Jag tycker generellt sett att Testdriven Development är en bra metod att utveckla sin kod.

Jag ser många olika fördelar, för att nämna några:

* Man måste tänka igenom vad koden ska leverera innan man börjar utveckla
* Man ser direkt när man bryter testet om man har implementerat en bugg
* Andra personer som kommer in senare i projektet har ett skyddsnät så att dom inte deployar kod som inte går igenom testerna

Däremot tycker jag att man måste ha mer än unittester för att få ihop helheten.

Har man inga integrationstester, smoketests osv så kommer man bara en liten bit på vägen med unittester.

Dessutom tycker jag att man ska ha sin UML, sina user stories och codedesign klar innan man börjar implementera sina test.

Det är svårt att hålla reda på vilka test man ska göra redan i en sån här liten applikation med bara 5-6 klasser. Då kan man tänka sig hur ett större program kan urarta om man skriver tester utan ens ett klassdiagram att hålla reda på.

Men jag förstår att man skulle bryta hela TDD tänket om man gör så, men personligen skulle en skiss på programmet underlätta innan man gör test.

Fast om man tänker sig att man gör en applikation med microservices så bör ju en microservice vara så pass liten att man kan köra på med TDD rakt upp och ner utan UML. 😊