1. Läs kravspec, skapa modell och metodnamn

Först läste kravspec och såg nyckelord som “Ska använda Repository, ingen InMemory databas, seedad data”. Sedan tänkte jag vilka metoder/metodnamn som minst behövs (kolumn 2 i tabellen nedan). Skapade också modellen Quote enligt Guid, Text (döpte om från Quote), Category och Approved. Sedan hade jag en mall med “//arrange //act //assert” som jag copy pasteade flitigt. De initiellt skapade metodnamnen hamnade i Act, men ibland behövdes ju metoder i arrange också.

Förklaring + httpmetod	**Metod (del1)**	**API Routes/Endpoints (del2)**
leverera ett slumpmässigt citat	GetRandomAsync	/api/Quotes/random (GET)
leverera ett slumpmässigt citat inom en specifik kategori	GetRandomByCategoryAsync	/api/Quotes/random?category=motivation (GET)
ta emot ett nytt citat från en användare	AddAsync	/api/Quotes (POST)
lista alla citat som ännu inte godkänts	GetAllQuotesAsync(false)	/api/Quotes?approved=false (GET)
godkänna ett citat	ApproveQuote(guid)	/api/Quotes/{id}/approve (PUT)

2. Implementera metod

Körde med ARRANGE/ACT/ASSERT och implementerade metoderna för att enhetstesten ska lysa grönt.

3. API routes och integrationstest (del2)

Ett personligt mål var att försöka göra mer modulärt API likt andra företags webbAPIer, istället för att ha jättemånga unika api endpoints. Endpointsen i kolumn 3 i tabellen är de som samarbetar med Repositoryn.

4. Lärdomar och funderingar

- Inledningsvis kändes Red-Green-Refactor och ACT lite löjligt då IDE:n klagar “detta kompilerar inte!”, men nu efteråt ganska praktiskt att “skissa upp flow:et” på ett o samma ställe istället för att hoppa eg mellan olika dokument. TDD leder nog till mer stabila grejer överlag.
- Fortfarande lite svårt o veta vad som är ett bra test, eller hur många grejer man ska testa i en testmetod. Det kommer nog bli tydligare med mer erfarenhet
- Metodnamnen för enhetstesten är besvärliga att döpa men där hjälpte det att fråga AI om tips och förslag