# Mantendo seu projeto atualizado com Dependabot e Github Actions

## Atualizar dependências é algo bem chato e demorado, mas não precisa ser.

## Introdução

Fala pessoal, mantendo a ideia de ter vídeo, mas também ter post escrito, hoje eu vou falar sobre um vídeo que fiz recentemente que é o [Mantendo projeto atualizado com Dependabot e Github Actions](https://youtu.be/zV9yZZzZGpU).

A ideia desse vídeo/post veio de um pedido de um aluno lá [no Slack dos cursos](https://willianjusten-cursos.slack.com/archives/C016MQE9CAD/p1596728688141600). Ele estava perguntando como fazer para manter o [boilerplate criado no curso React Avançado](https://github.com/React-Avancado/boilerplate) todo atualizado.

Mas vamos lá, enquanto eu escrevo esse post, eu vou ouvindo uma playlist chamada [Alone Again](https://open.spotify.com/playlist/37i9dQZF1DWX83CujKHHOn?si=GqJt0lTZSGywtV0G6VCt6A), que tem umas músicas calmas, eu confesso que estou de péssimo humor hoje, então talvez isso me acalme.

## Versão em vídeo

Para quem quiser assistir a versão em vídeo desse post, segue aí:

https://www.youtube.com/watch?v=zV9yZZzZGpU&feature=emb\_logo

## Sobre o Dependabot

O primeiro carinha que vamos mexer é o [Dependabot](https://dependabot.com/). Para quem não conhece, esse bot nasceu fora do Github como uma forma de indicar pacotes que poderiam apresentar falhas de segurança e o mesmo enviava um Pull Request com as devidas correções para que pudessem ser revisadas.

O bot começou a ser tão utilizado e funcionar tão bem, que a Microsoft/Github decidiu comprar para que o bot passasse a ser totalmente free e mais fácilmente integrado ao sistema, o que foi um grande ganho para nós desenvolvedores =)

## Configurando o Dependabot

Nós vamos utilizar o [Dependabot](https://dependabot.com/) então para que ele não somente busque os patches de segurança, mas também busque por atualizações das nossas dependências. E esse processo é bem simples e pode ser feito de duas maneiras:

### Utilizando via UI

Para usar via UI, você precisa se [cadastrar](https://app.dependabot.com/auth/sign-up) no site e logo após se cadastrar, você vai ver uma tela similar a essa:

Basta então clicar para adicionar os repositórios e você irá ver uma lista com todos os repositórios que você tiver, similar a imagem:

Basta selecionar os repositórios que desejar. Aconselho a escolher somente os que você realmente irá manter, caso contrário, você irá receber milhares de mensagens e pode acabar deixando de lado, o que não é o objetivo, não é mesmo?

Logo depois de adicionado, o seu projeto estará numa lista e você pode clicar na engrenagem para definir as informações desejadas, como na imagem abaixo:

Você ali pode definir quando atualizar (diariamente, semanalmente, mensalmente e até live). Eu recomendo o diariamente, que já é mais do que suficiente.

### Via dependabot.yml

Como o Github agora já funciona com o Dependabot nativamente, você pode criar um arquivo de configuração e salvá-lo dentro da pasta .github dentro do seu repositório.

Você pode ver [a documentação](https://docs.github.com/en/github/administering-a-repository/configuration-options-for-dependency-updates) sobre como criar o arquivo e abaixo segue o exemplo para a mesma configuração que eu fiz na UI:

version: 2

updates:

- package-ecosystem: npm

directory: "/"

schedule:

interval: daily

open-pull-requests-limit: 10

## Sobre o Github Actions

Vimos agora que o Dependabot irá enviar Pull Requests sempre com as atualizações, mas isso não garante que a atualização vai funcionar com o nosso código. Para isso, nós vamos utilizar o [Github Actions](https://docs.github.com/en/actions), que nos permite criar um fluxo de ações que devem ocorrer a partir de determinada ação.

No nosso caso, nós queremos rodar nosso lint, tests e build para garantir que tudo continua funcionando, mesmo com a nova atualização.

### Configurando o Github Actions

Para configurar, o processo é bem similar ao dependabot.yml, nós também teremos um arquivo yml com algumas configurações bem definidas.

A [documentação do Github Actions](https://docs.github.com/en/actions/reference/workflow-syntax-for-github-actions) é super completa e bem detalhada, então recomendo vocês darem uma olhadinha com carinho.

No caso do boilerplate, eu criei um arquivo em [.github/worflows/ci.yml](https://github.com/React-Avancado/boilerplate/blob/master/.github/workflows/ci.yml) com a seguinte configuração:

name: ci

on: [pull\_request]

jobs:

build:

runs-on: ubuntu-latest

steps:

- name: Checkout Repository

uses: actions/checkout@v2

- name: Setup Node

uses: actions/setup-node@v1

with:

node-version: 14.x

- name: Install dependencies

run: yarn install

- name: Linting

run: yarn lint

- name: Test

run: yarn test

- name: Build

run: yarn build

Nele eu defino que vai rodar sempre que um pull request for aberto, a máquina usada será uma simples do Ubuntu e a versão do Node será 14.x.

Os comandos abaixo são específicos para o meu boilerplate e eles vem do meu [package.json](https://github.com/React-Avancado/boilerplate/blob/master/package.json), então você pode escolher os comandos e ordem que você desejar para o seu projeto.

## Conclusão

Após essas duas configurações, sempre que alguma atualização aparecer para aquela sua dependência, você irá receber um PR com a atualização e automaticamente o Github também vai rodar o conjunto de comandos para você, caso algo esteja errado, ele vai te indicar, assim, você vai ter segurança em atualizar ou não alguma coisa.

Espero que você tenha curtido essa dica, e se você puder se [inscrever lá no canal do YouTube](https://www.youtube.com/WillianJustenCursos/?sub_confirmation=1), ajuda demais também!

https://willianjusten.com.br/mantendo-seu-projeto-atualizado-com-dependabot-e-github-actions/