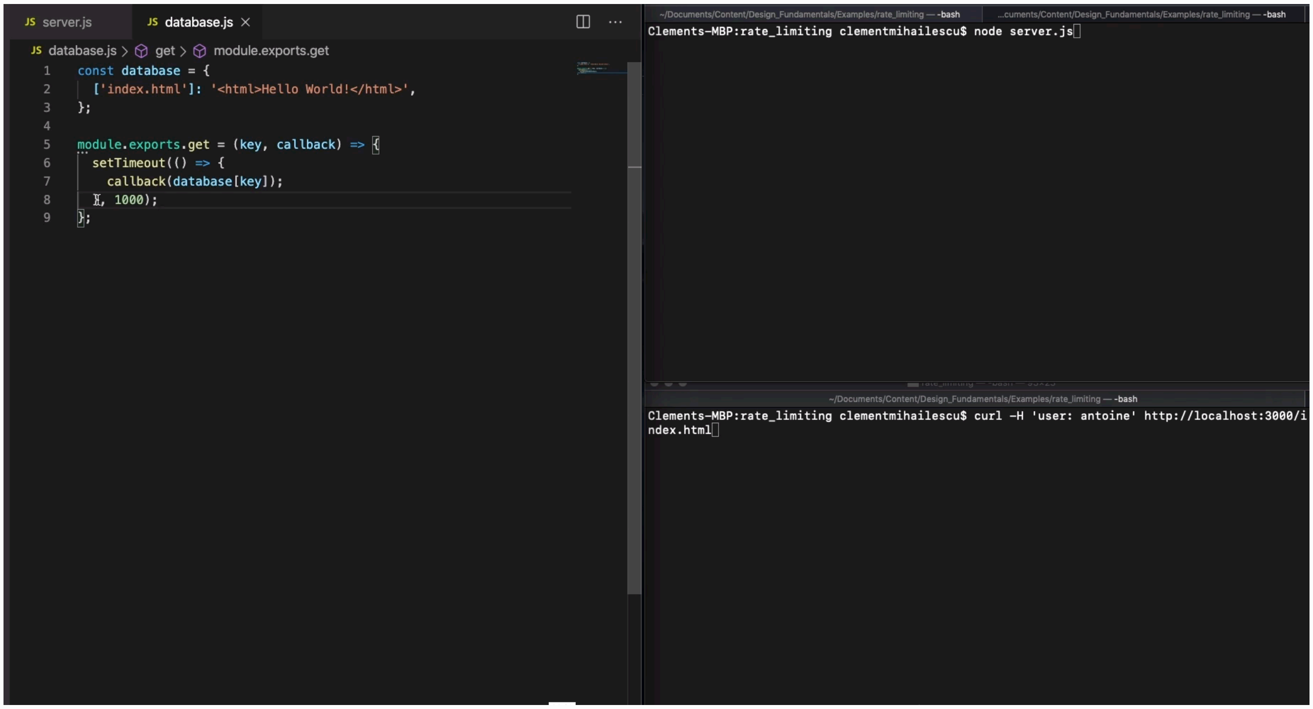
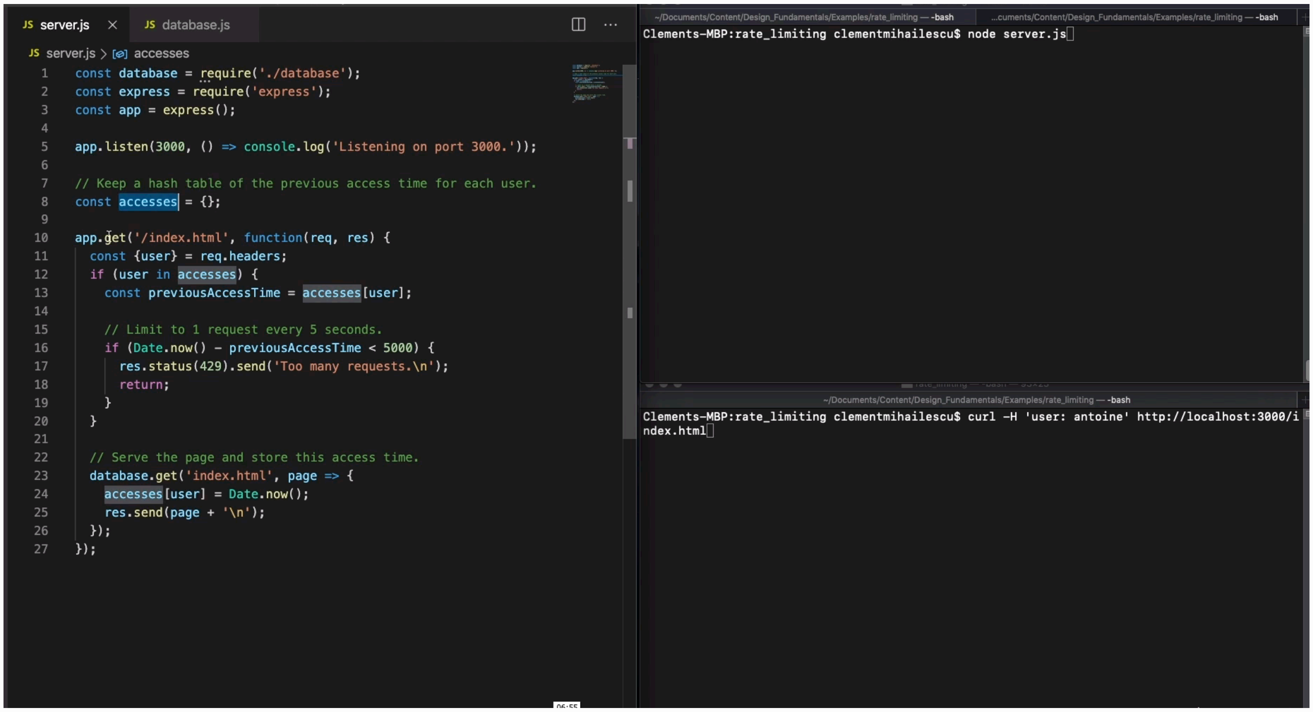
20_Rate_Limiting


server.js
database.js
database.js > get > module.exports.get
~/Documents/Content/Design_Fundamentals/Examples/rate_limiting — -bash
...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash
~/Documents/Content/Design_Fundamentals/Examples/rate_limiting — -bash

```javascript
const database = {
  ['index.html']: '<html>Hello World!</html>',
};

module.exports.get = (key, callback) => {
  setTimeout(() => {
    callback(database[key]);
  }, 1000);
};
```

```
Clements-MBP:rate_limiting clementmihailescu$ node server.js
```

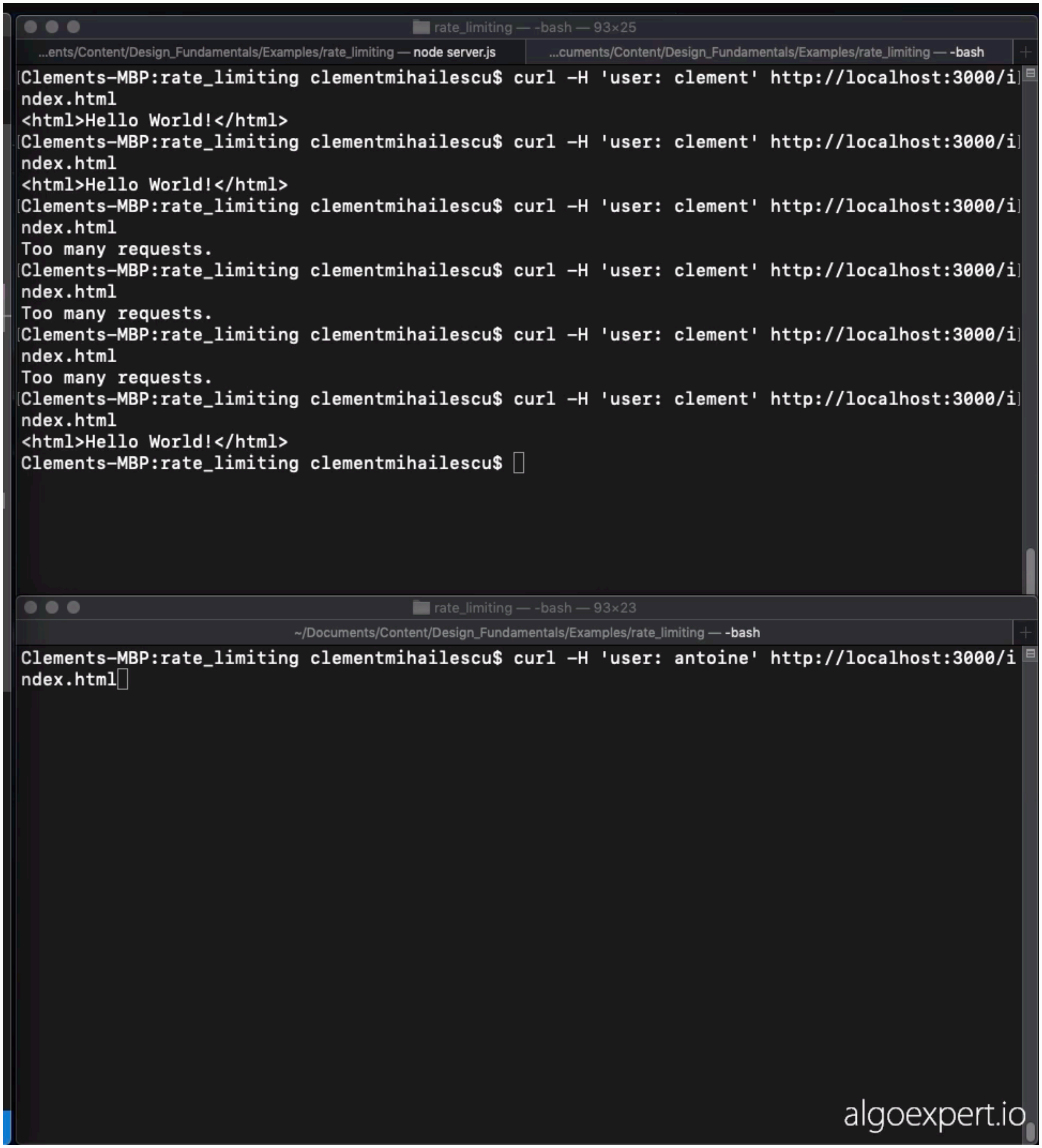
```
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: antoine' http://localhost:3000/i
ndex.html
```


server.js
database.js
server.js > accesses
~/Documents/Content/Design_Fundamentals/Examples/rate_limiting — -bash
...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash
~/Documents/Content/Design_Fundamentals/Examples/rate_limiting — -bash

```javascript
const database = require('./database');
const express = require('express');
const app = express();

app.listen(3000, () => console.log('Listening on port 3000.'));

// Keep a hash table of the previous access time for each user.
const accesses = {};

app.get('/index.html', function(req, res) {
  const {user} = req.headers;
  if (user in accesses) {
    const previousAccessTime = accesses[user];

    // Limit to 1 request every 5 seconds.
    if (Date.now() - previousAccessTime < 5000) {
      res.status(429).send('Too many requests.\n');
      return;
    }
  }

  // Serve the page and store this access time.
  database.get('index.html', page => {
    accesses[user] = Date.now();
    res.send(page + '\n');
  });
});
```

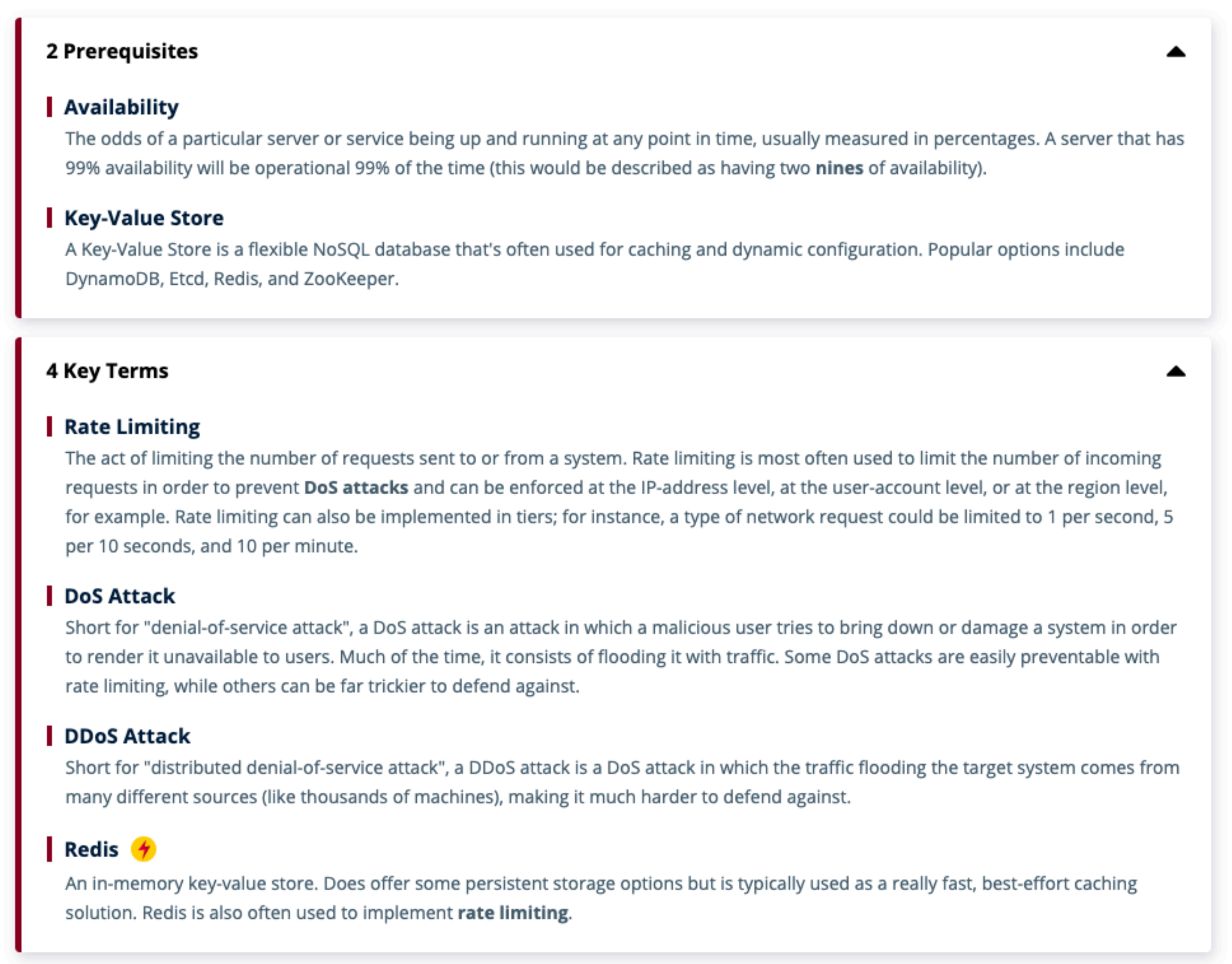
```
Clements-MBP:rate_limiting clementmihailescu$ node server.js
```

```
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: antoine' http://localhost:3000/i
ndex.html
```


rate_limiting — -bash — 93×25
...ents/Content/Design_Fundamentals/Examples/rate_limiting — node server.js
...cuments/Content/Design_Fundamentals/Examples/rate_limiting — -bash
rate_limiting — -bash — 93×23
~/Documents/Content/Design_Fundamentals/Examples/rate_limiting — -bash
algoexpert.io

```
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
<html>Hello World!</html>
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
<html>Hello World!</html>
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
Too many requests.
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
Too many requests.
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
Too many requests.
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: clement' http://localhost:3000/i
ndex.html
<html>Hello World!</html>
Clements-MBP:rate_limiting clementmihailescu$ 
```

```
Clements-MBP:rate_limiting clementmihailescu$ curl -H 'user: antoine' http://localhost:3000/i
ndex.html
```



2 Prerequisites ▲

Availability

The odds of a particular server or service being up and running at any point in time, usually measured in percentages. A server that has 99% availability will be operational 99% of the time (this would be described as having two **nines** of availability).

Key-Value Store

A Key-Value Store is a flexible NoSQL database that's often used for caching and dynamic configuration. Popular options include DynamoDB, Etcd, Redis, and ZooKeeper.

4 Key Terms ▲

Rate Limiting

The act of limiting the number of requests sent to or from a system. Rate limiting is most often used to limit the number of incoming requests in order to prevent **DoS attacks** and can be enforced at the IP-address level, at the user-account level, or at the region level, for example. Rate limiting can also be implemented in tiers; for instance, a type of network request could be limited to 1 per second, 5 per 10 seconds, and 10 per minute.

DoS Attack

Short for "denial-of-service attack", a DoS attack is an attack in which a malicious user tries to bring down or damage a system in order to render it unavailable to users. Much of the time, it consists of flooding it with traffic. Some DoS attacks are easily preventable with rate limiting, while others can be far trickier to defend against.

DDoS Attack

Short for "distributed denial-of-service attack", a DDoS attack is a DoS attack in which the traffic flooding the target system comes from many different sources (like thousands of machines), making it much harder to defend against.

Redis ⚡

An in-memory key-value store. Does offer some persistent storage options but is typically used as a really fast, best-effort caching solution. Redis is also often used to implement **rate limiting**.