

```

1: mod job;
2: mod printer;
3:
4: use actix_web::{web, App, HttpRequest, HttpServer, Responder,
HttpResponse};
5: use job::Job;
6: use printer::Printer;
7: use chrono::Utc;
8: use std::sync::Mutex;
9:
10: struct AppState {
11:     printer: Mutex<Printer>,
12: }
13:
14: #[derive(serde::Deserialize)]
15: struct PrintRequest {
16:     priority: u32,
17:     team_name: String,
18:     file_content: String,
19:     color: bool,
20: }
21:
22: async fn print_job(
23:     data: web::Data<AppState>,
24:     req: web::Json<PrintRequest>,
25: ) -> impl Responder {
26:     let job = Job::new(
27:         req.priority,
28:         req.team_name.clone(),
29:         Utc::now(),
30:         req.file_content.clone(),
31:         req.color,
32:     );
33:
34:     let printer = data.printer.lock().unwrap();
35:     match printer.submit_task(job) {
36:         Ok(_) => HttpResponse::Ok().body("🎉 🎉 🎉 🎉 🎉 🎉 🎉 🎉"),
37:         Err(_) => HttpResponse::Conflict().body("🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥 🔥"),
38:     }
39: }
40:
41: async fn greet(req: HttpRequest) -> impl Responder {
42:     let name = req.match_info().get("name").unwrap_or("World");
43:     format!("Hello {}!", &name)
44: }
45:
46: #[actix_web::main]
47: async fn main() -> std::io::Result<()> {
48:     let printer = Printer::new();

```

```
49: let app_state = web::Data::new(AppState {
50:     printer: Mutex::new(printer),
51: });
52:
53: HttpServer::new(move || {
54:     App::new()
55:         .app_data(app_state.clone())
56:         .route("/", web::get().to(greet))
57:         .route("/{name}", web::get().to(greet))
58:         .route("/print", web::post().to(print_job))
59: })
60: .bind("127.0.0.1:8080")?
61: .run()
62: .await
63: }
```