**GIT FLOW:**

Para começar a fazer as alterações do fatecarona é preciso preparar o fluxo de desenvolvimento, o fatecarona está no git na branch “redux” foi como o grupo anterior deixou... para melhorar a organização do GIT eu vou usar a metodologia do GIT FLOW, onde tem 3 branchs principais:

**master**: é a branch principal onde fica a versão de produção

**develop**: é onde vai ser criado as features e conter as atualizações, no fim ele vai dar merge na máster.

**Hotfix**: é a branch para corrigir bug de produção, ela altera tanto a develop como a máster, como o fatecarona não está em produção ela não será usada (mas é bom saber pra que ela serve).

O GIT FLOW é usado apartir de FEATURES, que são pequenas mudanças isoladas, sabe os itens do design thinking? Cada item daqueles vai ser uma feature que vai nascer a partir da develop, ser realizada na branch “feature/nome\_da\_branch” e por fim ser mesclada na develop novamente, por exemplo o card do design thinking de “arruma senha”... é criado uma nova branch com nome “feature/arrumar\_senha”, todo o código é desenvolvido nessa branch isolada da develop e após ser concluída ela volta para a develop. Isso permite desenvolvimento paralelo de features já que ficam em ambientes isolados (também ajuda em **code coverage** já que ajuda a fazer testes unitários em cada feature isoladamente, mas não abordarei isso nesse momento).

**Padrões:**

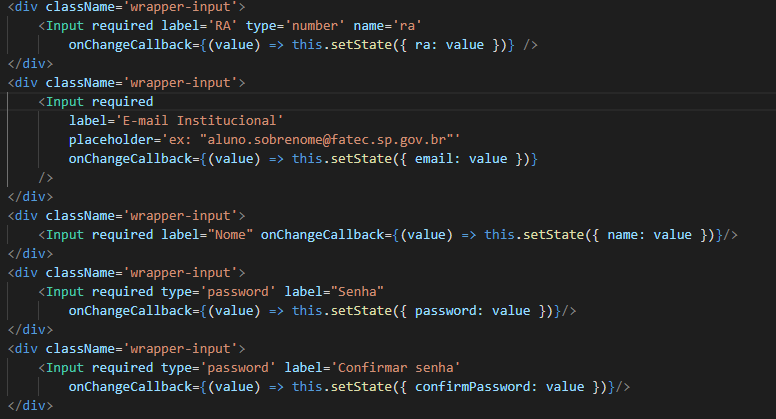
O código do WebApp não tem padrão, metade está em português e a outra metade em inglês, como tem mais trechos em português o código vai ser traduzido para inglês

**DRY (don’t repeat yourself):**

O DRY diz para retirarmos repetições no código usando abstrações, na feature “arrumar senha” por exemplo tínhamos esse código:

// print dos input repetido

Cheio de repetições de código e parâmetros, então decidi abstrair um input genérico que já vem com funcionalidades básicas necessárias no react com parâmetros default que é utilizado em todo lugar no webapp e ficou assim:



Caso tenha que mudar um layout de input do webapp, mudando apenas o componente “input” que criei altera em todos os trechos melhorando MUITO a manutenibilidade.

Dry não deve ser confundido com reduzir redundância no código, isso se faz com boas práticas e alguns code convetions.

**WET (write everything twice):**

O WET significa “escreva tudo duas vezes”... completamente diferente do DRY?

Não! O WET segue um principio que uma repetição de código pode fazer duas coisas diferentes parecerem iguais mas na verdade não são!

Tipo uma regra de negócio poder mudar no futuro, então com isso o WET fala que tudo bem ter uma repetição que isso não tem grande impacto na manutenibilidade e pode ajudar futuramente.

**WET VS DRY:**

E agora? A regra é clara, onde houver 2 repetições não iremos abstrair, 3 ou mais vamos abstrair, o melhor dos 2 mundos.