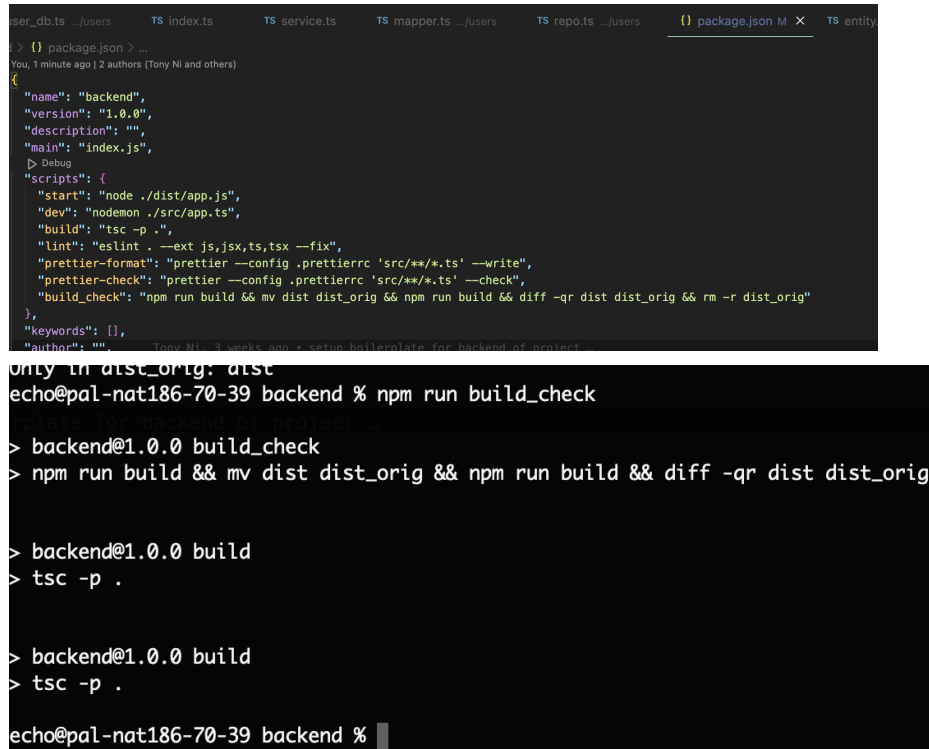


58 lines (46 sloc) | 1.19 KB

```
1  on: [push]
2  jobs:
3    lint:
4      runs-on: ubuntu-latest
5      name: Run ESLint
6      steps:
7        - name: Checkout code
8          uses: actions/checkout@v2
9
10       - name: Setup Node
11         uses: actions/setup-node@v1
12         with:
13           node-version: 14
14
15       - name: Install NPM packages
16         run: cd backend/ && npm install
17
18       - name: Lint
19         run: cd backend/ && npm run lint
20
21       - name: Check formatting
22         run: cd backend/ && npm run prettier-check
23
24    build:
25      runs-on: ubuntu-latest
26      name: Build Project
27      steps:
28        - name: Checkout code
29          uses: actions/checkout@v2
30
31        - name: Setup Node
32          uses: actions/setup-node@v1
33          with:
34            node-version: 14
35
36        - name: Install NPM packages
37          run: cd backend/ && npm install
38
39        - name: Build
40          run: cd backend/ && npm run build
41
42    test:
43      runs-on: ubuntu-latest
44      name: Test Project
45      steps:
46        - name: Checkout code
47          uses: actions/checkout@v2
48
49        - name: Setup Node
50          uses: actions/setup-node@v1
51          with:
52            node-version: 14
53
54        - name: Install NPM packages
55          run: cd backend/ && npm install
56
57        - name: Test
58          run: cd backend/ && npm run test
```

Our CI uses github actions to check our codebase. We have 3 jobs right now: lint, build, and test. The linter uses eslint to lint our code and then uses checks if it is following our prettier.rc's format. For the build we just build our project using tsc which should compile our typescript code into javascript. For the test use are testing using jest and we will have a test directory that will implement both unit tests and integrations test.



The image shows a VS Code editor with a file explorer at the top displaying several TypeScript files: `user_db.ts`, `index.ts`, `service.ts`, `mapper.ts`, `repo.ts`, `package.json` (selected), and `entity.ts`. The `package.json` file is open in the editor, showing the following content:

```
{
  "name": "backend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node ./dist/app.js",
    "dev": "nodemon ./src/app.ts",
    "build": "tsc -p .",
    "lint": "eslint . --ext js,jsx,ts,tsx --fix",
    "prettier-format": "prettier --config .prettierrc 'src/**/*.ts' --write",
    "prettier-check": "prettier --config .prettierrc 'src/**/*.ts' --check",
    "build_check": "npm run build && mv dist dist_orig && npm run build && diff -qr dist dist_orig && rm -r dist_orig"
  },
  "keywords": [],
  "author": ""
}
```

Below the editor, a terminal window shows the execution of the `build_check` script. The prompt is `echo@pal-nat186-70-39 backend % npm run build_check`. The output shows the following sequence of commands and their results:

```
> backend@1.0.0 build_check
> npm run build && mv dist dist_orig && npm run build && diff -qr dist dist_orig

> backend@1.0.0 build
> tsc -p .

> backend@1.0.0 build
> tsc -p .

echo@pal-nat186-70-39 backend %
```

As of right now our build is reproducible as two different build instances yielded the same javascript build directory. The only way we could improve on our development process to maintain this is by using clean-installs for all of our node packages to ensure that their installation is not dependent on any files previously held on our development systems.