server.js routes.http

server.js > ...

```
import { fastify } from "fastify";
import { DatabaseMemory } from "./database-memory.js";
import { title } from "process";


const server = fastify();

const database = new DatabaseMemory();

server.post("/videos", (request, reply) => {
    const { title, description, duration } = request.body;
    database.create({
        title,
        description,
        duration: 200
    });

    console.log(body);
    return reply.status(201).send();
})

server.get("/videos", () => {
    return "Gestão de Videos";
})
server.put("/videos/:id", () => {
    return "Atualizar video";
})
server.delete("/videos/:id", () => {
    return "Deletar video";
})

server.listen({port:3000}, (err, address) => {
    if (err) {
        console.error(err);
        process.exit(1);
    }
    console.log(`Servidor rodando em ${address}`);
});
```

server.js | database-memory.js | routes.http

routes.http > POST /videos

Send Request

```
POST http://localhost:3000/videos
Content-type: application/json

{
    "title": "Video 01",
    "description": "Este é o primeiro video",
    "duration": 180
}
```

server.js | database-memory.js | routes.http

database-memory.js > DatabaseMemory

```
import { randomUUID } from "crypto";

Windsurf: Refactor | Explain
export class DatabaseMemory {
    #videos = new Map();

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    list() {
        return this.#videos.values();
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    create(video) {
        const videoId = randomUUID();
        this.#videos.set(videoId, video);
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    update(id, video) {
        this.#videos.set(id, video);
    }

    Windsurf: Refactor | Explain | Generate JSDoc | ×
    delete(id) {
        this.#videos.delete(id);
    }
}
```