**Setup inicial do projeto**

* Criar um diretório
* npm init -y //criar um package .json inicial com valores default
* usando o vccode como editor instale o package vscode-icons

**Configurando o Typescript no projeto**

* Abre o terminal no vscode
* **npm i -D -E** [**typescript@2.5.3**](mailto:typescript@2.5.3) / instala o typescritp
* crie na raiz do projeto um arquivo tsconfig.json com o conteúdo:

{

"compilerOptions": {

"target": "es6", // padrão de modulo ecmascript 6

"module": "commonjs", // gerar módulos no padrão commonjs

"outDir": "dist" // diretório para onde o código js será jogado

},

"compileOnSave": false,

"include": [

"src/\*\*/\*.ts" // os arquivos que seram compilados

],

"exclude": [

"node\_modules" // diretório que será excluído da compilação

]

}

* node\_modules/.bin/tsc // comando para compilar o arquivo typescrip, gera o diretório dist e cria o arquivo js

**Automatizar as tarefas com Glup**

* npm i -D -E [gulp@3.9.1](mailto:gulp@3.9.1) [gulp-typescript@3.2.2](mailto:gulp-typescript@3.2.2) [gulp-clean@0.3.2](mailto:gulp-clean@0.3.2) ( -D é para indiciar que é um componente de desenvolvimento e o -E para baixar a versão exata na descrição )
* Criar um arquivo na raiz gulpfiles.js com a implementação abaixo: (tarefas)

const gulp = require('gulp');

const clean = require('gulp-clean');

const ts = require('gulp-typescript');

const tsProject = ts.createProject('tsconfig.json'); //cria um projeto baseado no tsconfig.json

//criação de tarefas para gulp

Tarefa script

gulp.task('script' , () => {

//call-back executado na logica

const tsResult = tsProject.src() // pega o código typescripe

.pipe(tsProject()); // faz uma operação com o código fonte

return tsResult.js

.pipe(gulp.dest('dist')); // faz uma nova operação, joga o resultado no diretório dist

})

Tarefa static

//tarefa para copiar os arquivos estáticos para o dist, especificamente //copia os json para o arquivo dist

gulp.task('static ', () => {

return gulp

.src(['src/\*\*/\*.json']) // pega os arquivos com a extensão json

.pipe(gulp.dest('dist')); // copia para o diretorio

});

Tarefa clean

//limpa o diretório dist

gulp.task('clean', () => {

return gulp

.src('dist')

.pipe(clean());

});

Tarefa build chama as outras tarefas em ordem

gulp.task('build', ['scripts' ]);

Tarefa para ficar ouvindo se há alterações

gulp.task('watch', ['build'], () => {

// diretórios que ficará ouvindo, sempre que houver alterações o build será executado

return gulp.watch(['src/\*\*/\*.ts', 'src/\*\*/\*.json'], ['build']);

});

Tarefa default para chamar as tarefas

gulp.task('default', ['watch']);

Criando a tarefa default apenas com o comando node\_modules/.bin/gulp já executa a tarefa

* Criar um script no package.json para rodar o comando gulp, no arquivo package.son add o scripts gulp

{

"name": "GraphQL-node-api",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"gulp:": "node\_modules/.bin/gulp"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"gulp": "3.9.1",

"gulp-clean": "0.3.2",

"gulp-typescript": "3.2.2",

"typescript": "2.5.3"

}

}

Rode um npm run gulp // a aplicação ficará ouvindo alterações

* Instalando a dependência nodemon para que o compilado fique ativo, sem na necessidade de compilar e executar sempre (comando para compilar “ node dist/app “ dist/app caminho para o arquivo)

**npm i -D -E** [**nodemon@1.12.0**](mailto:nodemon@1.12.0)

Após a instalação é necessário criar um script no npm assim como foi criado para o gulp ( --delay 5 comando para fazer a verificação a cada 5 s)

{

"name": "GraphQL-node-api",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"dev": "node\_modules/.bin/nodemon --delay 5 dist/app",

"gulp": "node\_modules/.bin/gulp"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"gulp": "3.9.1",

"gulp-clean": "0.3.2",

"gulp-typescript": "3.2.2",

"typescript": "2.5.3"

}

}

* Instalando a configurando o express ( utilizado nas api rest para criar endpoints, uma forma de receber requisições no servidor )

**npm i -S -E** [**express@4.16.2**](mailto:express@4.16.2)

* Por estar trabalhando com o typescrip é necessário baixar um pacote de definições de tipos do express

**npm i -D -E @types/express@4.0.37 @types/node@8.0.34**

* **Instalando as dependências do graphql:**

**npm i -S -E express-graphql@0.6.11 graphql@0.11.7** [**graphql-tools@2.3.0**](mailto:graphql-tools@2.3.0)

[express-graphql@0.6.11](mailto:express-graphql@0.6.11): é um midleware necessário para fazer o graphql trabalhar com o express

[graphql@0.11.7](mailto:graphql@0.11.7): implementação javascript do graphql

[graphql-tools@2.3.0](mailto:graphql-tools@2.3.0): utilitário para construir o esquema graphql da api

Definir os tipos para as dependências instaladas:

**npm i -D -E @types/express-graphql@0.0.34 @types/graphql@0.11.5**

* Instalação de pacotes para o sequelize

**npm i -S -E sequelize@4.13.10** [**mysql2@1.4.2**](mailto:mysql2@1.4.2)

pacotes typescript

npm i -D -E @types/sequelize@4.0.76