Olá colegas, bem hoje estava passando o tempo aqui esperando outra pessoa finalizar uma tratativa de erro para ajustar um component de mensagem no Angular, mas ai começou o ponto ajustar mock para simular error? aguardar o time finalizar?

Nenhuma opção era tão pratica.

Então pensei vou fazer um interceptor que vai gerar o error na chamada que eu preciso, simples... Pois bem funcionou e é isso que trago hoje para vocês.

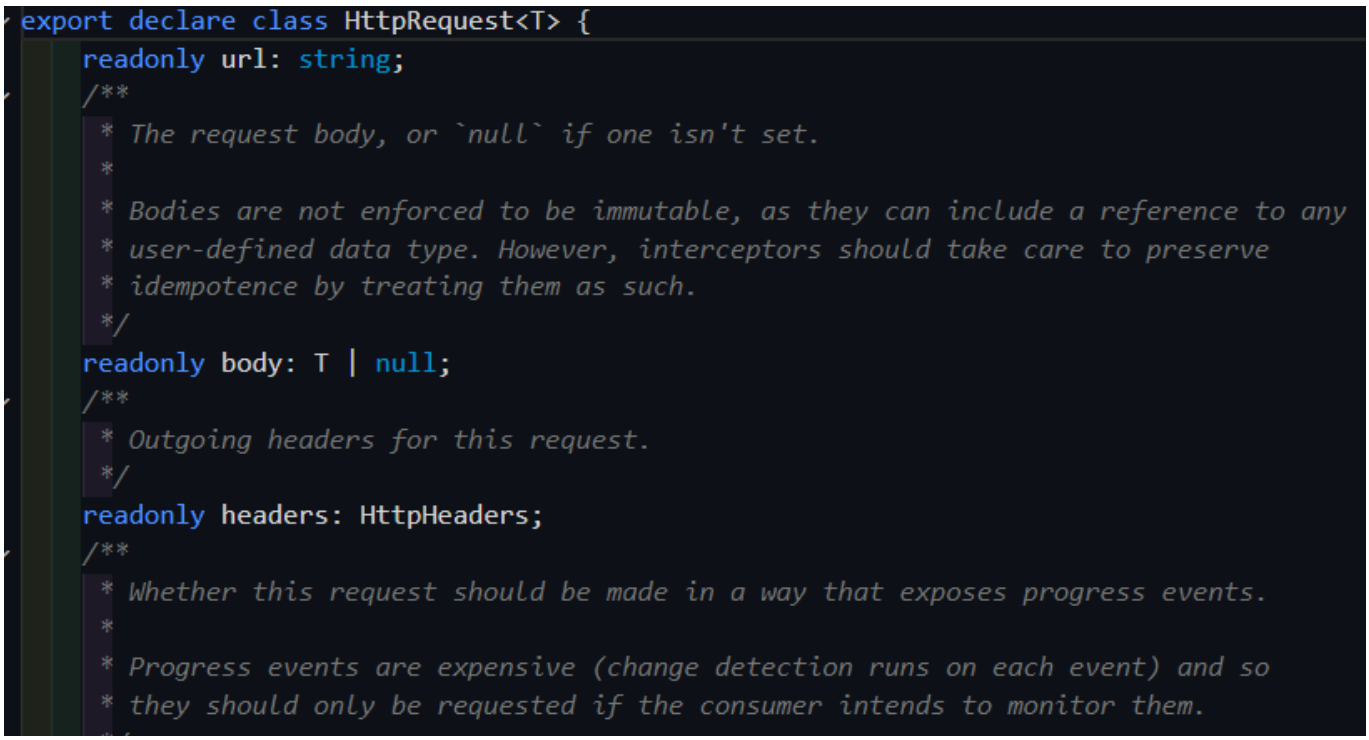
Direto ao código, você precisa criar um novo interceptor, indiferente da versão do Angular vai funcionar porem atente para a forma como o interceptor deve ser criado versão legada é diferente das atuais beleza (estou considerando que já sabe isso ok).
Estruture da seguinte forma:

```
intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
{
    const simulatedError = new HttpErrorResponse({
      error: { 'code': 112, 'actualMessage': 'Ola mundo erro simples',
'userMessage': 'Infelizmente não deu certo, sou um erro detalhado aqui!' },
      status: 500,
      statusText: 'Internal Server Error'
    });
    // Adicione a url que gostaria de validar o erro
    if (!req.) {
      return next.handle(req);
    } else {
      // Simula o erro
      return next.handle(req).pipe(() => throwError(simulatedError));
    }
  }
```

Identifique a constante “simulatedError” ela deve sempre ter esse formato, no qual são informados o status code, status text e a mensagem de erro que será devolvida pelo seu backend.

Observe o “req.url.includes” aqui é preciso colocar a rota como ela é passada então se tiver parâmetros adicione, porem e claro é possível ajustar para ficar mais genérica, mas “eu particularmente não gosto” devido que você vai simular um erro então quanto mais especifico melhor, outro ponto aqui estou usando por padrão um

GET métodos como PUT, POST podem ser melhor tratados também.



```
export declare class HttpRequest<T> {
    readonly url: string;
    /**
     * The request body, or `null` if one isn't set.
     *
     * Bodies are not enforced to be immutable, as they can include a reference to any
     * user-defined data type. However, interceptors should take care to preserve
     * idempotence by treating them as such.
     */
    readonly body: T | null;
    /**
     * Outgoing headers for this request.
     */
    readonly headers: HttpHeaders;
    /**
     * Whether this request should be made in a way that exposes progress events.
     *
     * Progress events are expensive (change detection runs on each event) and so
     * they should only be requested if the consumer intends to monitor them.
```

Conforme imagem o HttpRequest te possibilita muita coisa, então basta criatividade meu filho!

Depois disso será necessário adicionar o interceptor ao module.
Considerando um projeto pequeno o app.module.ts ou em um projeto maior no module responsável pelo interceptor.

Finalizado basta realizar o teste e pronto agora você tem um erro com as configurações que precisar para testar durante o desenvolvimento, mas vale ressaltar que também é possível fazer isso com testes unitários, porem essa é uma outra abordagem bem interessante quando queremos testar não apenas funcionalmente mas visualmente.

Bem por hoje é isso pessoal, fiquem com Deus e até mais.