PA1489 Uppgiftsbeskrivningar

Mikael Svahnberg*

2024-04-17

1 Introduktion

Det här dokumentet beskriver de examinerande uppgifterna i kursen PA1489 Grundläggande Mjukvaruutveckling.

Följande uppgifter ingår:

Uppgift	Omfattning	Beskrivning
Inlämningsuppgift 1	2.5 hp	Samarbete och Konfigurationshantering
Inlämningsuppgift 2	2.5 hp	Implementation och Dokumentation
Inlämningsuppgift 3	2.5 hp	Testning och Debugning

Alla uppgifterna löper över hela kursen, och ni behöver arbeta parallellt med dem (de hjälper och stödjer varandra). *Läs igenom hela dokumentet först* och planera för hur ni skall kunna arbeta med alla tre inlämningsuppgifterna under hela projektets gång.

2 Övergripande Uppgift: BurgerOrderer

Du och dina kollegor har blivit kontaktad av ett företag som vill ha ett system för att beställa hamburgare. Det skall dels finnas en web-klient där man kan beställa och anpassa sina hamburgare, dels skall en andra klient utvecklas där köket kan se vilka hamburgare som skall tillagas.

Med glädje i hjärtat och en sång på läpparna accepterar ni uppdraget. Det här skall bli kul, och ni bryr er inte om att de bara kan betala i friterad småpotatis.

Det är då som beställaren kommer med ytterligare krav. De vill att ni skall samarbeta väl och med hjälp av moderna konfigurationsverktyg; de har hört talas om "Git" och vill att ni skall använda detta. De vill att de olika klienterna i projektet skall utvecklas som fristående enheter ("Ryktet säger att Containers är det senaste, och då menar vi inte den sorten som man lastar på båtar!"). Vidare vill de att projektet skall hålla hög kvalitet. De kommit fram till att "hög kvalitet" består av tre delar. Den första delen handlar om att allting skall vara väl dokumenterat. Den andra delen är att det skall finnas automatiserad testning av den viktigaste funktionaliteten i projektet. Den tredje delen är att ni som utvecklare visar kunskap om grundläggande färdigheter som t.ex.

*Mikael.Svahnberg@bth.se

debuggning och att ni ständigt reflekterar över vad de gör och ständigt strävar efter att bli bättre på det ni gör.

Nåja. Småpotatisen är i varje fall friterad. Ni bestämmer er för att dela upp beställarens förväntningar i tre komponenter, (1) Visa på förmåga att samarbeta med hjälp av konfigurationsverktyg, (2) implementation och dokumentation av projektet, samt (3) testning och visa förmåga att debugga. Under alla tre komponenter skall ni arbeta aktivt med att dokumentera vad ni gör i en ingenjörsdagbok, och när ni är färdiga skall ni sammanfatta och reflektera över var och en av de tre komponenterna.

Sagt och gjort; Upp, upp, och iväg!

3 Inlämningsuppgift 1: Samarbete och Konfigurationshantering

Er första uppgift blir att bestämma vilken utvecklingsmiljö ni vill använda och ställa i ordning ert konfigurationshanteringssystem. Beställaren föredrar om ni arbetar med Git, men ni får själva välja vilken server (t.ex. *Codeberg* eller *Github*) ni vill använda.

3.1 Kom Igång

1. Forma utvecklingsteam
   - Planera hur ni vill arbeta
2. Bestäm utvecklingsmiljö
   - Det är inte ett krav att alla använder samma utvecklingsmiljö, men det kan underlätta.
3. Bestäm vilken git-server ni vill använda
   - Se till att alla har konton på servern.
4. Skapa projektet. Se till att ge det ett bra namn och beskriv det ordentligt.
5. Under katalogen `Planering` kan ni börja med att lista vilka varutyper (menyer, hamburgare, tillbehör, dricka, osv.) som skall finnas.
   - Tag chansen att skapa din första commit här.
   - Samarbeta t.ex. genom att var och en tar ansvar för någon typ av vara (hamburgare, tillbehör, ...) som skall säljas.
   - skriv i en egen textfil för den varutypen.
   - Byt sedan fil och fortsätt på en fil som någon annan påbörjat.
   - Upprepa tills alla har funderat på alla varutyper.
6. Under katalogen `Reflektioner` skapar ni varsin textfil (med ert namn i filnamnet) där ni dokumenterar vad ni lärt er.
   - Det här är er *Ingenjörsdagbok*
   - Vilka git-kommandon använder ni? Vad gör de?

- Vilka svårigheter stöter ni på? Hur löser ni dem?
- Finns det något som ni inte vet hur man gör? Hur kan ni ta reda på det?

Med den här grunden är ni redo att gå vidare till de övriga uppgifterna, men notera att ni måste fortsätta arbeta med punkterna *Under Arbetets Gång* och *Sammanfatta och Reflektera* samtidigt som ni arbetar med inlämningsuppgift 2 och 3.

3.2 Under Arbetets Gång

Under tiden som ni arbetar med de övriga uppgifterna gäller följande:

1. Under hela utvecklingsarbetet skall ni *regelbundet* uppdatera med *commits* när ni har implementerat någonting.
   - Varje gruppmedlem måste aktivt och regelbundet bidra med relevanta commits.
   - Tänk på att commit-meddelanden skall tydligt beskriva vad commiten innehåller.
   - Det ger extrapoäng att organisera arbetet så att utvecklingen sker i separata förgreningar (*branches*).
     - Glöm inte att dokumentera er plan så att ni kan hänvisa till den i era reflektioner.
   - Det är en bonus men påverkar inte betyget om ni hantera konflikter där olika versioner av en fil behöver kombineras.
2. Under hela utvecklingsarbetet skall ni *uppdatera er ingenjörsdagbok* med vad ni lär er om konfigurationshantering.
   - Försök få in en vana att ni ofta tänker igenom vad ni gjort, vad ni lärt er, och vad ni måste ta reda på.
   - Skriv hellre litegrand varje dag än jättemycket en gång i veckan.
   - Glöm inte att committa vad ni skrivit så att det syns i projektloggen.
   - Det ger extrapoäng om ni läser och sammanfattar externa källor om konfigurationshantering.

3.3 Sammanfatta och Reflektera

Skriv gemensamt en sammanfattande text i katalogen `Reflektioner`. Följande skall ingå:

1. Namn på alla i teamet.
2. Länk till projektets sida på git-servern.
3. Kort sammanfattning om vad konfigurationshantering är och varför det används.
4. Kort sammanfattning om det vanligaste arbetsflödet med git, inklusive de git-kommandon som används.

5. Era erfarenheter om att arbeta med konfigurationshantering.

    - Vad gick bra?
    - Vad gick mindre bra?
    - Hur löste ni svårigheterna? Hade ni kunnat göra annorlunda?
    - Vad lyckades ni inte lösa? Varför inte?

Kopiera texten och skicka in den på Canvas för bedömning. Se vidare instruktioner på Canvas om inlämningsdatum mm.

3.4 Bedömning

Följande ingår i bedömningen, och vägs samman till ett betyg på inlämningsuppgiften:

Regelbundna bidrag (Bedöms enskilt) bidrar aktivt med commits till minst en del av projektet.

Dagboksreflektioner (Bedöms enskilt) har regelbundna och relevanta anteckningar och reflektioner om att arbeta med konfigurationshanteringsverktyg.

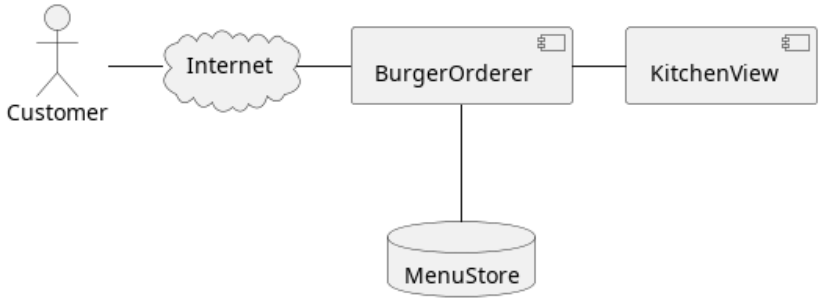
Välbeskrivna commits Så gott som alla commits har commit-meddelanden som väl beskriver bidraget.

Förgreningar Gruppen arbetar aktivt med separata förgreningar (branches) i utvecklingen.

Reflekterande Praktik Arbetet är väl sammanfattat med reflektioner om förbättringsmöjligheter.

4 Inlämningsuppgift 2: Implementation och Dokumentation

Beställaren har gett er en övergripande arkitektur som ni skall förhålla er till:



BurgerOrderer Det huvudsakliga webgränssnittet.

- Presenterar de olika varutyperna

- Kunden kan välja vad de vill ha med i sin beställning
- Kunden kan anpassa sin beställning (t.ex. ta bort "lök" från sin "Metric Ton Bacon Burger")
- Hämtar information om de olika varutyperna från databasen `MenuStore`
- När beställningen är klar skickas den via ett REST-anrop till `KitchenView`
- *Beställaren hälsar* att ni inte behöver göra det snyggt. Funktionalitet är i nuläget viktigare än Form.

MenuStore En databas som innehåller information om varje typ av vara.

- Information om de olika varutyperna och hur man kan anpassa dem kan skötas via ett separat gränssnitt såsom *adminer*.
- *Beställaren hälsar* att ni själva får välja om ni vill använda en relationsdatabas eller en NoSQL-databas.

KitchenView Tar emot beställningar från BurgerOrderer och visar dem för kökspersonalen.

- När en beställning är mottagen via ett REST-API skall den skrivas ut på skärmen.
- *Beställaren hälsar* att det är ok med en enkel textbaserad utskrift.
- *Beställaren hälsar* också att det är ok om man inte kan interagera mer med beställningarna (t.ex. behöver man inte kunna markera att en beställning är färdig för leverans).

Beställaren önskar en containerbaserad platform, med separata containers för `BurgerOrderer`, `KitchenView`, och `MenuStore`. Ni får själva välja programspråk.

4.1 Kom Igång

1. Under katalogen `Containers`, skapa en katalog för `BurgerOrderer` och en för `KitchenView`
2. Skapa en `Dockerfile` för respektive container.
3. Skapa en `package.json` (eller vad som krävs för ert val av programspråk) och fyll i relevant information.
4. Dokumentera och/eller automatisera hur man bygger containers respektive hur man kör projektet.

Planera arbetet:

1. Hur ser er `MenuStore`-databas ut? Vilken information måste finnas med för varje varutyp?
2. Hur ser er `KitchenView` ut?
3. Hur ser er `BurgerOrderer` ut? Ni har lämpligen separata moduler för:
   - Webgränssnittet och API-ändpunktera

- Kontakt med `MenuStore`
- Kontakt med `KitchenView`
- Formattera presentationen av olika varutyper
- Hantera när kunden beställer olika varutyper så att de kan anpassa beställningen
- Sammanfattning av det som redan är tillagt i beställningen (med möjlighet att ta bort delar av beställningen)
- Tänk på att flera personer vill kunna beställa samtidigt, så titta på t.ex. `cookie-session` för att lagra data under tiden beställningen görs.

4. Vem skall vara ansvarig för vilken/vilka delar av projektet?
5. När skall de olika delarna vara klara?

4.2 Under Arbetets Gång

1. *Håll koll på er plan.*
   - Det är ok att avvika från planen om ni ser att det behövs, men det skall vara ett *medvetet* beslut.
   - Ju tidigare ni ser att ni inte kan hålla er plan, desto mer tid har ni att göra något åt det.
2. *Dokumentera* ert arbete.
   - Sammanfatta vad varje modul gör.
   - Beskriv med kod-dokumentation vad varje metod gör.
3. *Uppdatera regelbundet* med commits när ni har implementerat någonting.
4. *Uppdatera er ingenjörsdagbok* med vad ni lär er om implementering och att utveckla i containers.
   - Försök få in en vana att ni ofta tänker igenom vad ni gjort, vad ni lärt er, och vad ni måste ta reda på.
   - Skriv hellre litegrand varje dag än jättemycket en gång i veckan.
   - Glöm inte att committa vad ni skrivit så att det syns i projektloggen.

4.3 Sammanfatta och Reflektera

Skriv gemensamt en sammanfattande text i katalogen `Reflektioner`. Följande skall ingå:

1. Namn på alla i teamet.
2. Länk till projektets sida på git-servern.
3. Kort sammanfattning om vad ni implementerat. Beskriv med 5–10 meningar vad ni gjort och hur ni tänkt för

- Projektet i stort
- Varje container
- Varje modul

4. Era erfarenheter om hur projektet gick att genomföra.
   - Vad gick bra?
   - Vad gick mindre bra?
   - Hur löste ni svårigheterna? Hade ni kunnat göra annorlunda?
   - Vad lyckades ni inte lösa? Varför inte?
5. Era erfarenheter om att arbeta med containers.
   - Vad gick bra?
   - Vad gick mindre bra?
   - Hur löste ni svårigheterna? Hade ni kunnat göra annorlunda?
   - Vad lyckades ni inte lösa? Varför inte?

Kopiera texten och skicka in den på Canvas för bedömning. Se vidare instruktioner på Canvas om inlämningsdatum mm.

4.4 Bedömning

Följande ingår i bedömningen, och vägs samman till ett betyg på inlämningsuppgiften:

Dokumenterad Kod Alla containers är dokumenterade. Så gott som alla metoder är dokumenterade.

Dokumenterad Uppstart Det är väl dokumenterat eller automatiserat hur man startar upp hela projektet.

Implementerad Funktionalitet Följande saker är implementerade (I ökad svårighetsgrad):

1. *Visa Varutyper* Kunder kan se olika varutyper som hämtas från `MenuStore` databasen
2. *Sökbar Databas* `MenuStore` innehåller information om de olika varutyperna och används av `BurgerOrderer`
3. *Beställa varor* Kunder kan beställa varor, och de skickas till i `KitchenView`
4. *Visa beställningar* `KitchenView` tar emot beställningarna och skriver ut dem.
5. *Justera Beställning* Kunder kan plocka bort varor från sin beställning innan de skickas till `KitchenView`
6. *Anpassa beställda varor* Kunder kan anpassa sina beställda varor innan de skickas till `KitchenView`

Reflekterande Praktik Arbetet är väl sammanfattat med reflektioner om förbättringsmöjligheter. Både gemensamt och i individuella ingenjörsdagböcker.

5 Inlämningsuppgift 3: Testning och Debugning

Slutligen begär beställaren att projektet skall hålla hög kvalitet. I den här inlämningsuppgiften fokuserar ni främst på två kriterier för detta:

- Det skall finnas automatiserade enhetstester för den viktigaste funktionaliteten, och
- Samtliga utvecklare vet hur man hanterar en debugger.

5.1 Kom Igång

1. Planera vilka delar som skall testas.
   - Vilka moduler? Vilka metoder? Vilka API-ändpunkter?
   - Hur skall de testas? Hur anropas de? Vad får ni för svar?
   - Vilka teknologier (t.ex. testramverk) behöver ni?
   - Hur ofta skall testerna köras? Vad händer om testerna misslyckas?
2. Se till att allt som behöver installeras (t.ex. testramverk) finns tillgängligt i var och en av de containers ni skall utveckla.
3. Se till att det går att köra de automatiserade testerna. Dokumentera/Automatisera hur man går tillväga.

5.2 Under Arbetets Gång

1. *Följ er testplan.*
   - Genomför testerna när ni har planerat dem, och hantera resultaten som planerat.
2. *Dokumentera* ert arbete.
   - Sammanfatta vad som skall testas, hur, och när.
   - Beskriv era tester (Ni behöver inte beskriva varje enskilt test, det räcker med övergripande sammanfattningar).
   - Dokumentera resultaten från era tester.
3. *Uppdatera regelbundet* med commits när ni har implementerat någonting. Även testkod skall konfigurationshanteras.
4. *Uppdatera er ingenjörsdagbok* med vad ni lär er om automatiserad testning.
   - Försök få in en vana att ni ofta tänker igenom vad ni gjort, vad ni lärt er, och vad ni måste ta reda på.
   - Skriv hellre litegrand varje dag än jättemycket en gång i veckan.
   - Glöm inte att committa vad ni skrivit så att det syns i projektloggen.

5.3 Genomför och Dokumentera en Debug-session

Någon gång under projektets gång skall var och en av er genomföra en debug-session och dokumentera denna i er ingenjörsdagbok.

1. Välj någon funktionalitet, till exempel *beställ en "Dripping With Lard Heartstopper"-meny*
2. Vilka breakpoints sätter du för att starta debug-sessionen? Var hittar du filen du skall sätta dem i?
3. Hur fortsätter du? Hur använder du knapparna "Continue", "Step over", "Step into", och "Step out"?
4. Bevaka någon variabel.
   - Hur gör du det?
   - Vad har den för värde?
   - Kan du få reda på när värdet på variabeln ändras? Hur?
5. Prova lite olika "vägar" genom funktionaliteten, till exempel beställ något annat, avbryt halvägs igenom, osv.
   - Hur påverkar detta vilken kod som körs?
   - Hur påverkat detta dina bevakade variabler?

Varje steg skall dokumenteras. Upprepa varje steg tillräckligt många gånger så att du känner dig säker på hur det fungerar.

Avsluta med att sammanfatta och reflektera, till exempel om:

- Vad gick bra? Vad gick mindre bra?
- Vad var lätt? Vad var svårt?
- Kan debugging bli ett användbart verktyg för dig? Varför? Varför inte?

1. Tips Att debugga en node.js-applikation som körs inuti en container är inte riktigt samma sak som att debugga något som man utvecklar lokalt. En guide att följa för vscode/vscodium är: `https://medium.com/fl0-engineering/the-best-way-to-debug-a-node-js-app-running-in-a-docker-container-99241afb4781`

   Lämplig sökterm för att hitta hur man skall göra för ditt val av programspråk är t.ex. *"debug <programspråk> in container"* , möjligen kan man också behöva lägga till *<utvecklingsmiljö>*.

5.4 Sammanfatta och Reflektera

Skriv gemensamt en sammanfattande text i katalogen `Reflektioner`. Följande skall ingå:

1. Namn på alla i teamet.
2. Länk till projektets sida på git-servern.
3. Kort sammanfattning om vilken funktionalitet ni har testat.

4. Kort sammanfattning om hur ni har genomfört testerna.
5. Utskrift från er senaste testsession, så att man kan se:
   - Hur många tester ni har skrivit
   - Vad de testar
   - Hur många tester som lyckas respektive misslyckas
6. Era erfarenheter om att skriva automatiserade enhetstester.
   - Vad gick bra?
   - Vad gick mindre bra?
   - Hur löste ni svårigheterna? Hade ni kunnat göra annorlunda?
   - Vad lyckades ni inte lösa? Varför inte?
7. Länk till dokumentationen från era respektive debug-sessioner i era individuella ingenjörsdagböcker.

Kopiera texten och skicka in den på Canvas för bedömning. Se vidare instruktioner på Canvas om inlämningsdatum mm.

5.5 Bedömning

Följande ingår i bedömningen, och vägs samman till ett betyg på inlämningsuppgiften:

Testplan Det är dokumenterat vad som skall testas, hur det skall testas, hur ofta, och vad som skall göras om något test misslyckas.

Funktionalitet Testad Det finns enhetstester som testar någon del av någon funktionalitet i systemet.

Fungerande Enhetstester Enhetstesterna går framgångsrikt att köra genom att följa beskrivning från testplanen-

Följer Testplanen Den testade funktionaliteten följer testplanen.

Dokumenterad Erfarenhet av Debugging (bedöms enskilt) dokumentation och reflektioner från en genomförd debug-session.

Reflekterande Praktik Arbetet är väl sammanfattat med reflektioner om förbättringsmöjligheter. Både gemensamt och i individuella ingenjörsdagböcker.

6 Avslutning och Sammanfattning

"Några frågor?"

Förväntansfullt ser ni på er beställare och hoppas tyst att de inte skall ha några frågor att ställa. Ni har precis avslutat en fantastisk presentation som beskriver i detalj hur ert BurgerOrderer-system fungerar och använder de Allra Senaste Teknikerna™ med `git`, containers, och enhetstestning. Ni har visat hur alla i projektet minsann vet hur man använder sin utvecklingsmiljö, och

att ni använder er ingenjörsdagbok på ett naturligt sätt för att ständigt kunna reflektera över ert arbete och se hur ni kan förbättra er.

Ni hoppas att kunden inte skall ställa alltför många frågor om hur ni kunde missa något så grundläggande i era tester och det där lilla missödet med flera samtidiga kunder, men ni tror samtidigt att ni lyckades avleda uppmärksamheten från dessa – trots allt – mindre brister i ert projekt. Om man bortser från bristerna så var det ju faktiskt nästan helt perfekt.

Beställaren lutar sig framåt och tittar ner i sin anteckningsbok. De harklar sig och undrar:

"Är ni säkra på att ni har pushat alla commits och laddat upp alla tre rapporterna på Canvas?"

... och självklart har ni ju det.