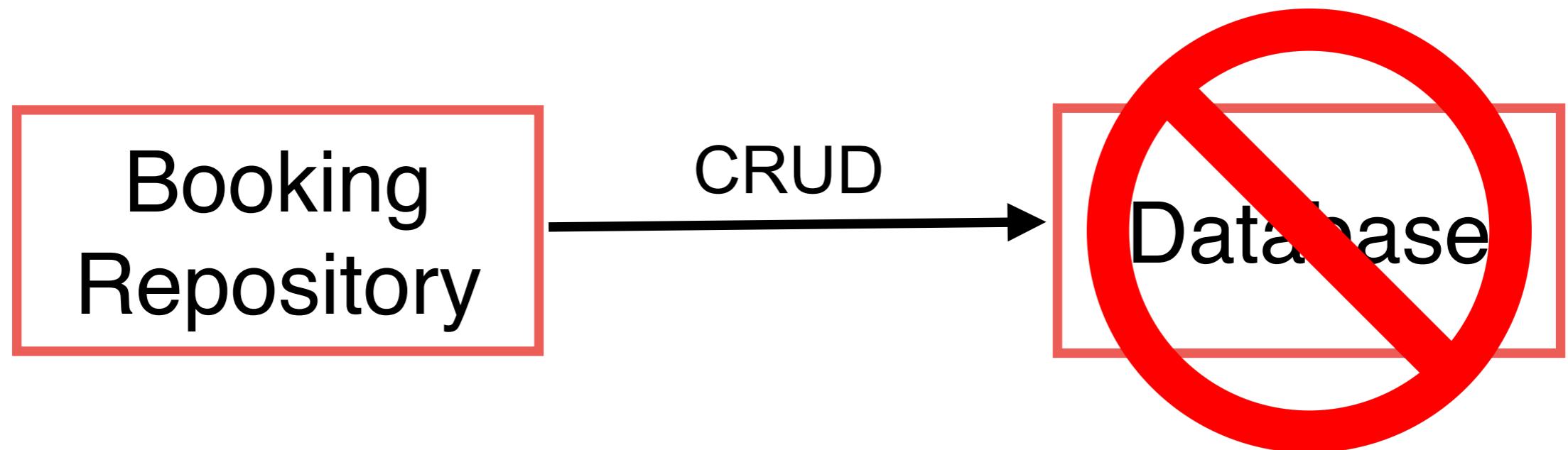
Booking
Repository



Validate input ?

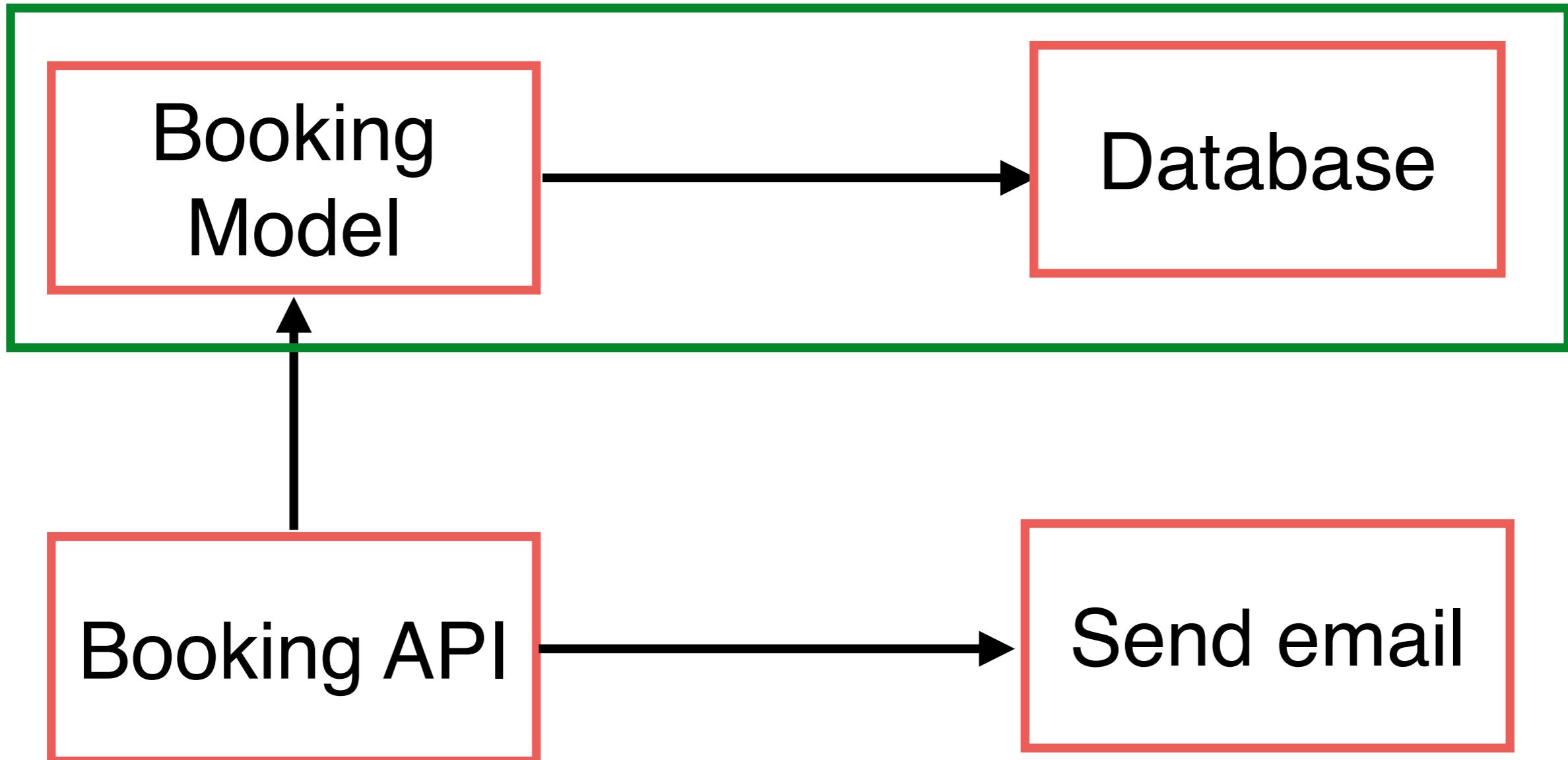


CRUD with Database



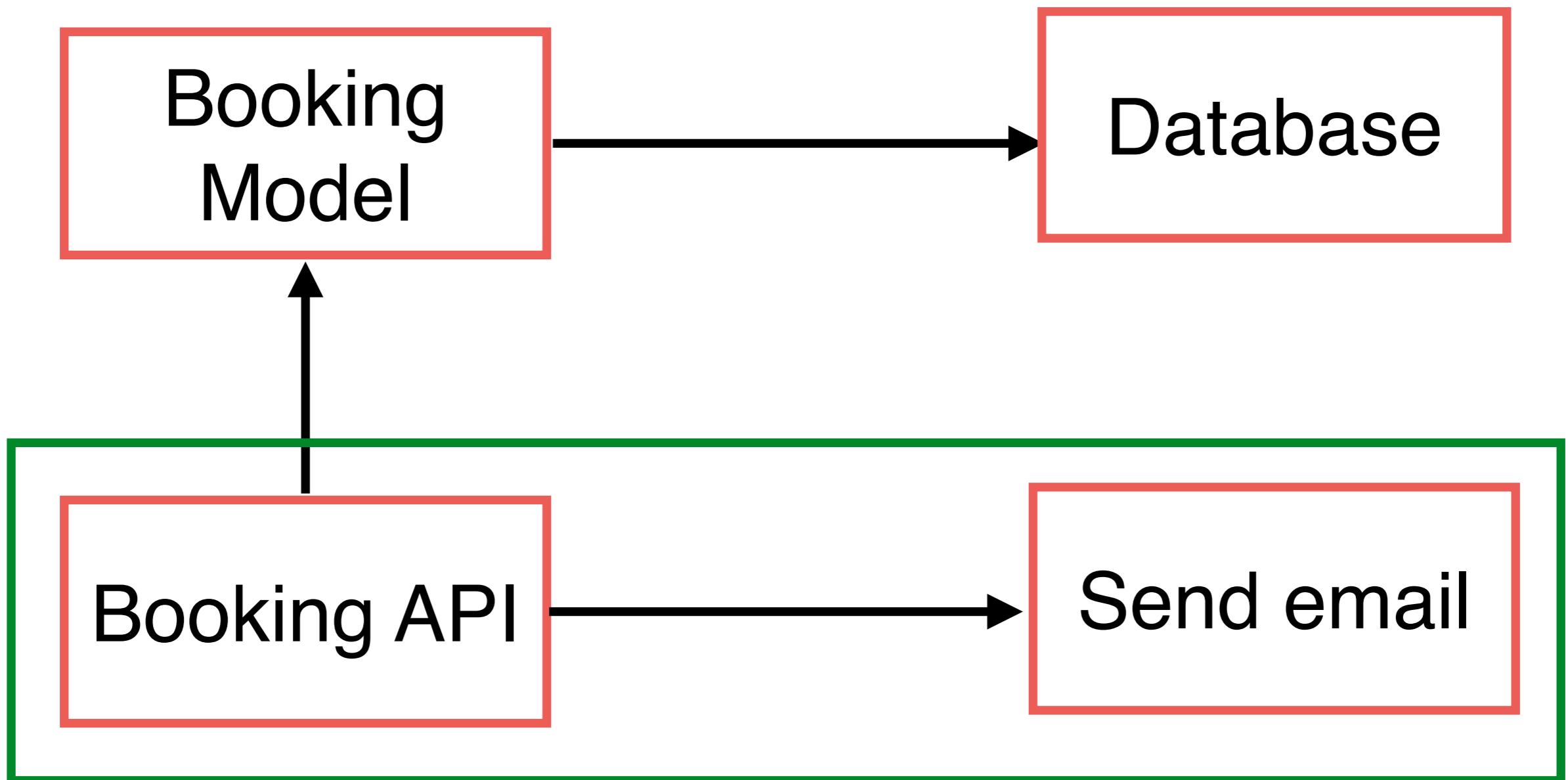
Testing relationship ::

Integration test



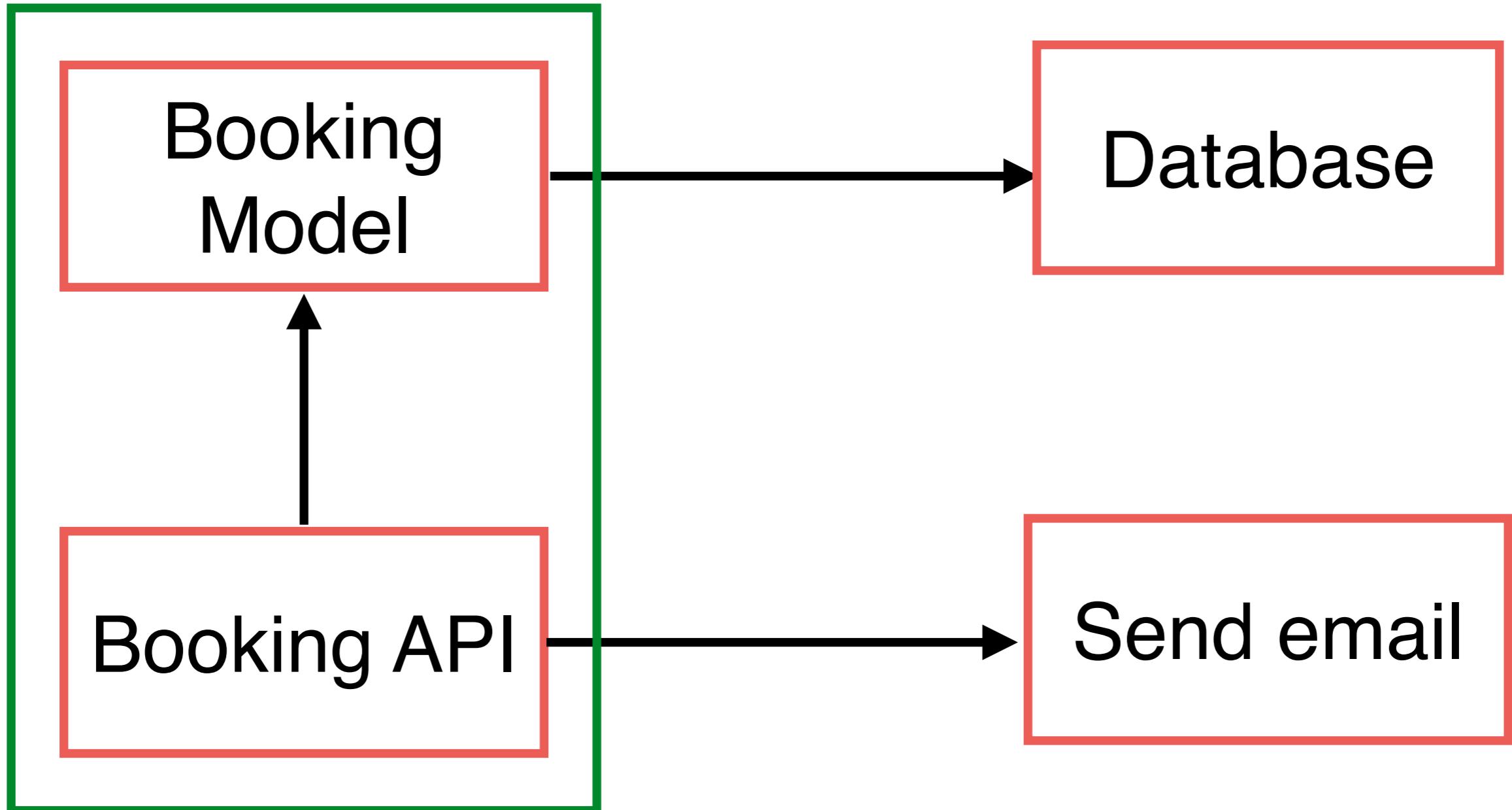
Testing relationship ::

Integration test



Testing relationship ::

Integration test



What to tests ?

