**API Node JS com SOLID**

01 – Definindo requisitos e regras



02 – Criando projeto node js

No terminal npm init -y e npm i typescript @types/node -D e npm i tsx tsup -D depois no terminal executar npx tsc –init



Instalar npm i fastify







03 – Usando versões exatas do npm

Deixa as versão do npm fixa



04 – Carregando variáveis ambiente

Instalar o npm i dotenv e npm i zod







05 – Criando aliases de importação





06 – Fundamentos do prisma ORM

Instalar o npm i prisma -D e para inicializar o prisma npx prisma init



Para criar as tabelas



Depois execute o comando npx prisma generate para gerar as tipagens e depois instalar o npm i @prisma/client

07 – PostgreSQL com Docker

Para achar imagems no Docker <https://hub.docker.com/search> usar essa <https://hub.docker.com/r/bitnami/postgresql> e para criar a imagem docker run –name api-solid-pg -e POSTGRESQL\_USERNAME=Docker -e POSTGRESQL\_PASSWORD=Docker -e POSTGRESQL\_DATABASE=apisolid -p 5432:5432 bitnami/postgresql

08 – Docker compose

Para subir o container docker composse up -d e depois para criar as tabela roda o comando npx prisma migrate dev e para visualizar a tabela npx prisma studio





09 – Criando schema do prisma

Depois de alterar as tabela no prisma npx prisma migrate dev



10 – Relacionamentos entre tabelas

1 – 1 => Um para um

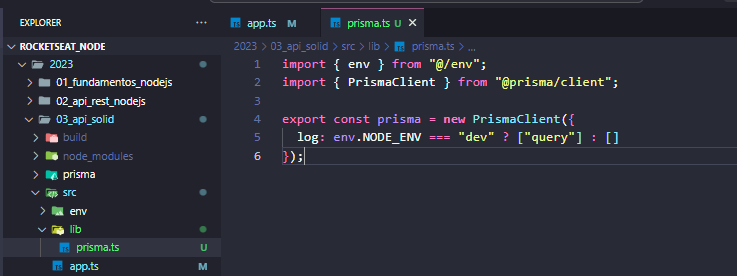
1 – N => Uma para muitos

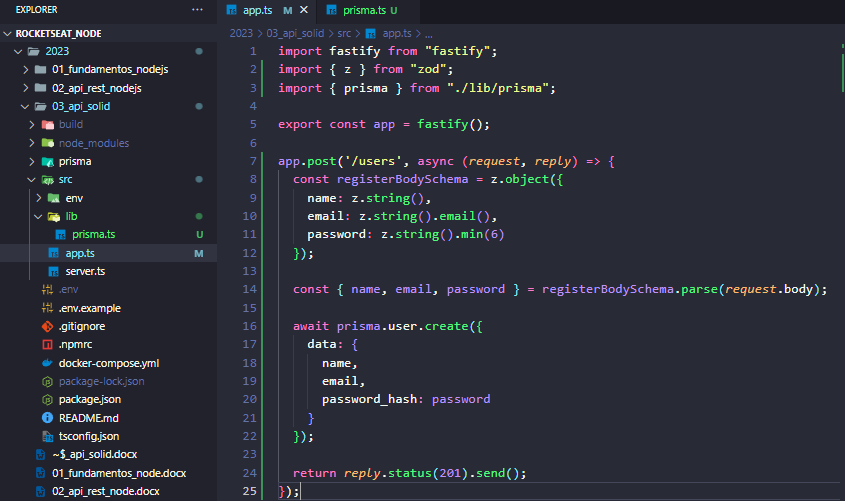
N – N =>Muitos para muitos



Depois npx prisma migrate dev

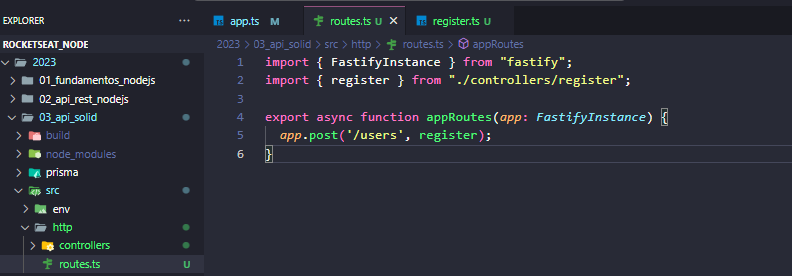
11 – Criação de um usuário

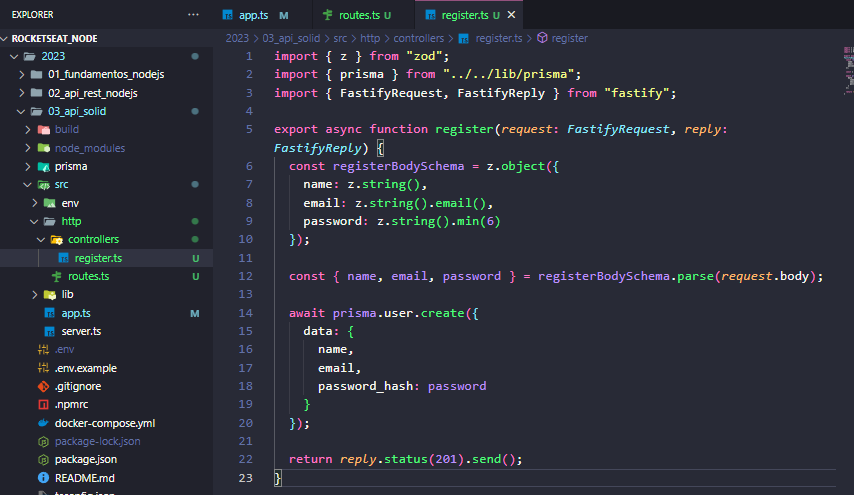




12 – Controller de registro

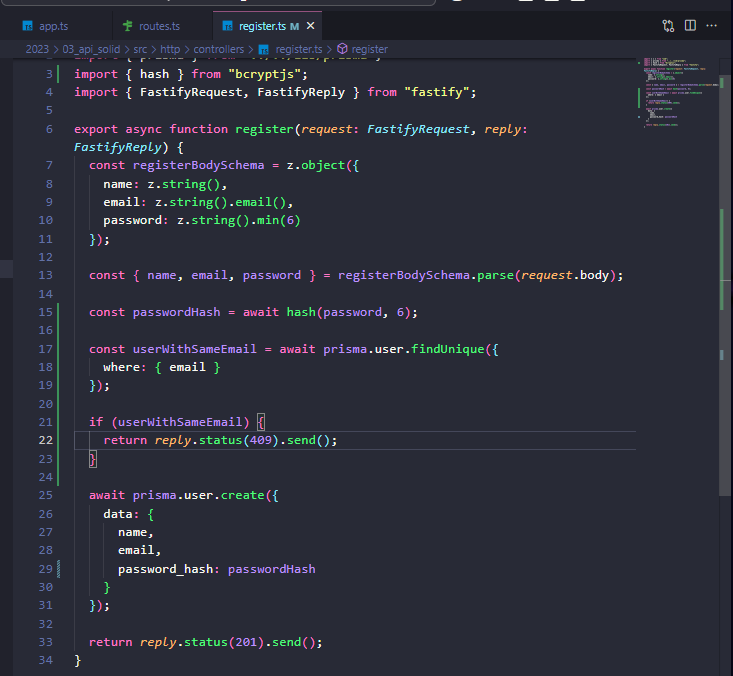




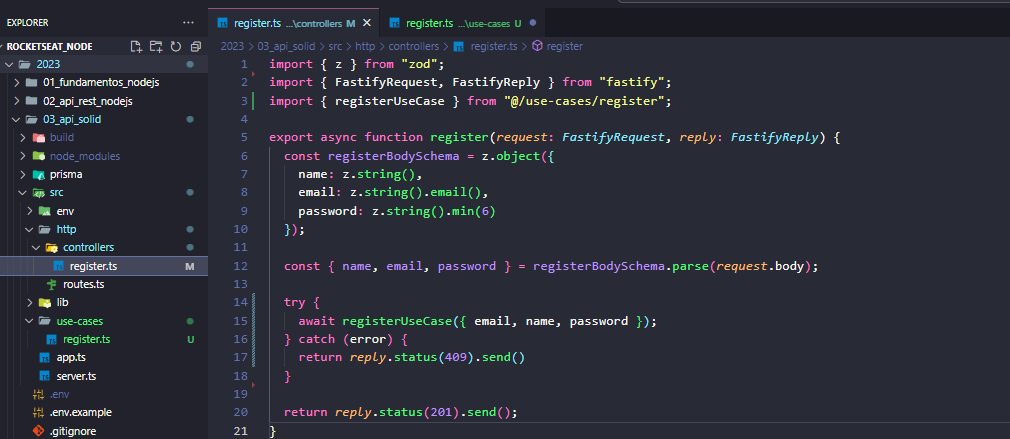


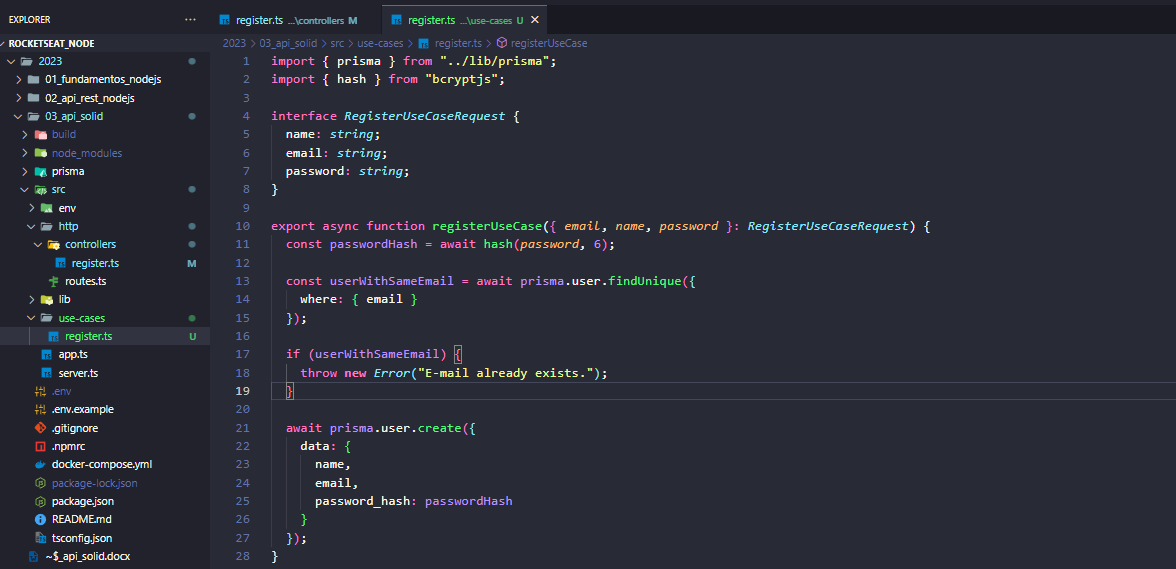
13 – Hash de senha e validação

Instalar npm i bcryptjs e depois npm i @types/bcryptjs -D



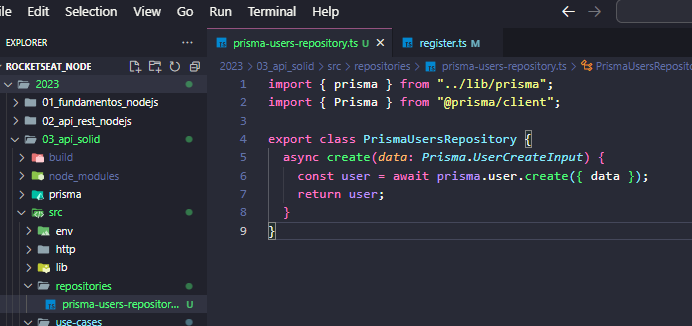
14 – Caso de uso de registro



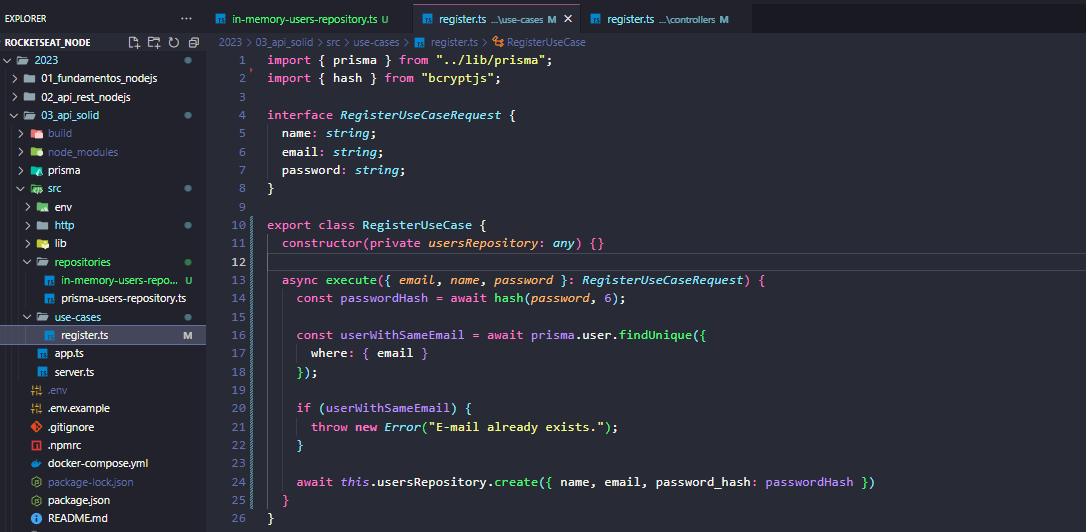


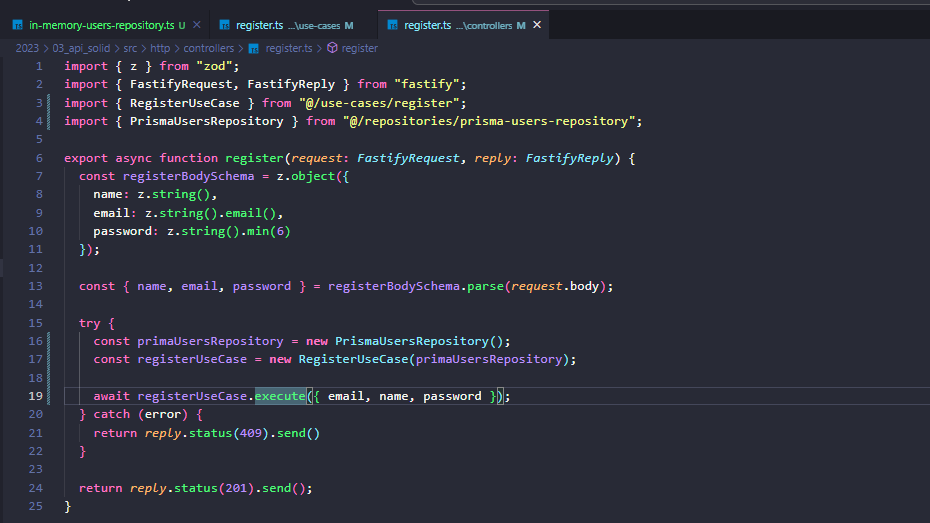
15 – Repository patterns



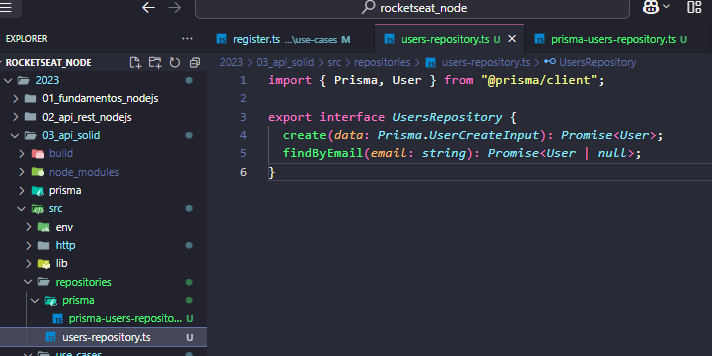


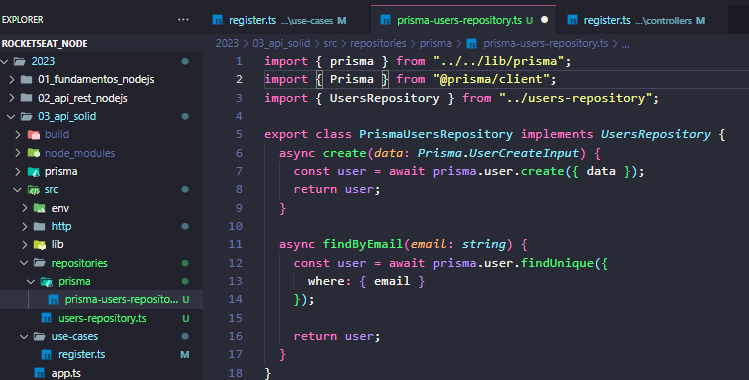
16 – Inversão de dependência

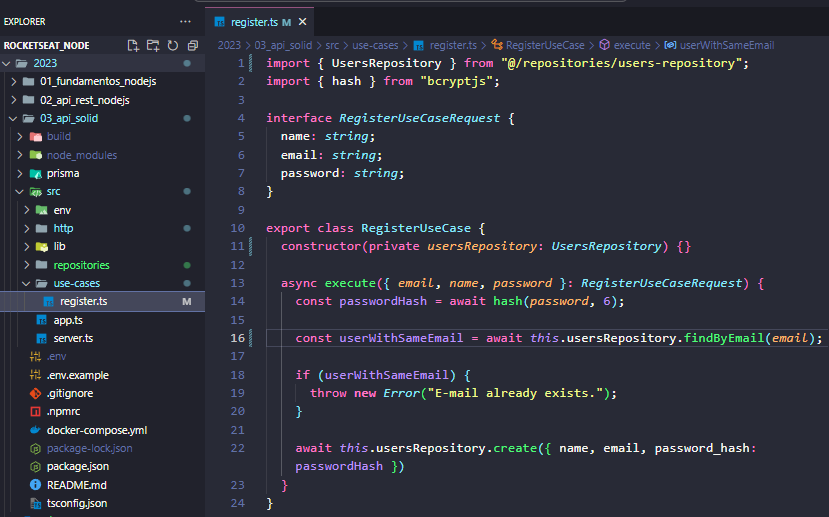




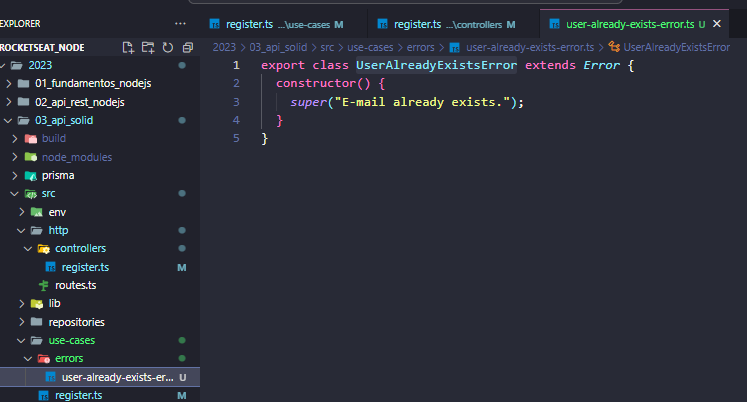
17 – Interaface do repositorio

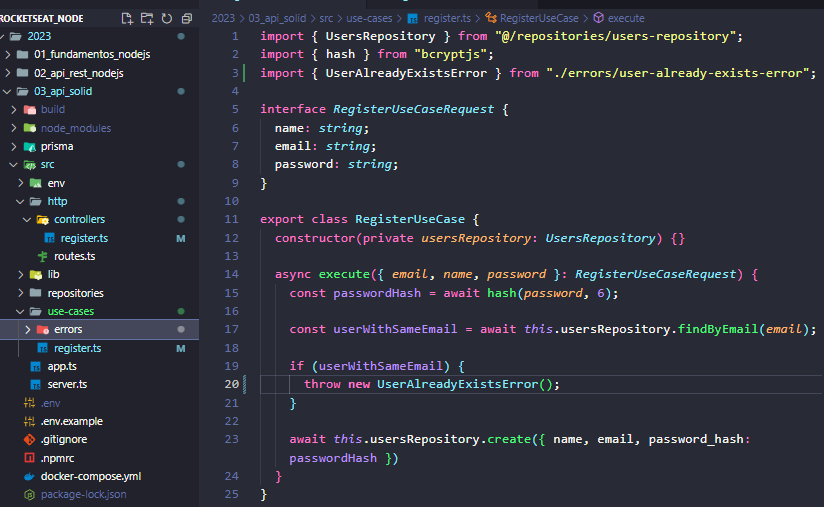


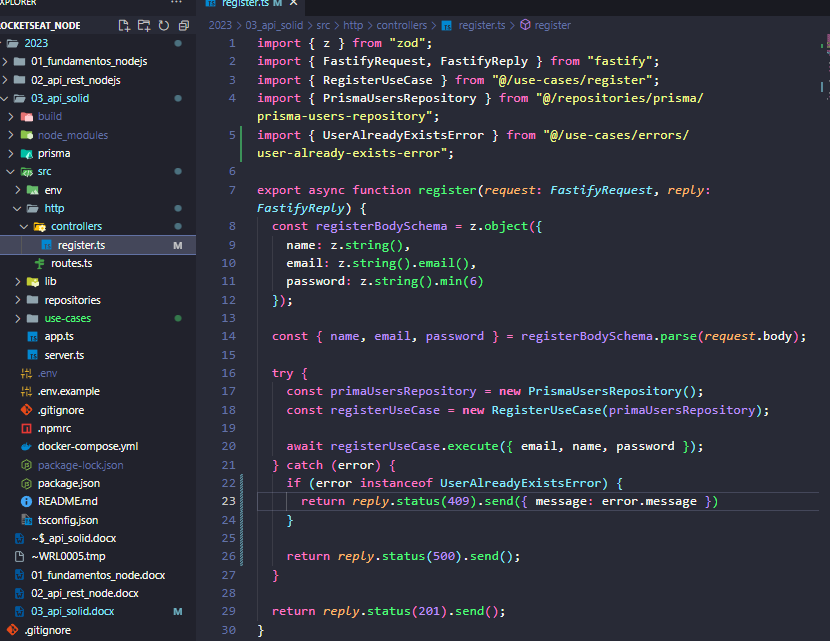




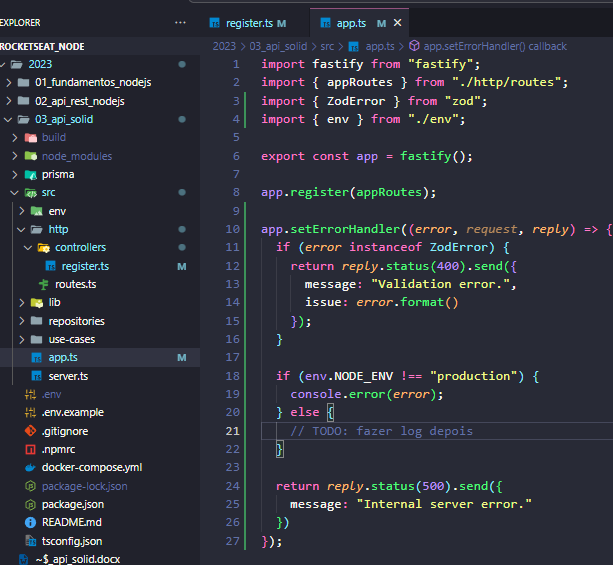
18 – Lidando com erros de use case

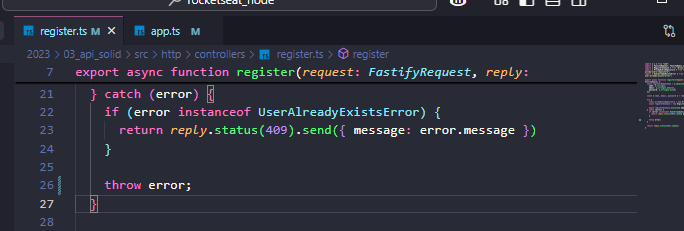






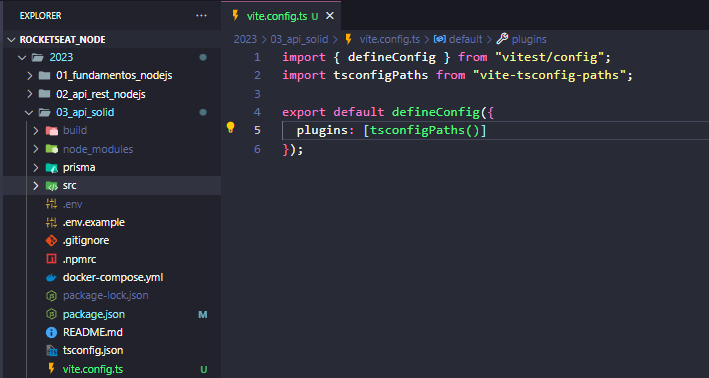
19 – Handle de error global

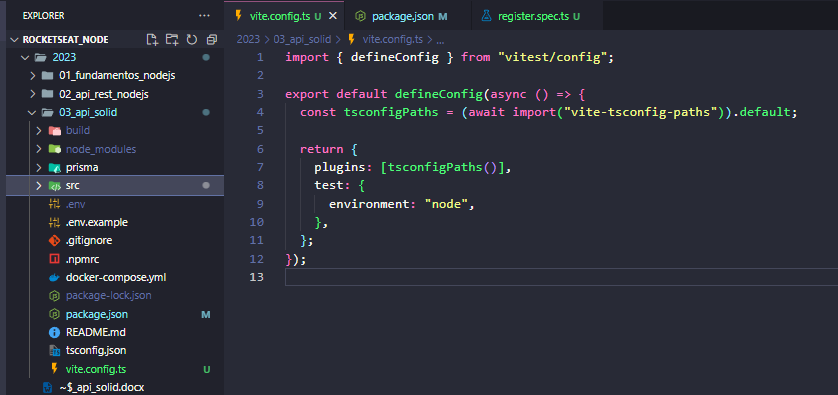


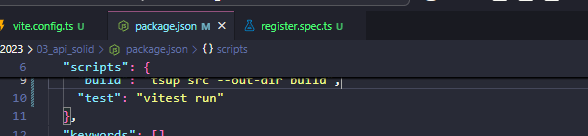


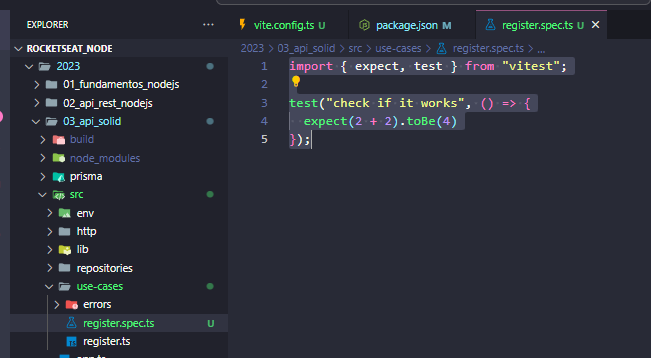
20 – Configurando o vitest

Instalar npm i vitest vite-tsconfig-paths -D

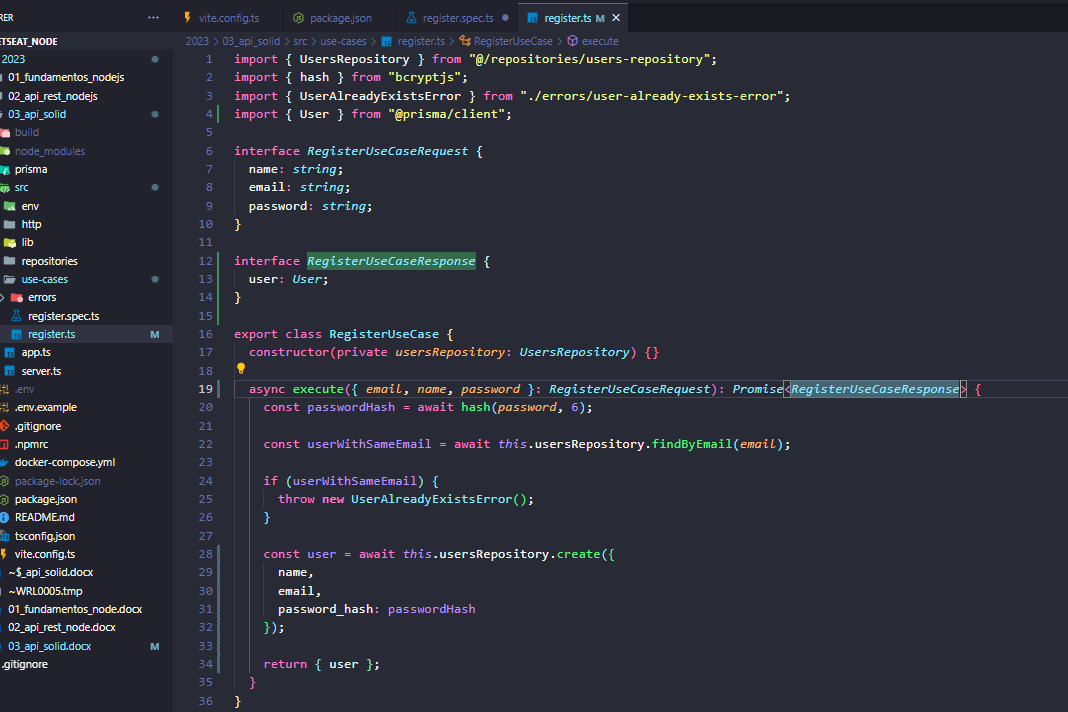


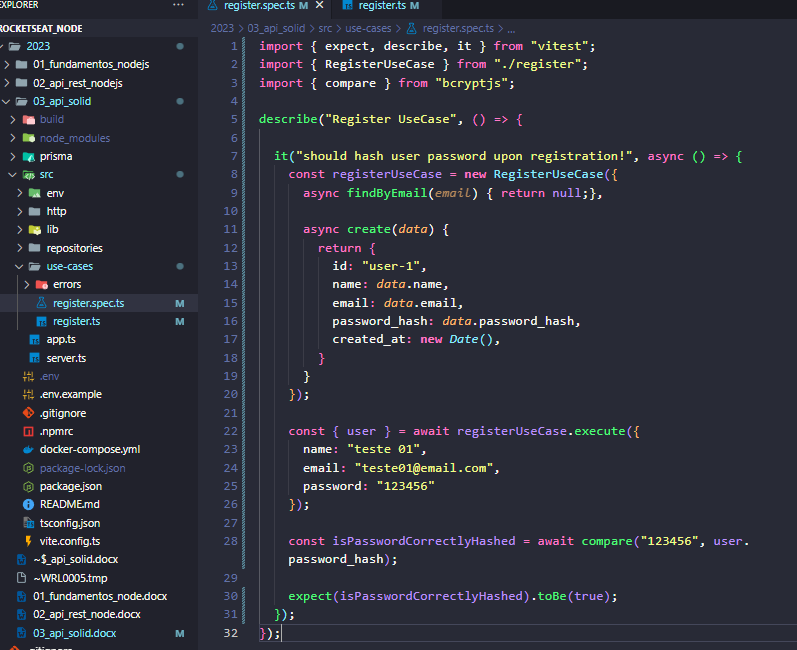




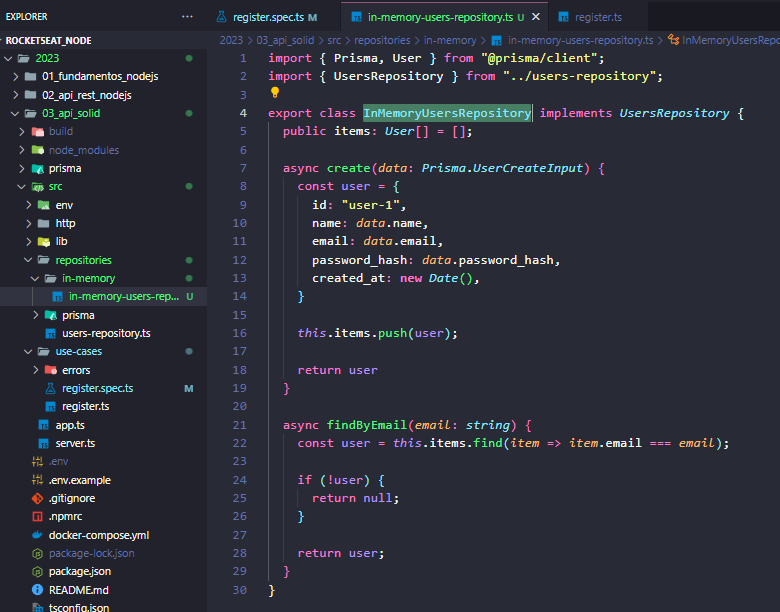


21 – Primeiro teste unitário

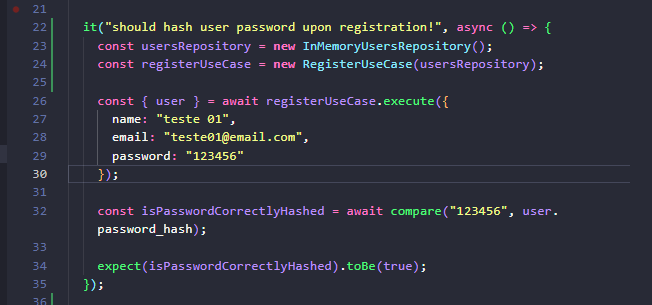


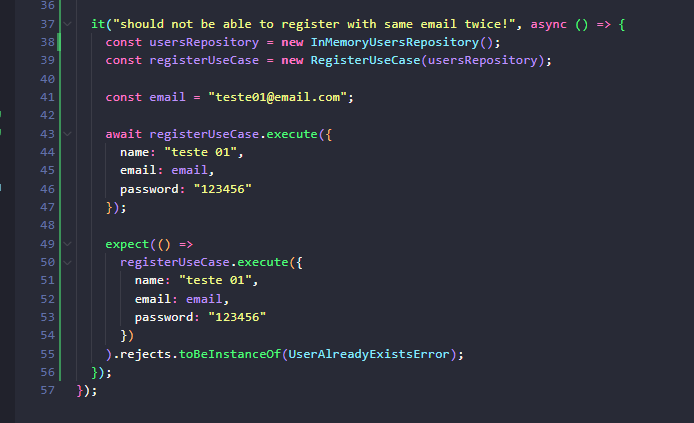


22 – In-Memory database

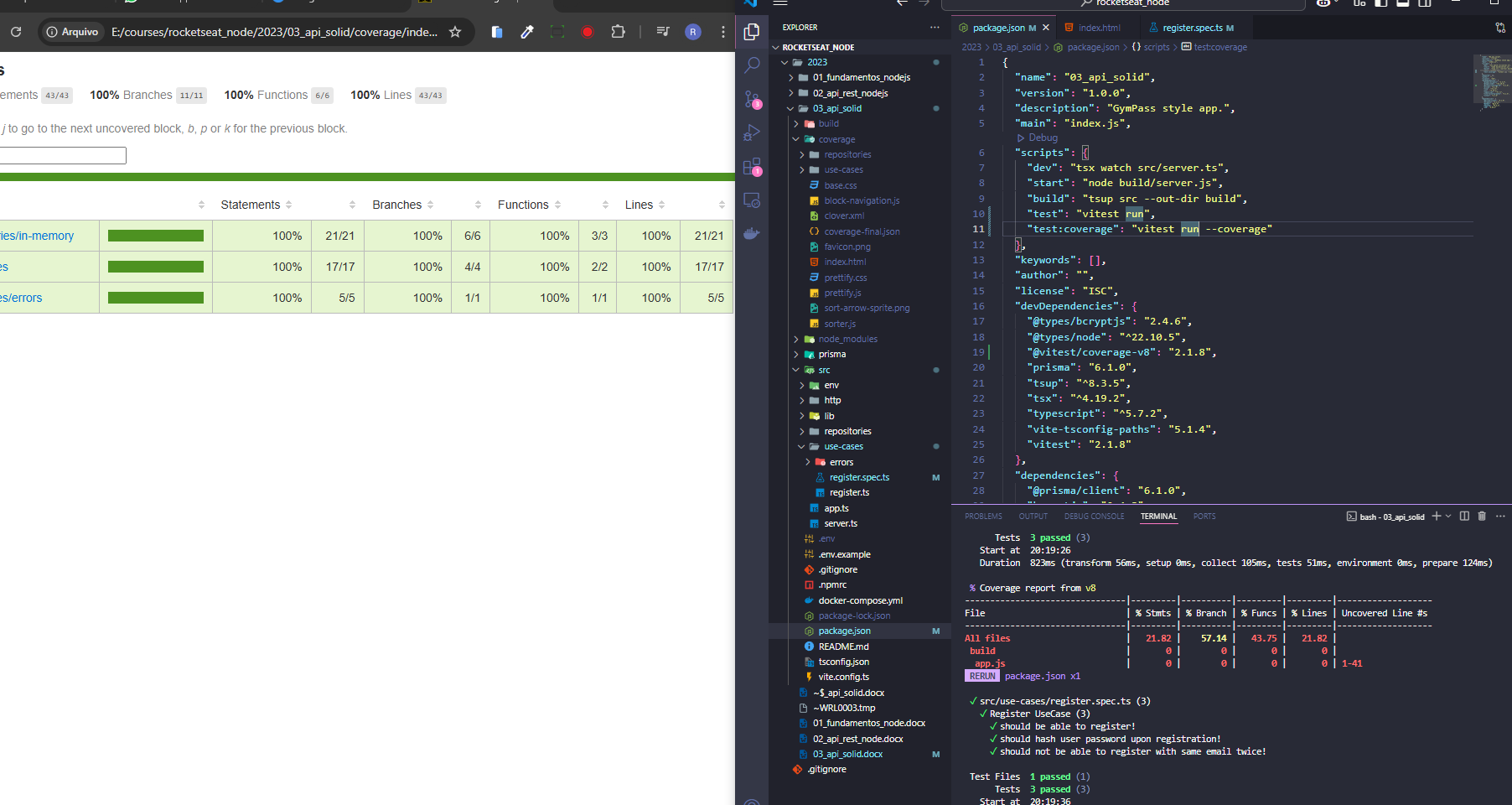




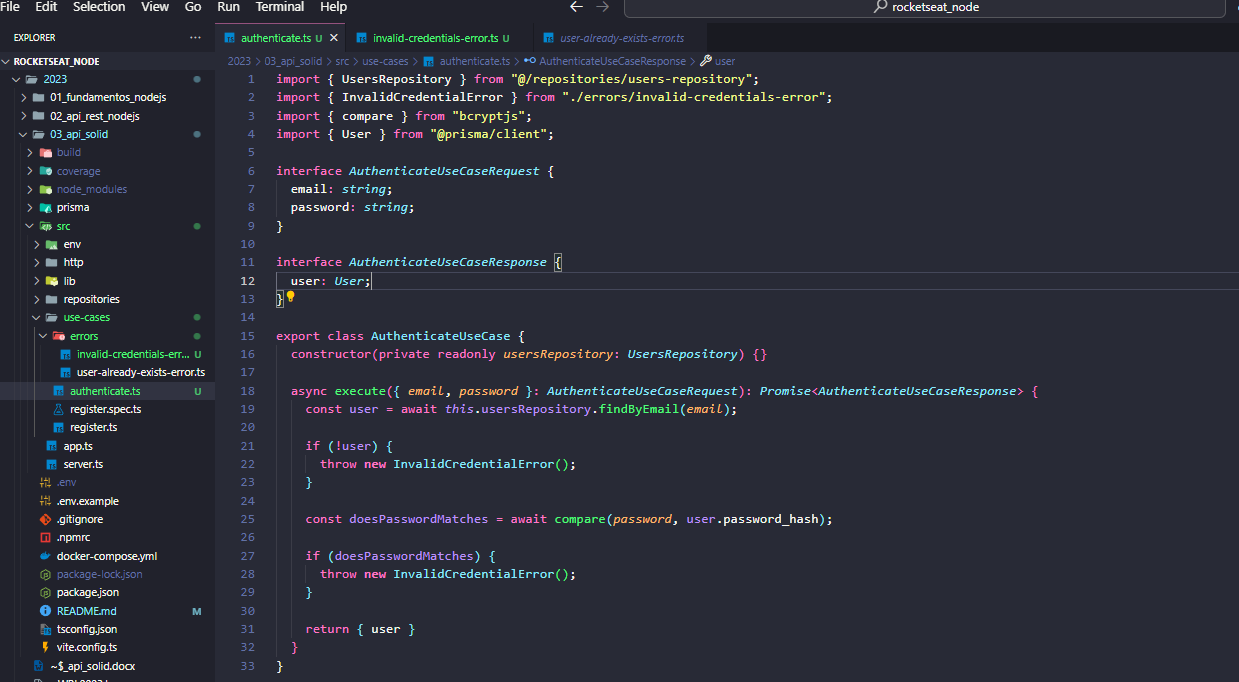


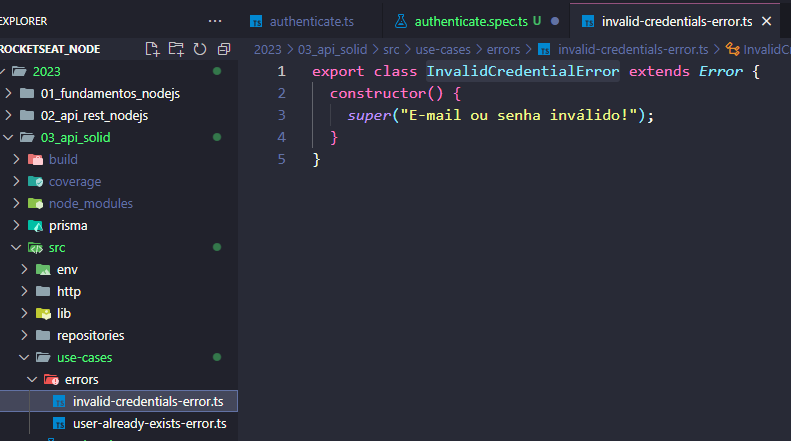


23 – Gerando coverage de testes

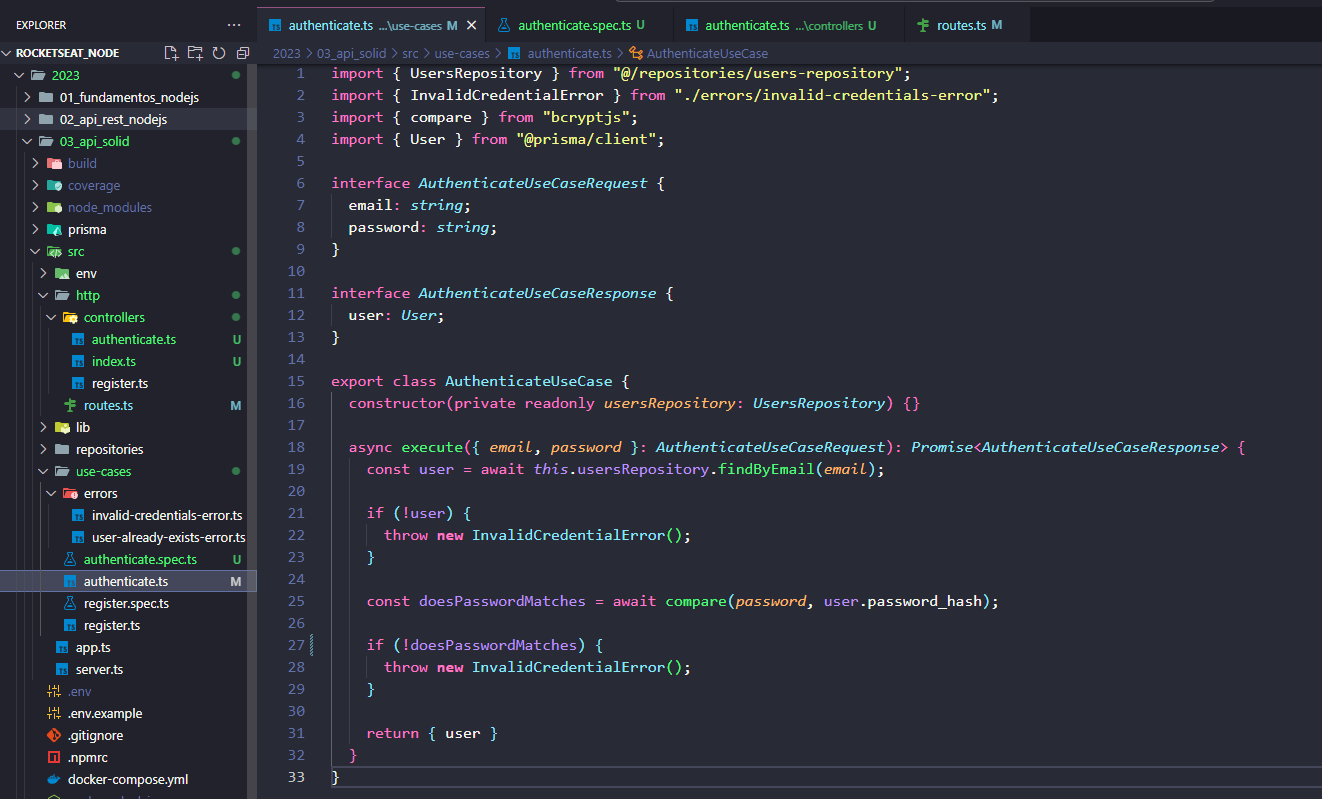


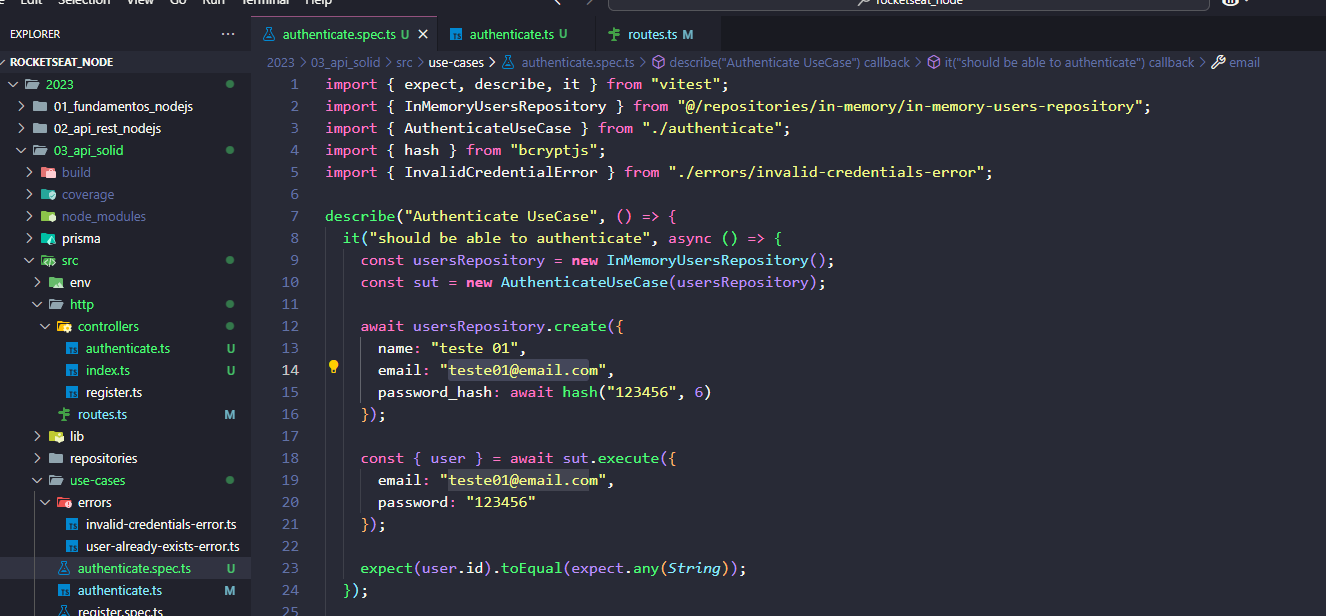
24 – Caso de uso de autenticação



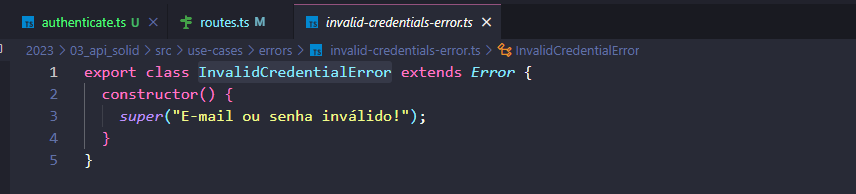


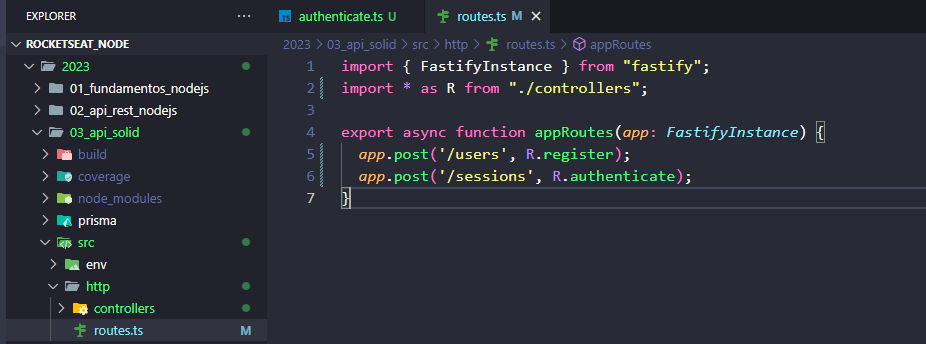
25 – Teste e controler de autenticação

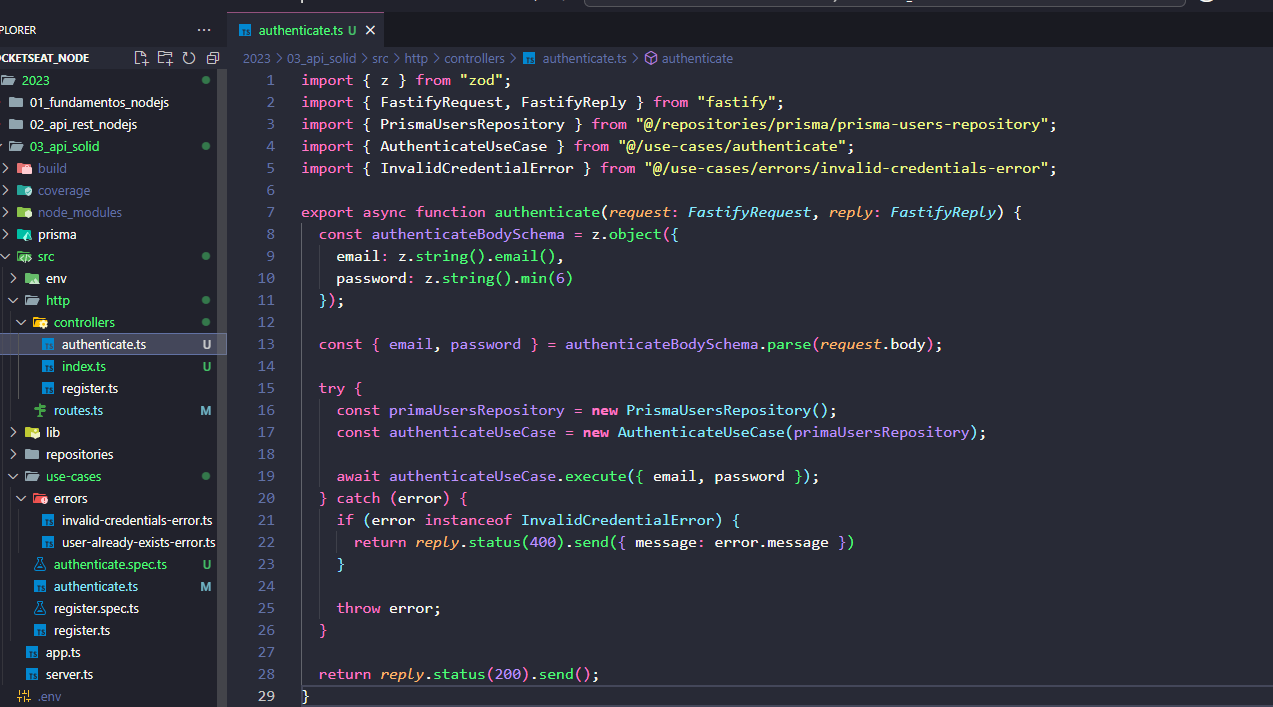




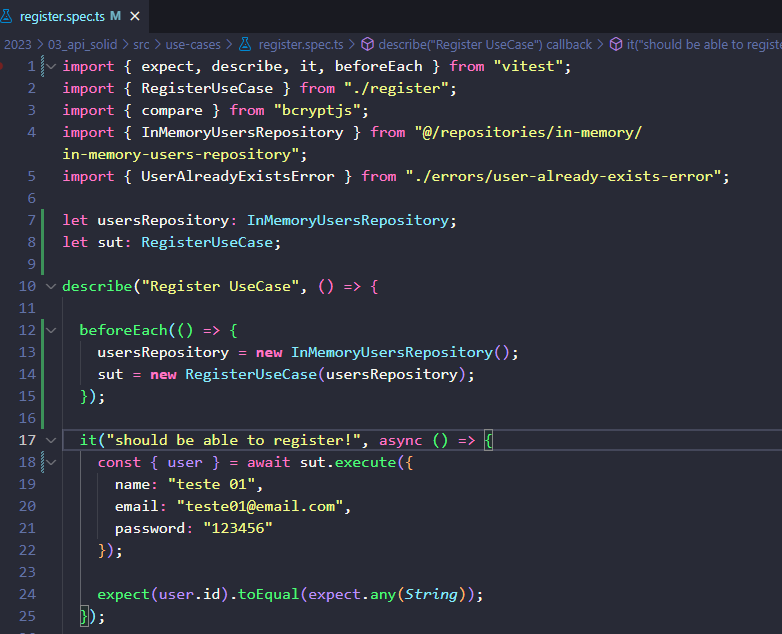




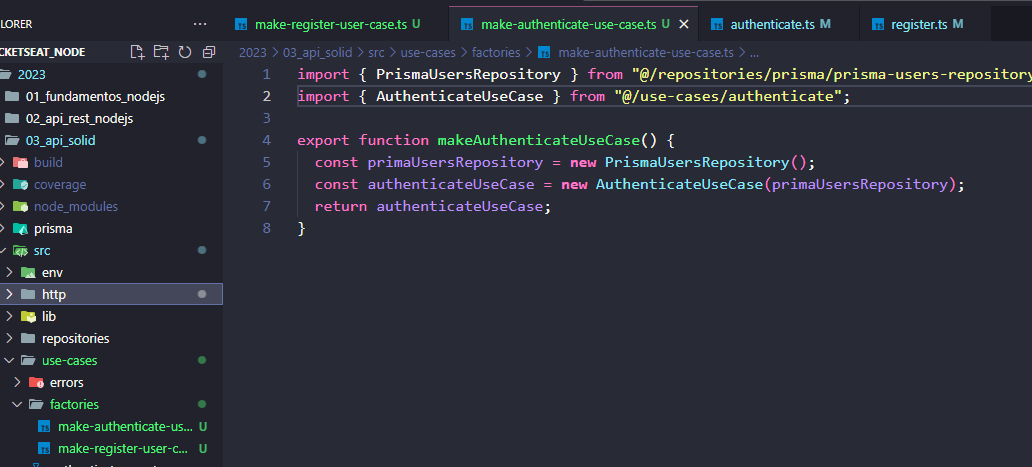


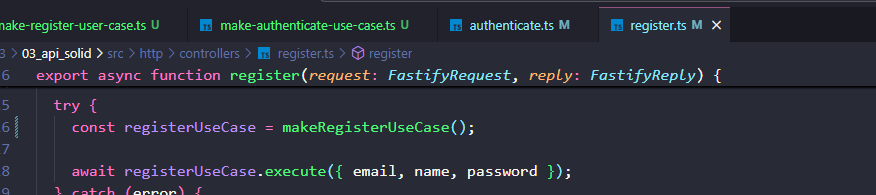


26 – Refatorando instancias no testes

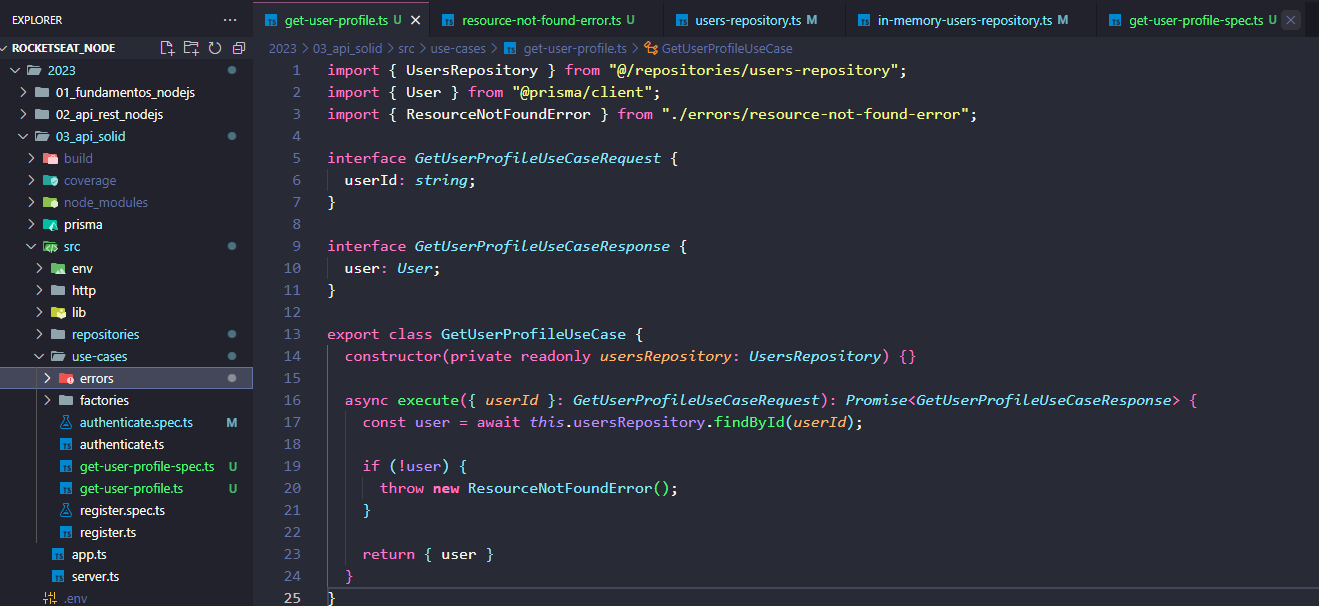


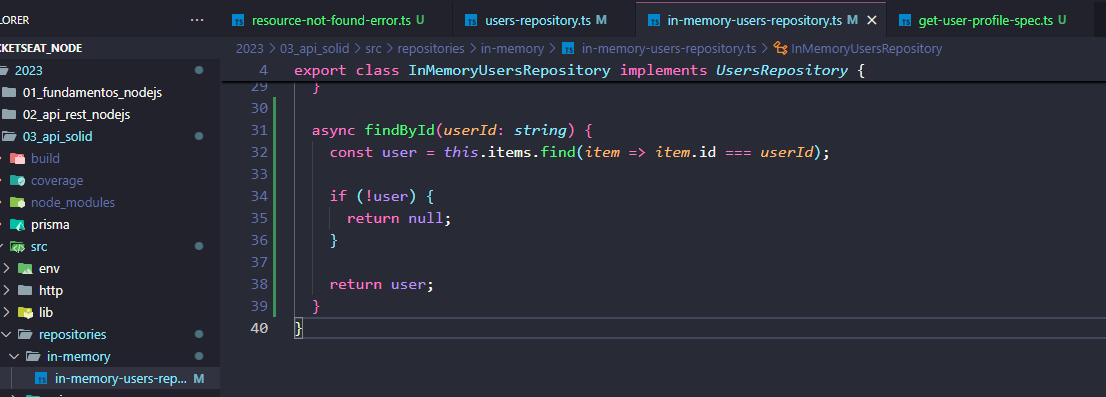
27 – Utilizando factory pattern

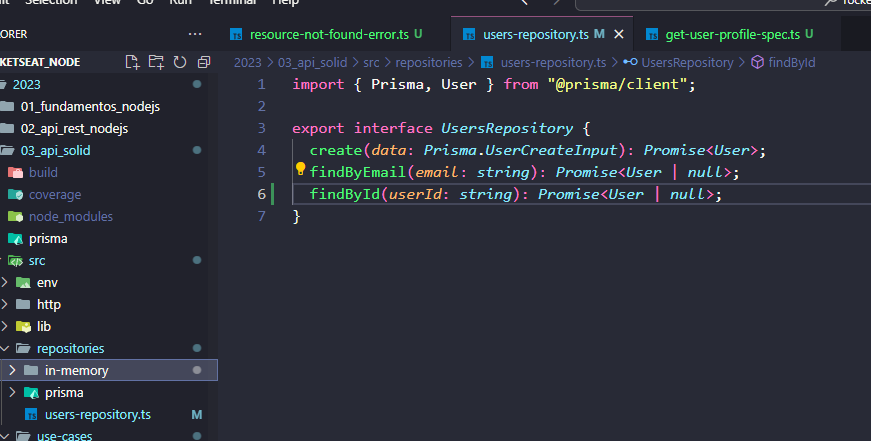


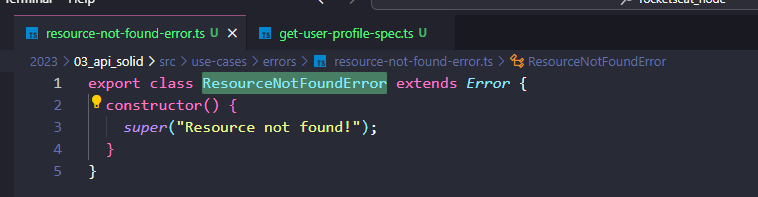


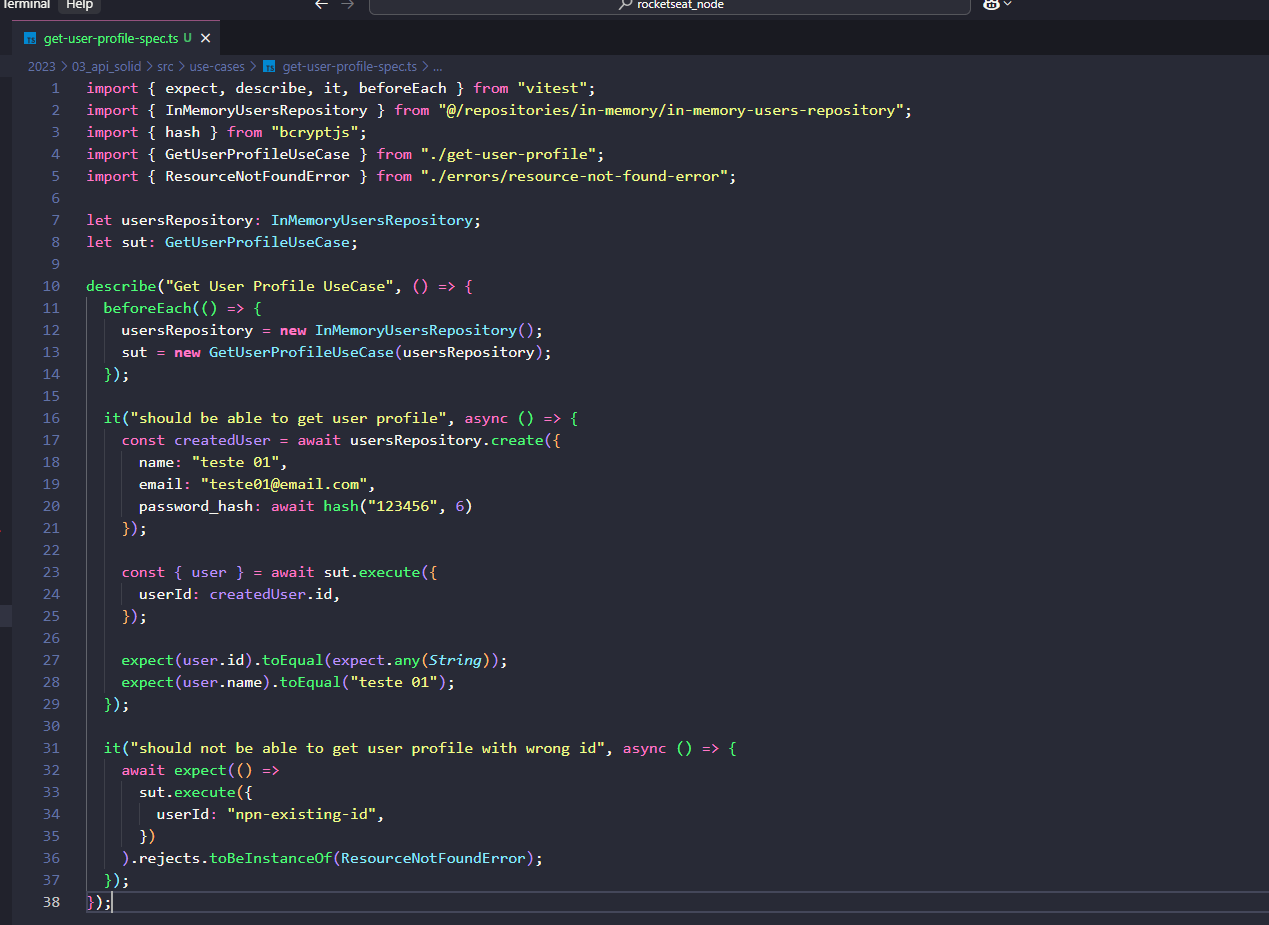
28 – Caso de uso de perfil



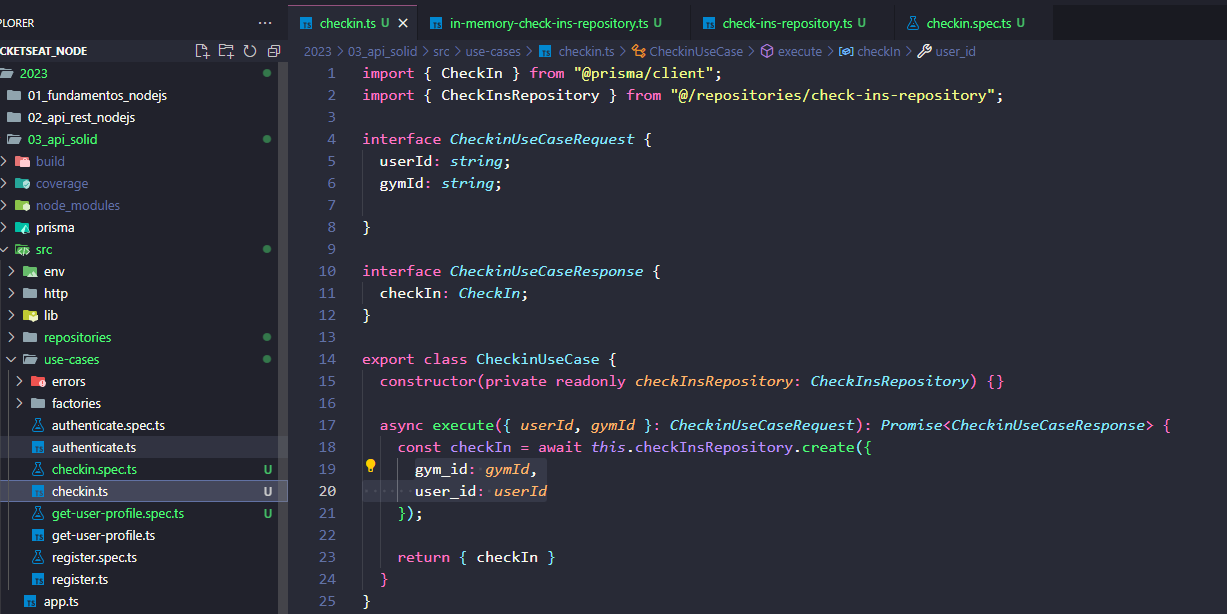


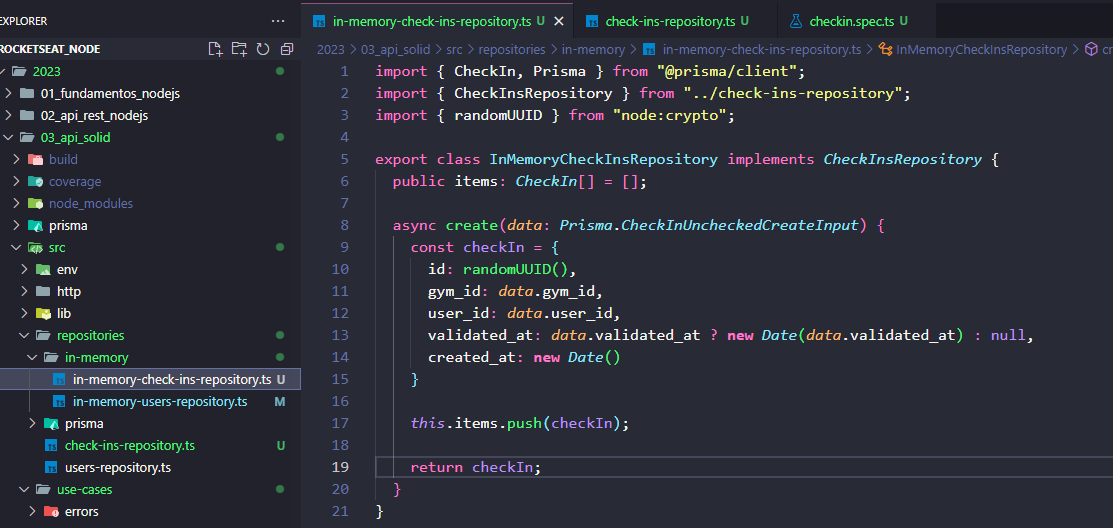


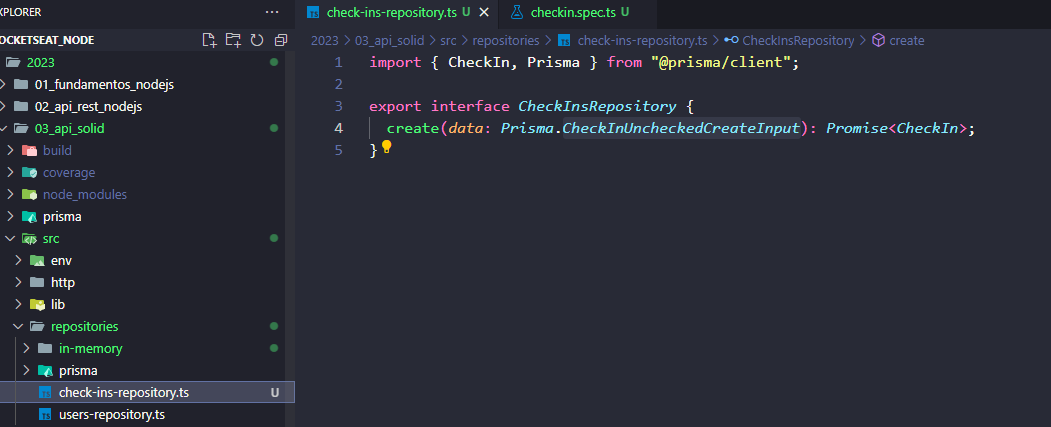


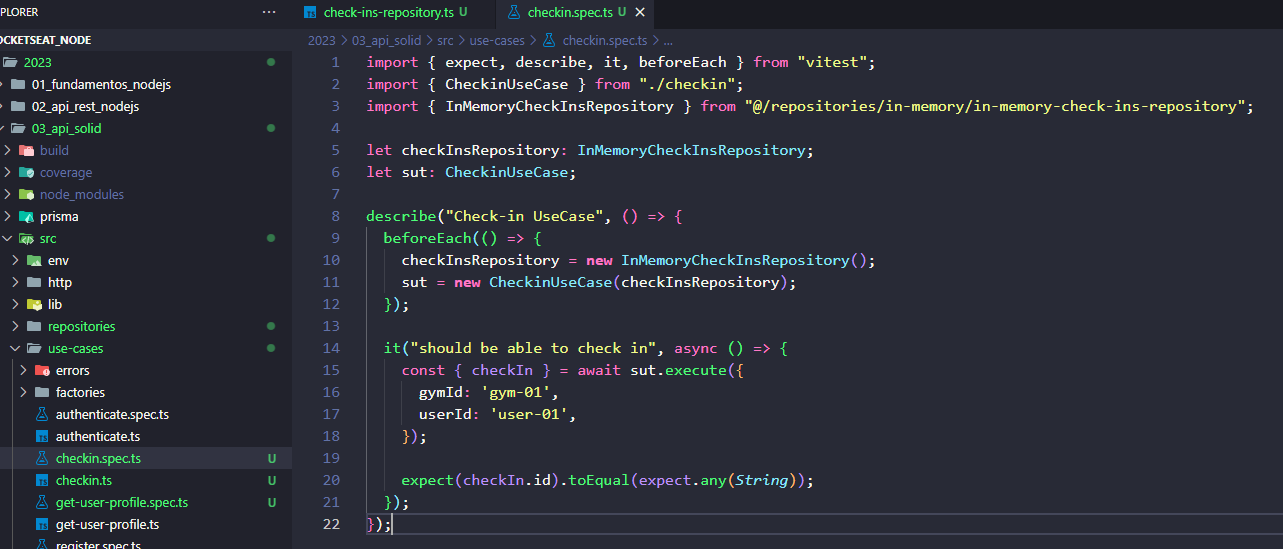


29 – Caso de uso de check-in

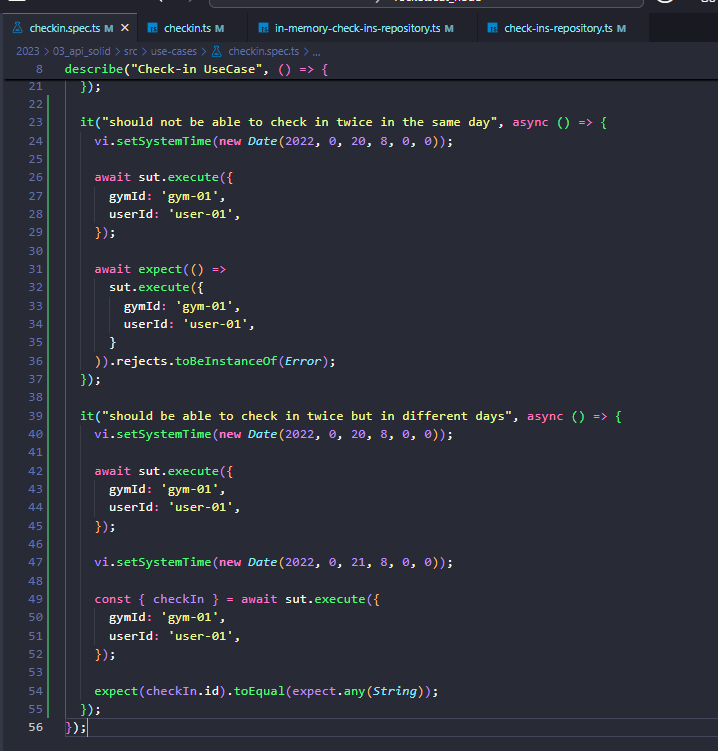


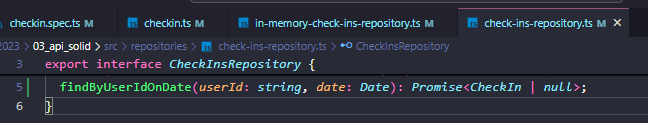


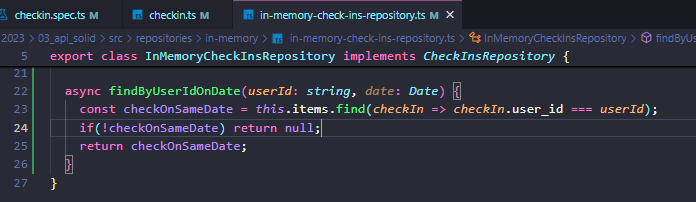


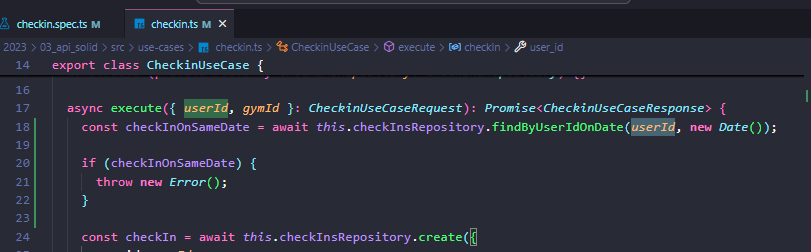


30 – TDD & Mocking



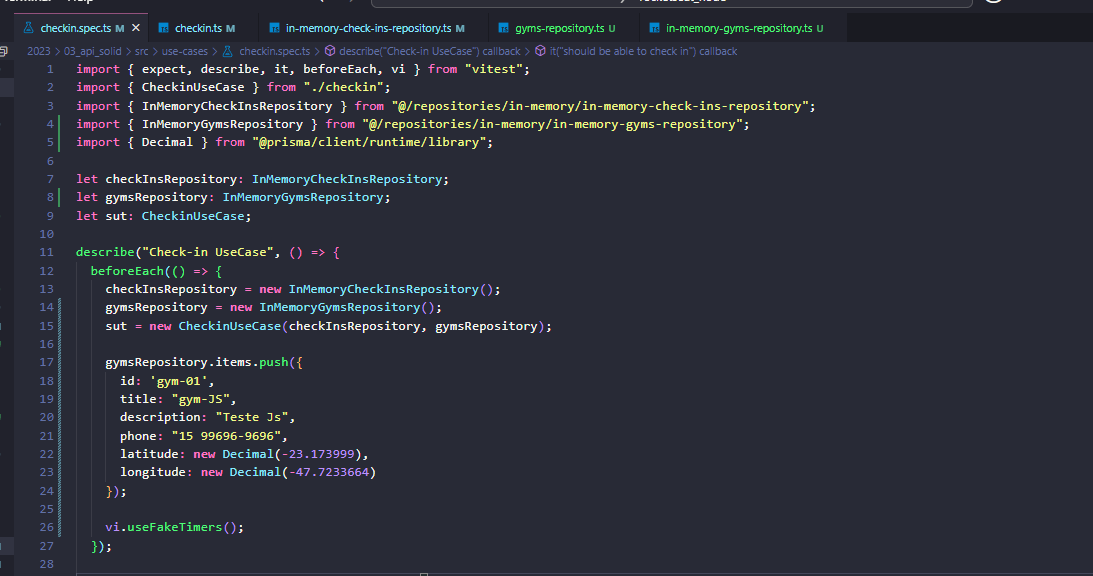


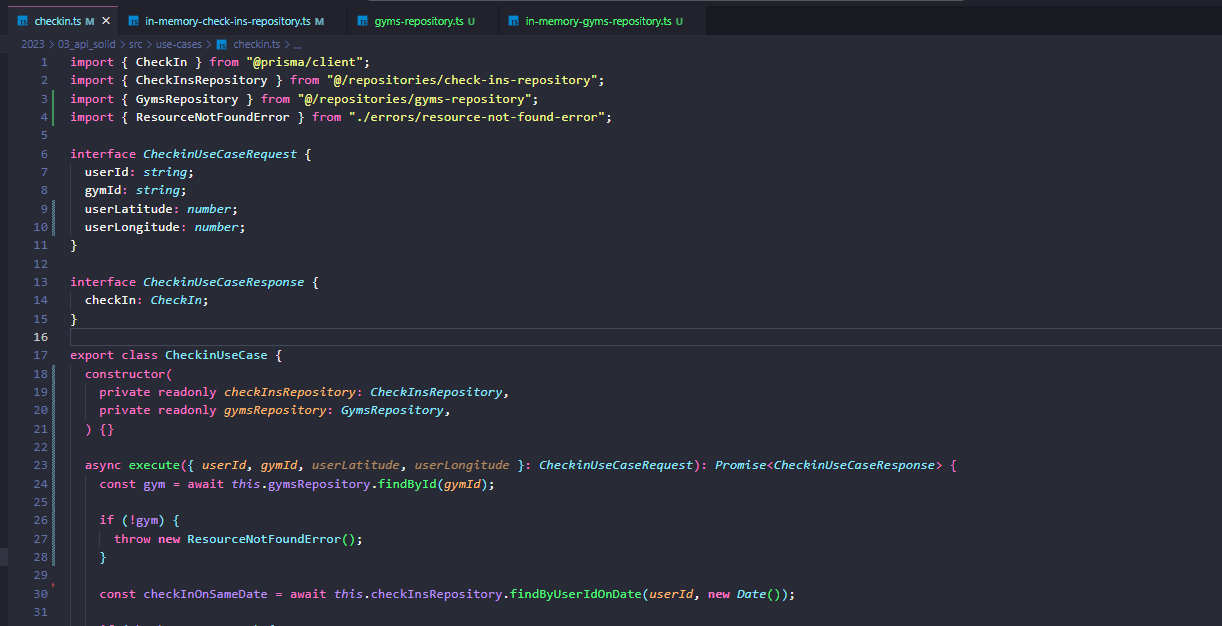


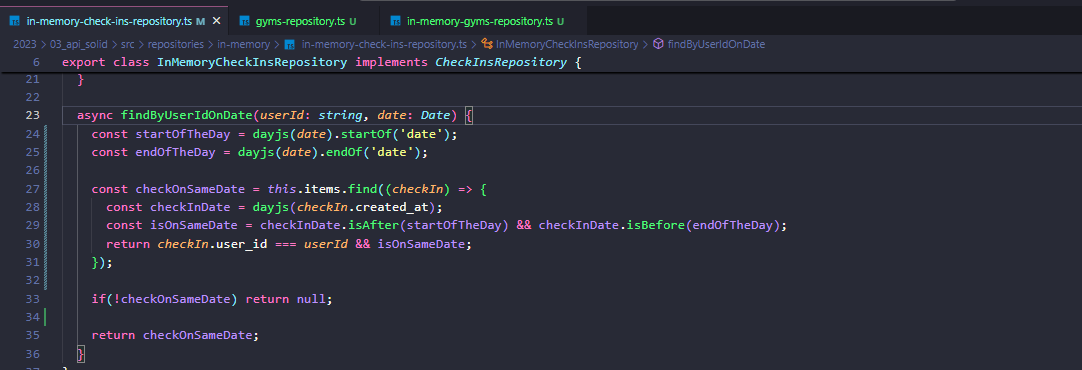


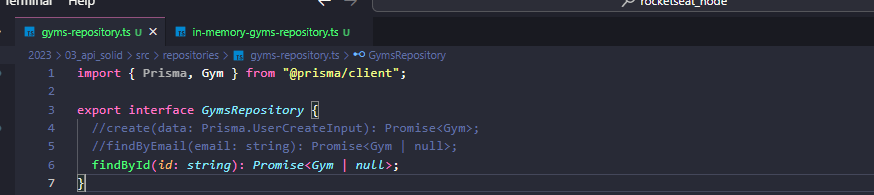
31 – Validando data do check-in

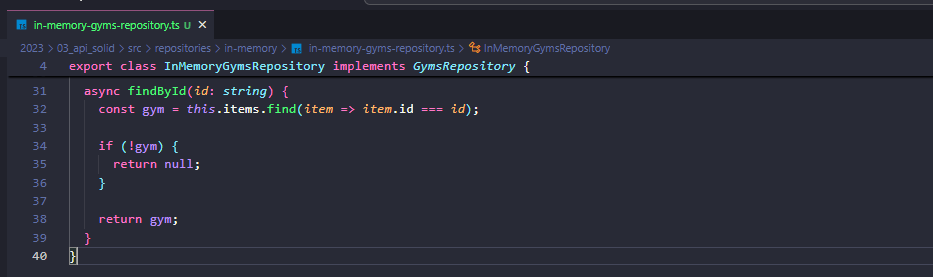
Instalar npm i dayjs











32 – Validando distância do ckeck-in

33 – Caso de uso de criação de academia

34 – Caso de uso de histórico

35 -

36 -

37 -

38 -

39 -

40 -