

Mensageria e tarefas assíncronas com Node.js e BullMQ

Oiee, eu sou o Doce!

- ◆ 20 anos
- ◆ Dev frontend @ Voolt
- ◆ Fullstack (com foco em front)
- ◆ Pessoa não-binária
- ◆ Amo ouçar música









Por que implementar filas?









Produtores



BullMQ

Consumidores



Produtores



BullMQ

Consumidores



```
import express from "express";
const app = express();
const port = 3000;

app.get("/number", (req, res) => {
  res.send("10");
});

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

```
app.get("/number", (req, res) => {
  res.send("10");
});

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

```
app.get("/number", (req, res) => {  
  res.send("10");  
});
```

```
const randomNumber = (min: number, max: number) =>
  Math.floor(Math.random() * max + min);

app.get("/number", (req, res) => {
  res.send("10");
});
```



```
const randomNumber = (min: number, max: number) =>
  Math.floor(Math.random() * max + min);

app.get("/number", (req, res) => {
  const result = randomNumber(1, 10);
  res.send(`Your lucky number is ${result} ✨`);
});
```

```
const randomNumber = (min: number, max: number) =>
  Math.floor(Math.random() * max + min);

app.get("/number", (req, res) => {
  setTimeout(() => {
    const result = randomNumber(1, 10);
    res.send(`Your lucky number is ${result} ✨`);
  }, randomNumber(1000, 5000));
});
```

```
const randomNumber = (min: number, max: number) =>
  Math.floor(Math.random() * max + min);

const sleep = (ms: number) => new Promise((resolve) =>
  setTimeout(resolve, ms));

app.get("/number", (req, res) => {
  setTimeout(() => {
    const result = randomNumber(1, 10);
    res.send(`Your lucky number is ${result} ✨`);
  }, randomNumber(1000, 5000));
});
```

```
const randomNumber = (min: number, max: number) =>
  Math.floor(Math.random() * max + min);

const sleep = (ms: number) => new Promise((resolve) =>
  setTimeout(resolve, ms));

app.get("/number", async (req, res) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
  res.send(`Your lucky number is ${result} ✨`);
});
```

```
app.get("/number", async (req, res) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
  res.send(`Your lucky number is ${result} ✨`);
});
```

```
const queue = new Queue("example");

app.get("/number", async (req, res) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
  res.send(`Your lucky number is ${result} ✨`);
});
```

```
const queue = new Queue("example");

const worker = new Worker("example", async (job) => {

});

app.get("/number", async (req, res) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
  res.send(`Your lucky number is ${result} ✨`);
});
```

```
const queue = new Queue("example");

const worker = new Worker("example", async (job) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
});

app.get("/number", async (req, res) => {
  res.send(`Your lucky number is ${result} ✨`);
});
```



```
const queue = new Queue("example");

const worker = new Worker("example", async (job) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
});

app.get("/number", async (req, res) => {
  const job = await queue.add("lucky-number", null);
  res.send(`Your job ID is ${job.id}`);
});
```

```
const queue = new Queue("example");

const worker = new Worker("example", async (job) => {
  await sleep(randomNumber(1000, 5000));
  const result = randomNumber(1, 10);
});

const addLuckyNumberJob = () => queue.add("lucky-number", null);

app.get("/number", async (req, res) => {
  const job = await addLuckyNumberJob();
  res.send(`Your job ID is ${job.id}`);
});
```

```
app.get("/number", async (req, res) => {  
  const job = await addLuckyNumberJob();  
  res.send(`Your job ID is ${job.id}`);  
});
```

```
app.get("/number", async (req, res) => {  
  const job = await addLuckyNumberJob();  
  res.send(`Your job ID is ${job.id}`);  
});
```

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

```
app.get("/number", async (req, res) => {  
  const job = await addLuckyNumberJob();  
  res.send(`Your job ID is ${job.id}`);  
});
```

```
app.get("/jobs/:id", async (req, res) => {  
  const job = await Job.fromId(queue, req.params.id);  
  res.send(`Your lucky number is ${job?.returnValue} ✨`);  
});
```

```
app.get("/number", async (req, res) => {
  const job = await addLuckyNumberJob();
  res.send(`Your job ID is ${job.id}`);
});

app.get("/jobs/:id", async (req, res) => {
  const job = await Job.fromId(queue, req.params.id);
  if (!job?.returnvalue) {
    res.send("We're still calculating...");
    return;
  }
  res.send(`Your lucky number is ${job?.returnvalue} ✨`);
});
```

Como implementar esses endpoints no frontend?

Como implementar esses endpoints no frontend?

1. Adicionar trabalho na fila

Como implementar esses endpoints no frontend?

1. Adicionar trabalho na fila
2. Mostrar tela de carregando

Como implementar esses endpoints no frontend?

1. Adicionar trabalho na fila
2. Mostrar tela de carregando
3. Implementar long polling

Como implementar esses endpoints no frontend?

1. Adicionar trabalho na fila
2. Mostrar tela de carregando
3. Implementar long polling
4. Apresentar os dados

Como implementar esses endpoints no frontend?

1. Adicionar trabalho na fila
2. Mostrar tela de carregando
3. Implementar long polling
4. Apresentar os dados

✨ WebSockets & Event-driven Architecture ✨

Seria legal um painel pra monitorar as filas...

```
const serverAdapter = new ExpressAdapter();  
serverAdapter.setBasePath("/bullmq");
```

```
const serverAdapter = new ExpressAdapter();
serverAdapter.setBasePath("/bullmq");

createBullBoard({
  queues: [new BullMQAdapter(queue)],
  serverAdapter,
});
```

```
const serverAdapter = new ExpressAdapter();
serverAdapter.setBasePath("/bullmq");

createBullBoard({
  queues: [new BullMQAdapter(queue)],
  serverAdapter,
});


app.use("/bullmq", serverAdapter.getRouter());
```


QUEUES

example

● ACTIVE ● WAITING ● COMPLETED ● FAILED ● DELAYED



LATEST ACTIVE **1** WAITING COMPLETED **9** FAILED DELAYED PAUSED 

#10 lucky-number

Timeline: Added at 16:48:14 (3 milliseconds) | Process started at 16:48:14

[Data](#) [Options](#) [Logs](#)

```
{  
  "data": null,  
  "returnValue": null  
}
```

0%

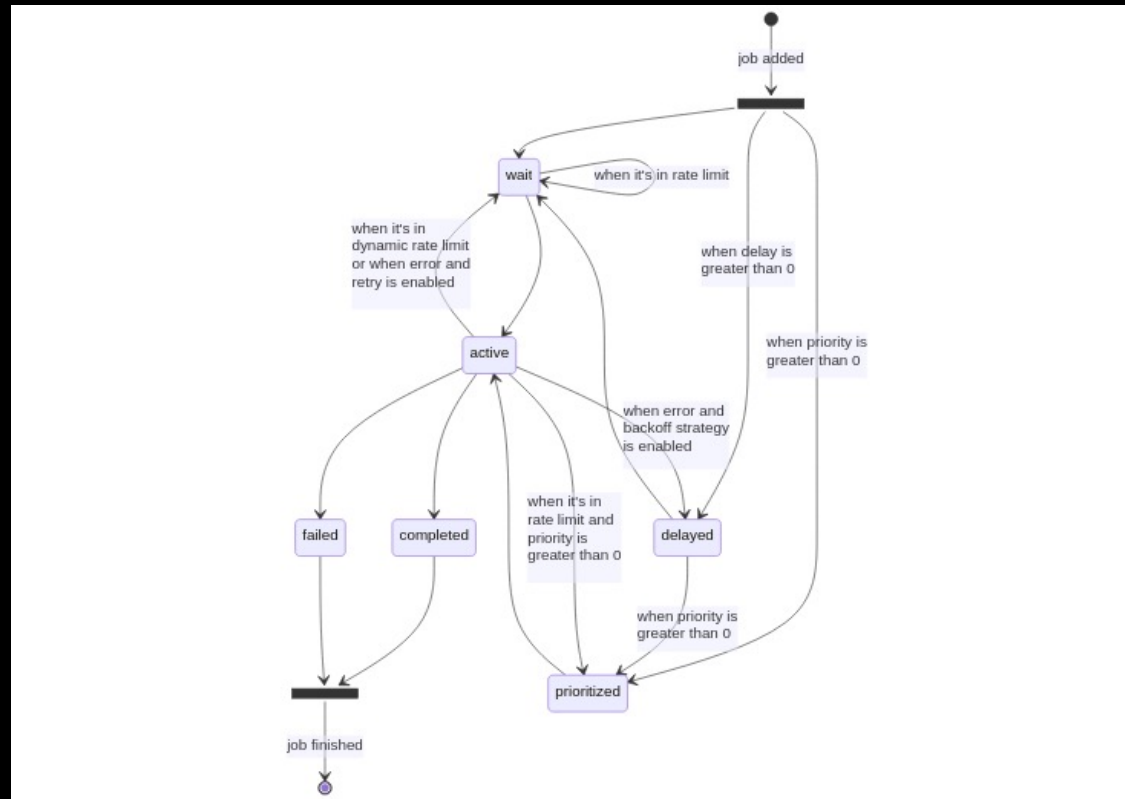
#6 lucky-number attempt #2

Timeline: Added at 16:14:33 (1 minute) | Process started at 16:15:53

[Data](#) [Options](#) [Logs](#)

```
{  
  "data": null,  
  "returnValue": 4  
}
```

Ciclo de vida de um job



Executando jobs em paralelo

```
const worker = new Worker("example", (job) => {  
  await sleep(randomNumber(1000, 5000));  
  return randomNumber(1, 10);  
});
```

```
const worker = new Worker("example", (job) => {
  await sleep(randomNumber(1000, 5000));
  return randomNumber(1, 10);
},
{
  concurrency: 50,
}
);
```

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Load balancer

(ELB, Nginx...)

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Produtor

Adiciona job na fila

Consumidor

Processa os jobs da fila

Alternativas ao BullMQ

Alternativas ao BullMQ



BullMQ é maneiro se você precisa...

BullMQ é maneiro se você precisa...

✦ Processar tarefas demoradas

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo

BullMQ é maneiro se você precisa...

- ✦ Processar tarefas demoradas
- ✦ Processar muitas tarefas ao mesmo tempo
- ✦ Processar tarefas de forma assíncrona

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo
- ◆ Processar tarefas de forma assíncrona
- ◆ Escalar suas aplicações

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo
- ◆ Processar tarefas de forma assíncrona
 - ◆ Escalar suas aplicações
 - ◆ Agendar tarefas

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo
- ◆ Processar tarefas de forma assíncrona
 - ◆ Escalar suas aplicações
 - ◆ Agendar tarefas
- ◆ Monitorar e registrar sua aplicação

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo
- ◆ Processar tarefas de forma assíncrona
 - ◆ Escalar suas aplicações
 - ◆ Agendar tarefas
- ◆ Monitorar e registrar sua aplicação
- ◆ Lidar com erros e manter uptime

BullMQ é maneiro se você precisa...

- ◆ Processar tarefas demoradas
- ◆ Processar muitas tarefas ao mesmo tempo
- ◆ Processar tarefas de forma assíncrona
 - ◆ Escalar suas aplicações
 - ◆ Agendar tarefas
 - ◆ Monitorar e registrar sua aplicação
 - ◆ Lidar com erros e manter uptime
- ◆ Comunicar entre aplicações em tempo real

Valeu! 🥰

✦ doceazedo.com

✦ github.com/doceazedo

✦ twitter.com/doceazedo911

✦ twitch.tv/doceazedo911

Alguma pergunta?