**Domain Driven Design**

01 – Design de software e DDD

É um design de software, como vai desenhar a aplicação



02 – Entidade e caso de uso

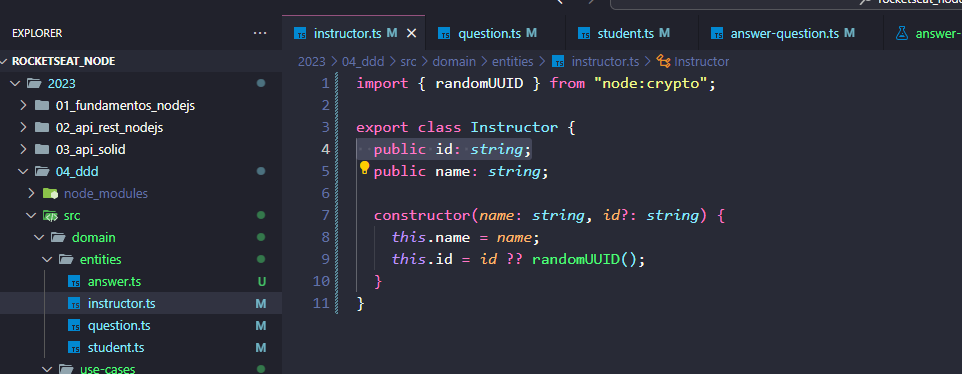
Fazer o comando npm init -y e npm i typescript e @types/node -D e depois roda o npx tsc –init

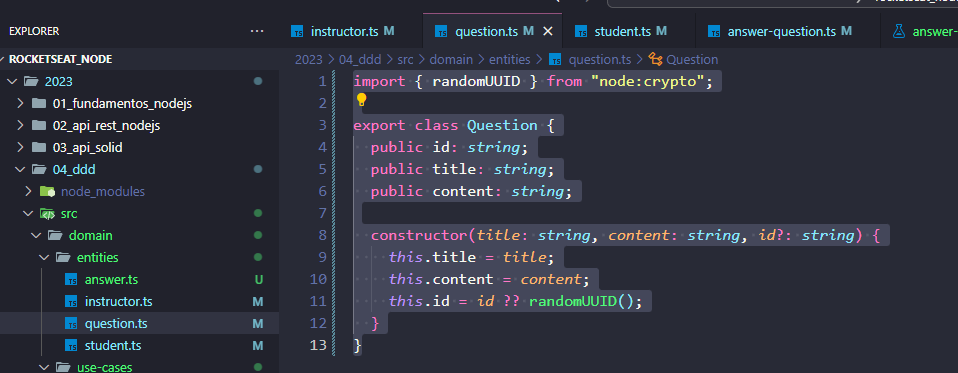


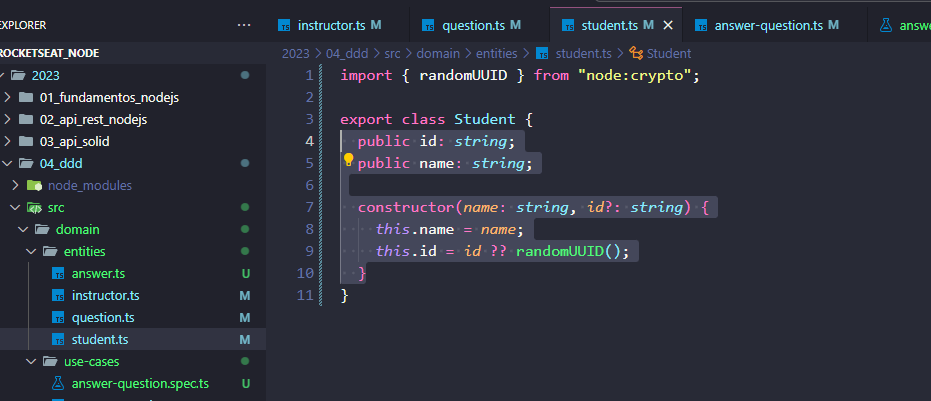


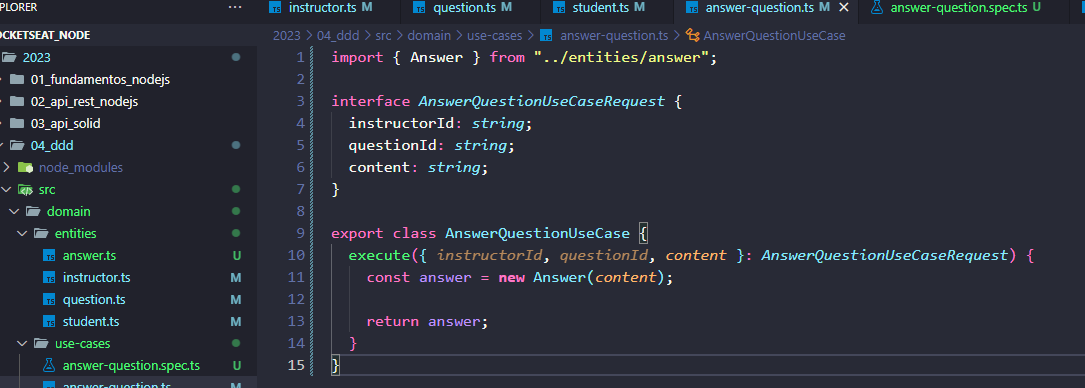
03 – Primeiro caso de uso

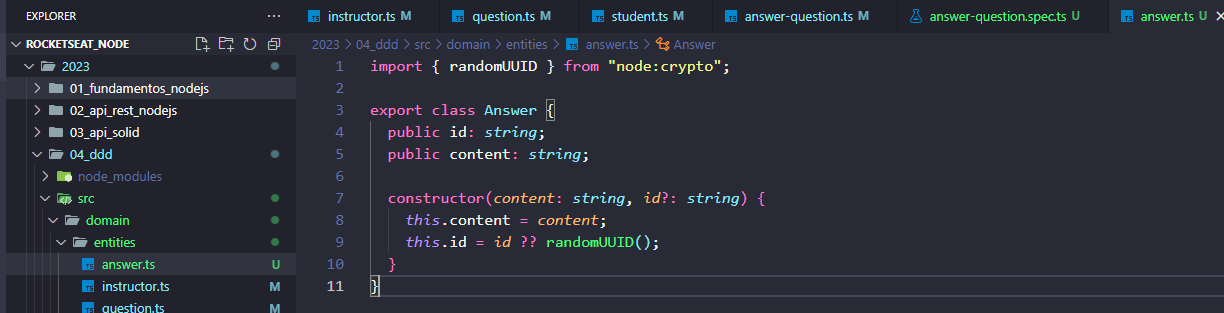
Instalar o npm i vitest -D

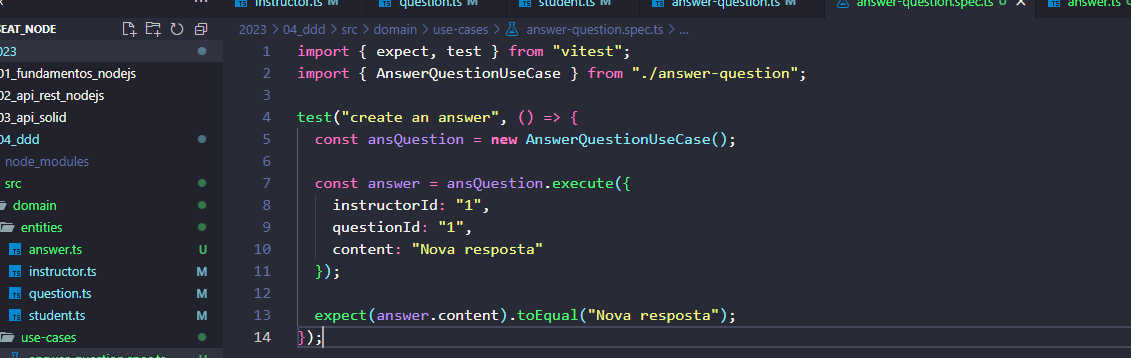






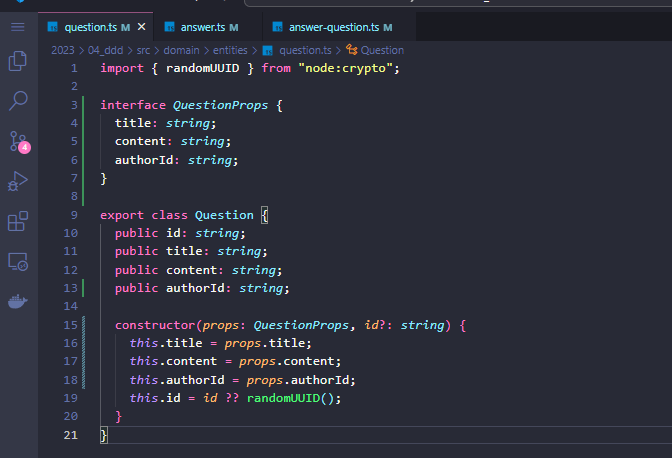


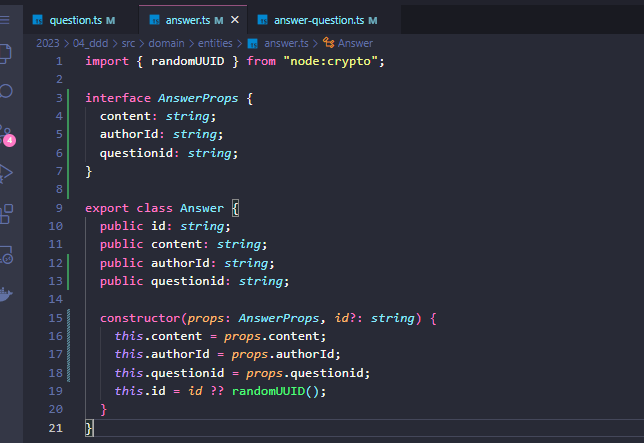


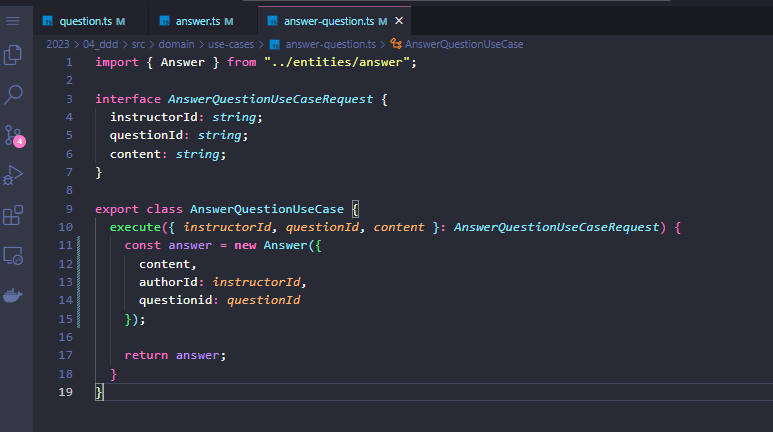


04 – Mapeando relacionamentos

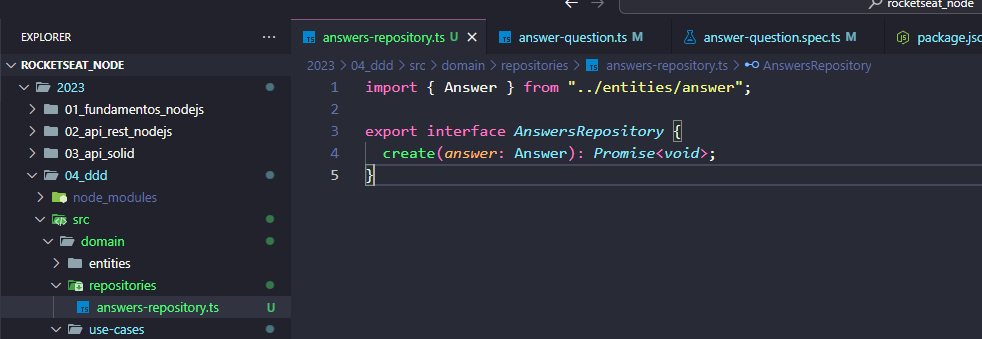
Dry = don’t repeat yourself

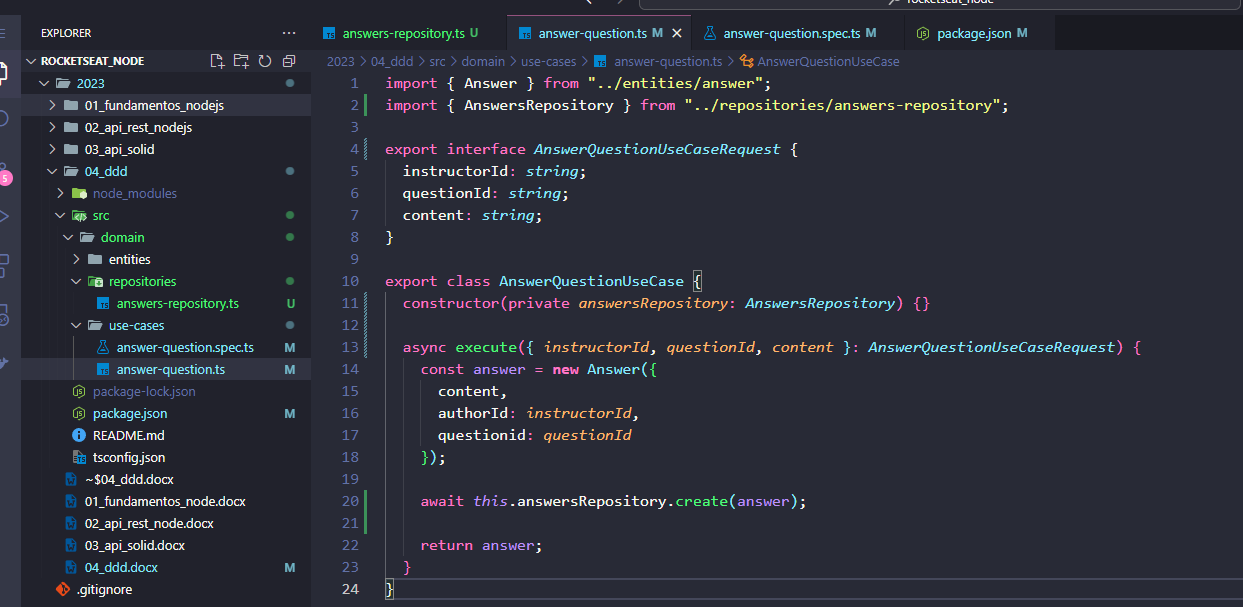


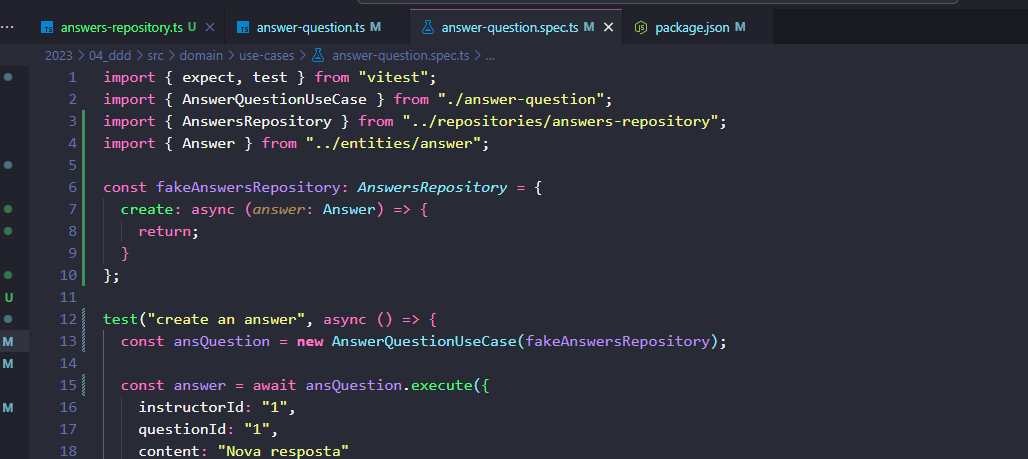


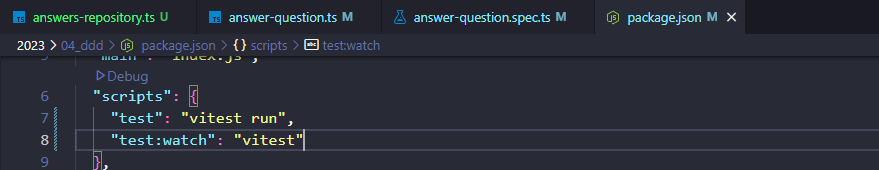


05 – Dependências externas

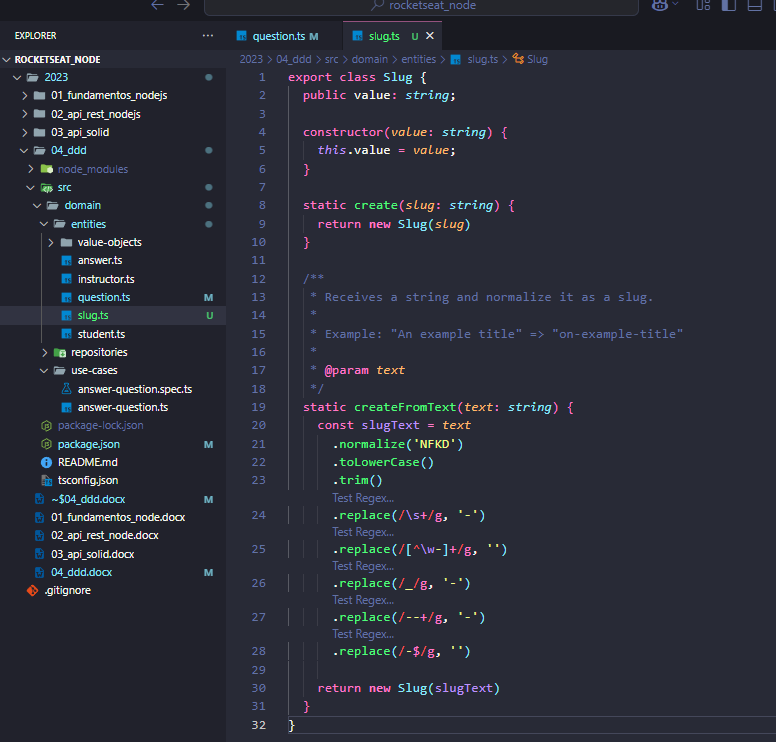


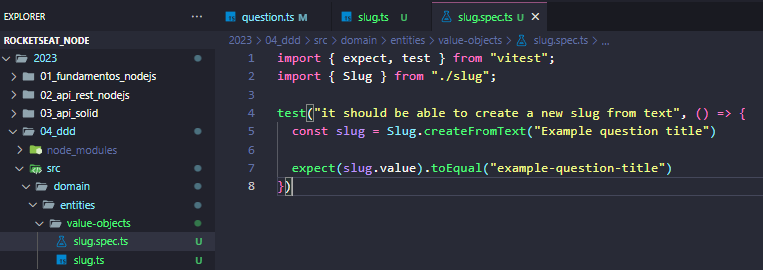




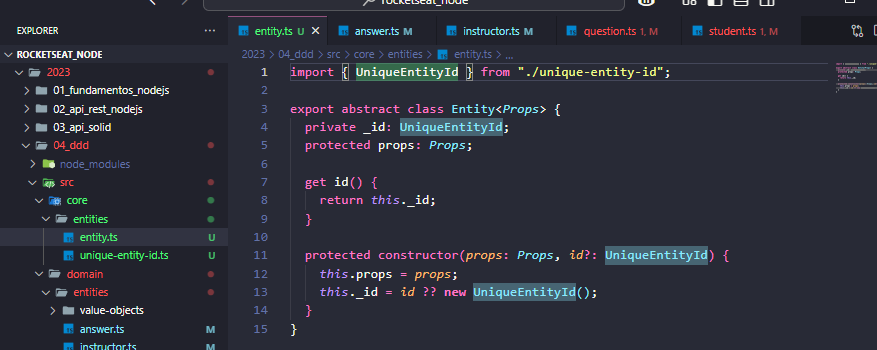


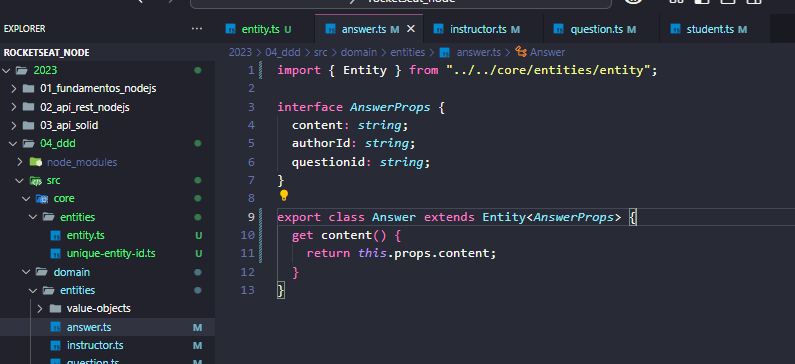
06 – Value object de slug

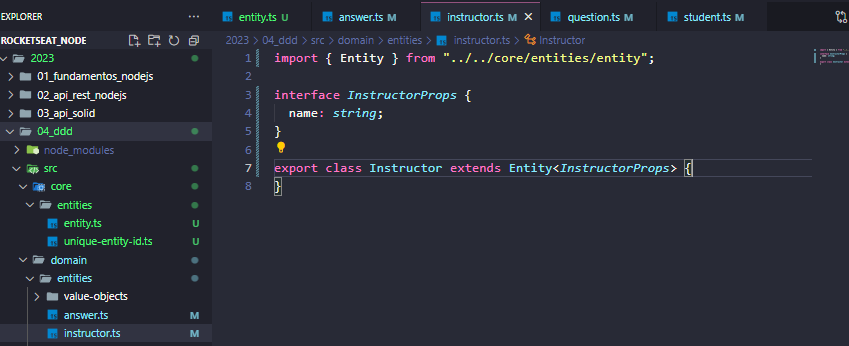


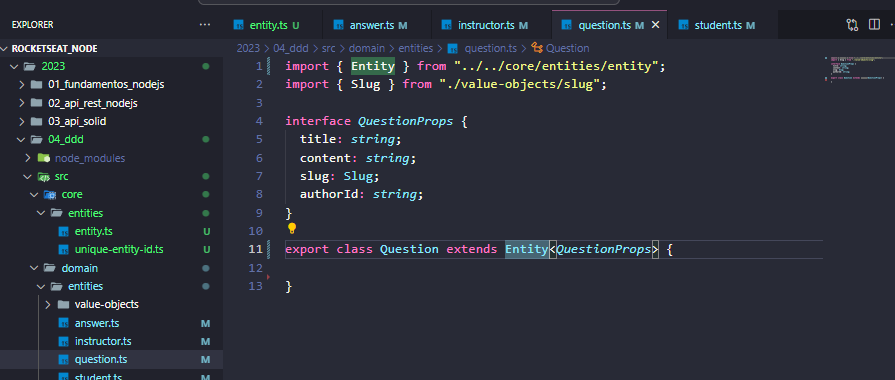


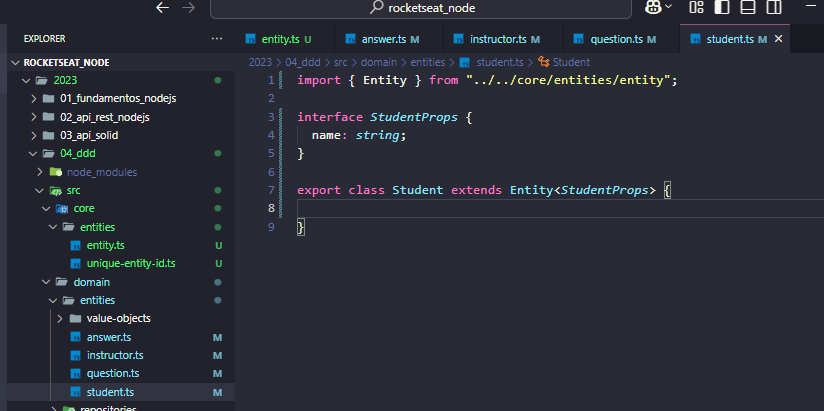
07 – Classe base de entidades







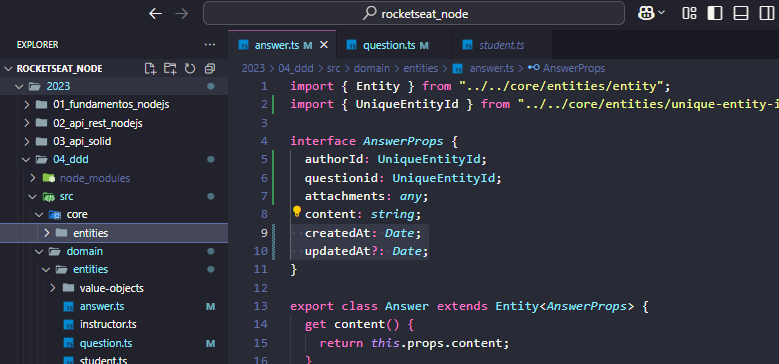


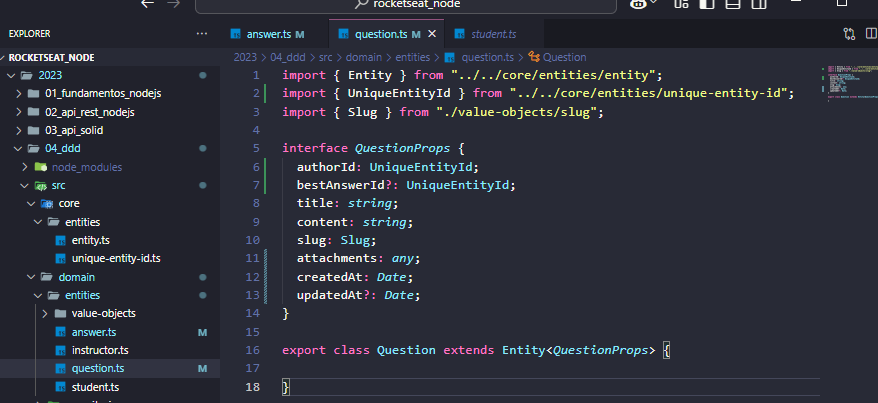


08 – ID das entidades

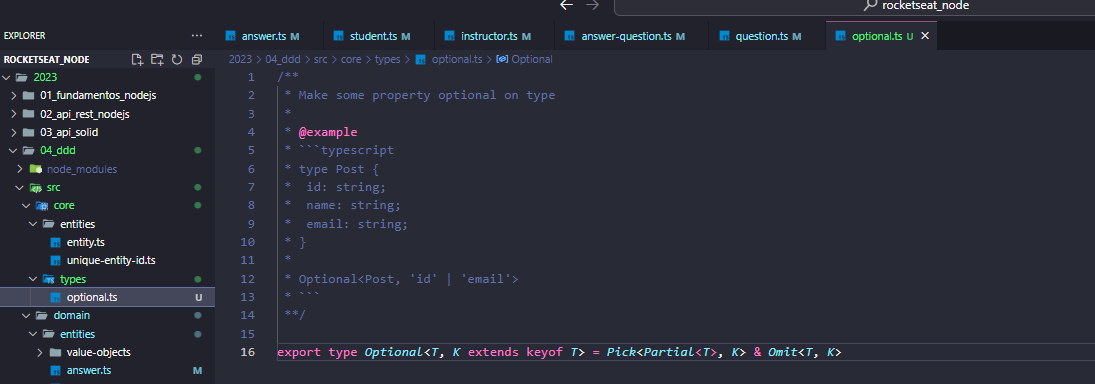


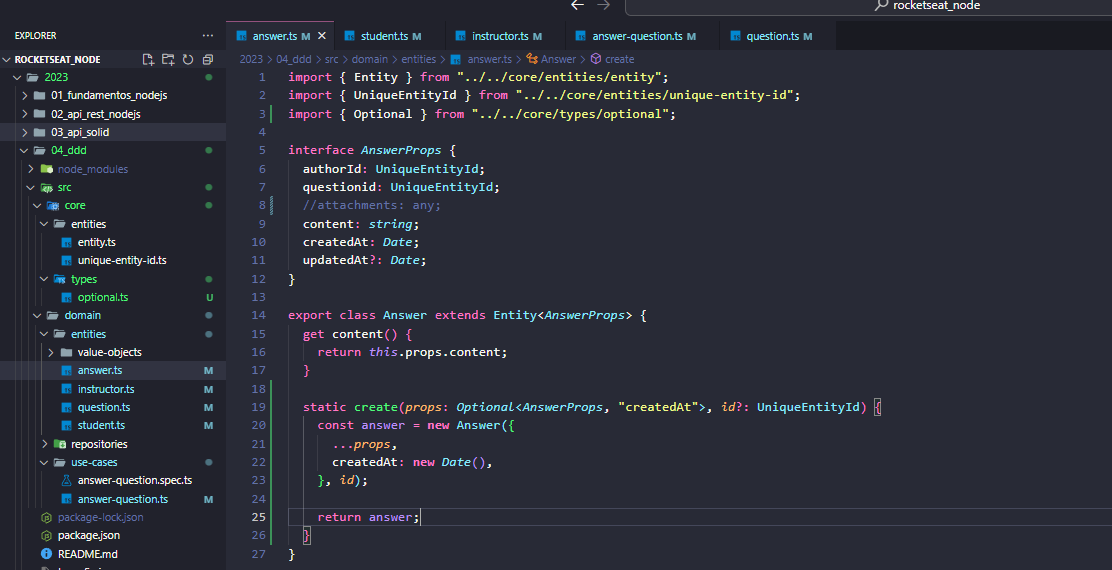
09 – Mapeando propriedades

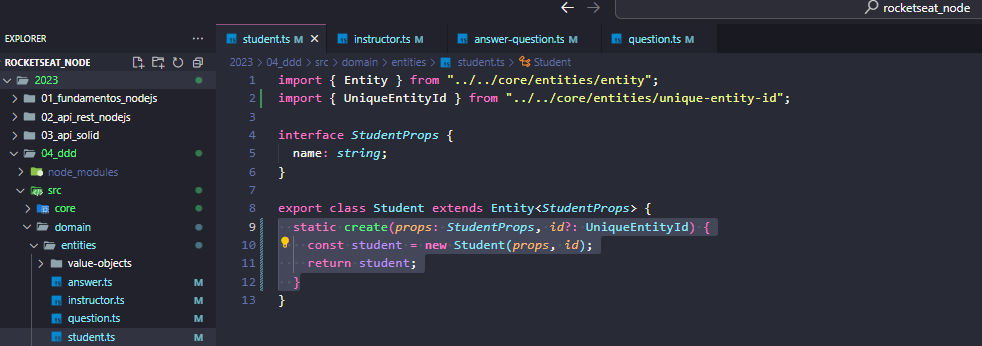


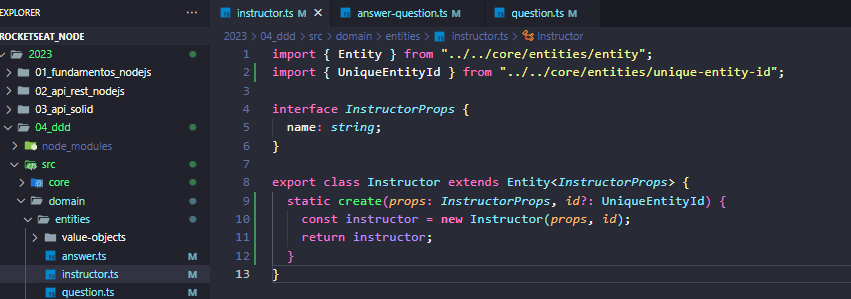


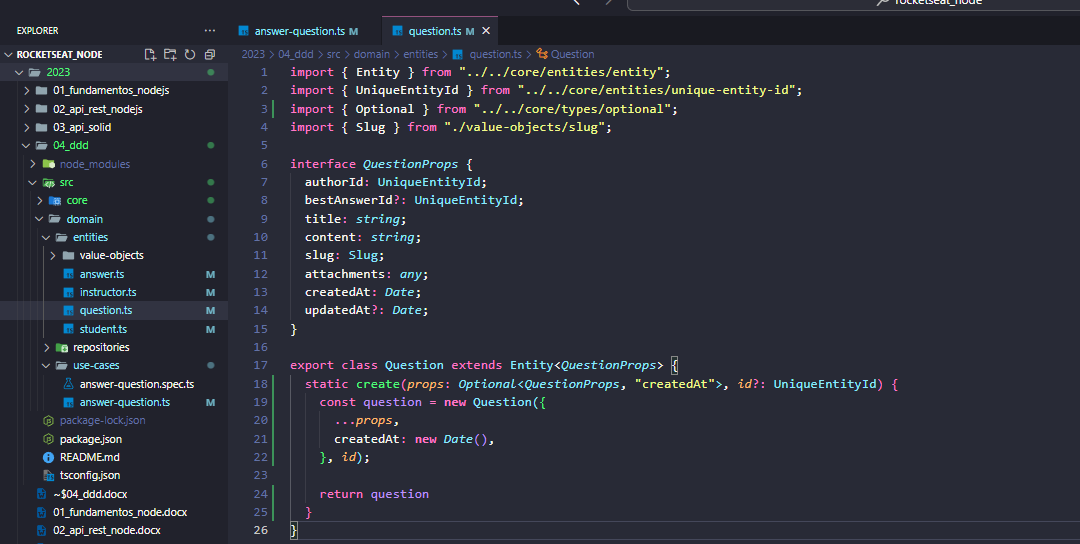
10 – Abstraindo criação de entidades

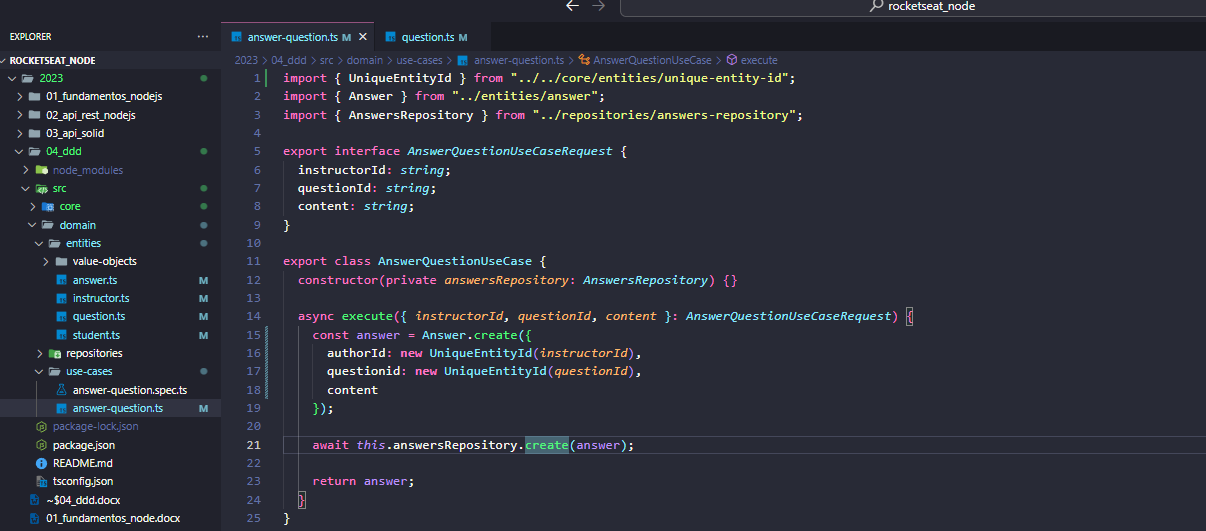






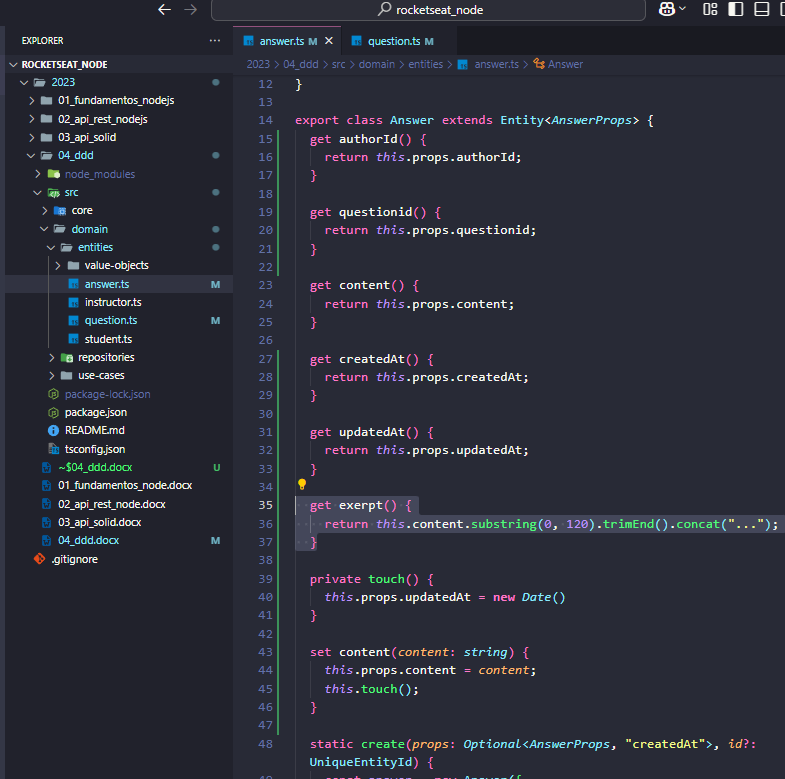


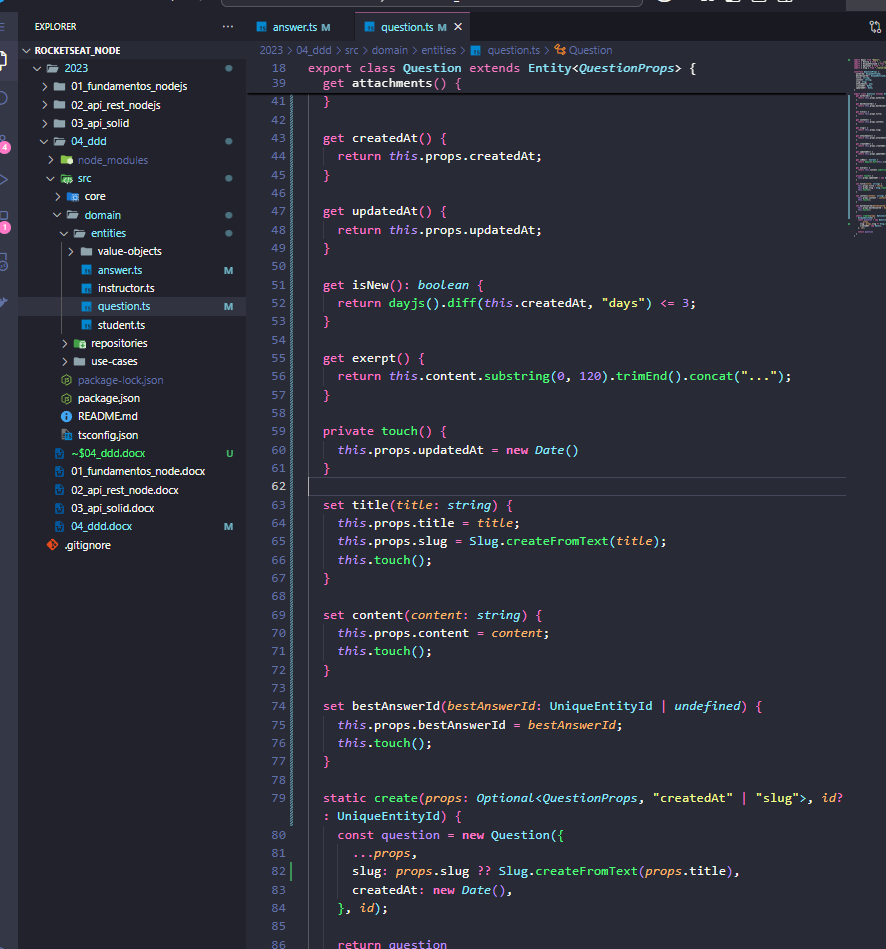




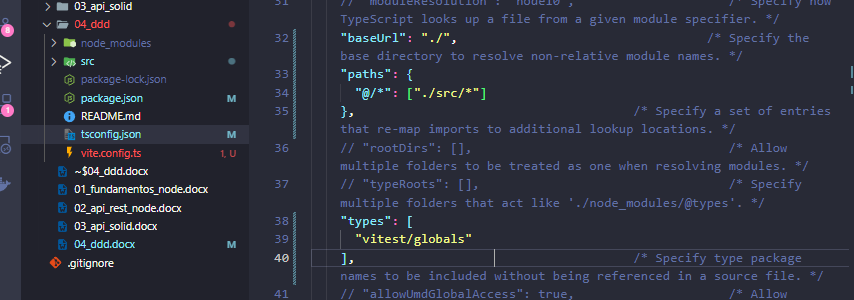
11 – Getters & setters das entidades

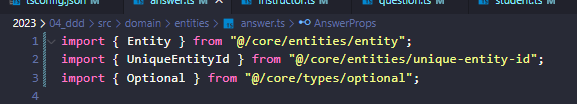
Instalar o npm i



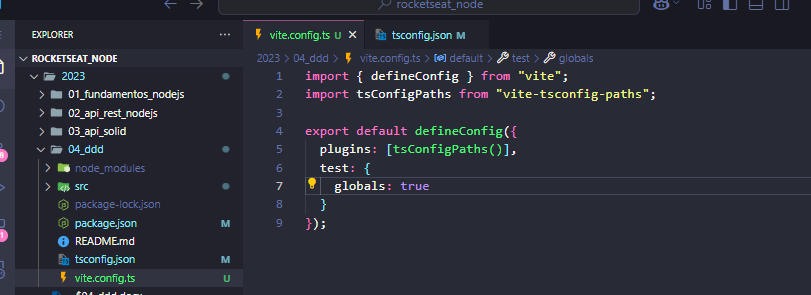


12 – Path aliases e vitest globals





E para usar no teste instale npm i vite-tsconfig-paths -D



13 – Configuração do ESLint

14 – Fundamentos e clean architecture

15 – Refatorando as pastas

16 – Caso de uso criar pergunta

17 – Refatorando os testes unitários

18 -

19 -

20 -

21 -

22 -

23 -

24 -

25 -

26 -

27 -

28 -

29 -

30 -