1DV610 - Laboration 3 - App

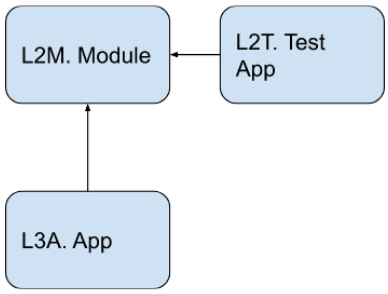
Syfte

- Skriva kod som bygger på modul-koden från laboration 2. Projekten är separerade så att app-koden i laboration 3 inte påverkar modulen.
- Skriva kod för en App med gränssnitt som uppfyller er egenskapade kravspecifikation. Denna kod skall bygga på det vi lär oss på föreläsningar och ur boken.
- Förbättra koden från laboration 2 enligt CC och Workshop 2
- Reflektera över och arbeta med CC kapitel 2-11

En app

I laborationen kommer ni använda er modul från laboration 2 för att skriva en app. Appen skall ha en eller flera **slutanvändare** som använder den genom ett interface. Interfacet kan vara webbapp, fristående applikation, mobilapplikation, eller en konsol. En viktig skillnad från modulen är att en app har ett syfte som uttrycks i krav. En användare får något behov uppfyllt av appen. Behovet beskrivs som vision och krav (se 1dv613).

Till slut så skall ni ha tre olika delar i ert projekt. De tre delarna skall hanteras som individuella projekt och beroendena mellan dem skall vara som följer.



1. Ert modul (L2M)
2. Era tester för er modul (L2T), har ett beroende till er modul (L2M) detta betyder att kod i biblioteket anropar modulens gränssnitt (ex metoder, api eller funktioner). Men modulen ska fungera även om all kod i testerna försvinner.
3. Er app (L3A) beror på er modul (L2M). Appen fungerar INTE utan modulen men modulen skall fungera utan appen.
4. Era tester(L2T) och er app(L3A) skall vara oberoende. Er app skall fungera utan testerna.
5. Er modul eller era tester får inte ha ett beroende TILL appen. Men appen skall ha ett beroende TILL modulen.

Regler

- Skriv **all** kod själv, skriv inte tillsammans med någon (sida vid sida-programmering).
- Kopiera inte kod från någon annan. Skriv inte av kod.
- Generera inte kod (Exempelvis med LLM)
- Ni skall använda er modul från laboration 2. Ni får skriva om modulen men modulen skall hållas som ett separat projekt och skall bibehålla kraven och testerna från laboration 2. Ja, ni får lägga till tester eller skriva om testerna (eller testappen).
- Skriv objektorienterad kod med klasser och metoder i klasser. Du kan ha kod utanför klasser men bara om den behövs för att starta upp koden (ex. nodejs “server.listen(port…”). Inga metoder i dina klasser får vara statiska mer än om det behövs för att starta upp koden eller om du behöver speciella namngivna konstruktorer (se CC). Försök hålla så mycket som möjligt private (se CC)
- Projekten dokumenteras på lämpligt sätt på git. Fundera över hur de olika målgrupperna (Slutanvändare, Apputvecklare, Modulanvändare, Modulutvecklare, Examinator) separat får sina informationsmål uppfyllda av er dokumentation. Fokus på effektiv dokumentation. Om ni inte tillför information och dokumentationen “känns tråkig” fundera på om ni gör ett värdefullt bidrag eller ej.

Validering och verifiering

Fundera ut vilka testfall som behövs för att täcka in de olika kraven för er App. Skapa en tabell med testfall som alla har beskrivande namn, indata samt förväntat utfall. Ni får testa automatiskt eller manuellt men enklast är ofta att testa appen manuellt via dess gränssnitt.

Minst ett test per krav (Se 1dv613 för instruktioner att skriva tester)

Kodkvalitetskrav för betyg A-E

Gå igenom all kod inklusive kod från laboration 2 och uppdatera enligt bokens clean code kapitel 2-11 och det vi diskuterat på föreläsningar och workshops. Skriv en kort (4-6 meningar) reflektion för varje kapitel om hur just det kapitlet har påverkat eller inte påverkat din kod. **Använd bokens termer**. Ge exempel med **läsbara** screenshots från er kod till varje reflektion. Dokumentera detta till mig i ett separat dokument reflection.md där jag är mottagaren.

Fokusera på tydlighet, variation, ärlighet och vad som är intressant. Exempelvis om du har icke självklara överväganden med olika kvalitetsregler som står i konflikt med varandra så är dessa extra intressanta.

Jag kommer även titta på och bedöma er kod. Den skall därför i största mån vara skriven för att kunna fortsätta utvecklas av andra programmerare.

Betyg sätts efter:

- 20%, funktionalitet, hur väl ni uppfyller kraven samt hur omfattande och komplex funktionaliteten är.
- 25%, projektens framställning, hur väl ni presenterar projekten via dess artefakter, ex README.md
- 10%, testning, hur väl ni testat kraven
- 20%, reflektion, hur väl ni kan diskutera bokens innehåll
- 25%, kodkvalitet, hur väl jag kan förstå er kod utifrån Clean Code och föreläsningar.