ITERATION 3

PUBLICERING AV LAGRAT DOKUMENT I SYSTEMET

SPECIFICERAD PLANERING

KRAV	PLANERAD TID	VERKLIG TID
Analysera användningsfall	15 Minuter	15 Minuter
Specificera	30 Minuter	30 Minuter

DESIGN / IMPLEMENTATION	PLANERAD TID	VERKLIG TID
Identifiera klasser	30 Minuter	15 Minuter
Specificera klasser	1 Timme	45 Minuter
Implementation	1 Timma 15 Minuter	1 Timme

TEST	PLANERAD TID	VERKLIG TID
Test design	45 Minuter	45 Minuter
Test specifikation	45 Minuter	45 Minuter
Test implementation	1 Timma 15 Minuter	1 Timma 30 Minuter
Test exekvering	30 Minuter	45 Minuter

REFLEKTION	PLANERAD TID	VERKLIG TID
Reflektera	45 Minuter	1 Timme

TOTAL PLANERAD TID: 8 TIMMAR
TOTAL VERKLIG TID: 7 TIMMAR 30 MINUTER

DESIGN OCH IMPLEMENTATION

FLÖDESSCHEMA

PRE-Villkor

1. Användaren skall ha ett registrerat användarkonto i systemet.
2. Det måste finnas åtminstone ett sedan tidigare lagrat dokument.

POST-Villkor

1. Dokumentet är publicerat.

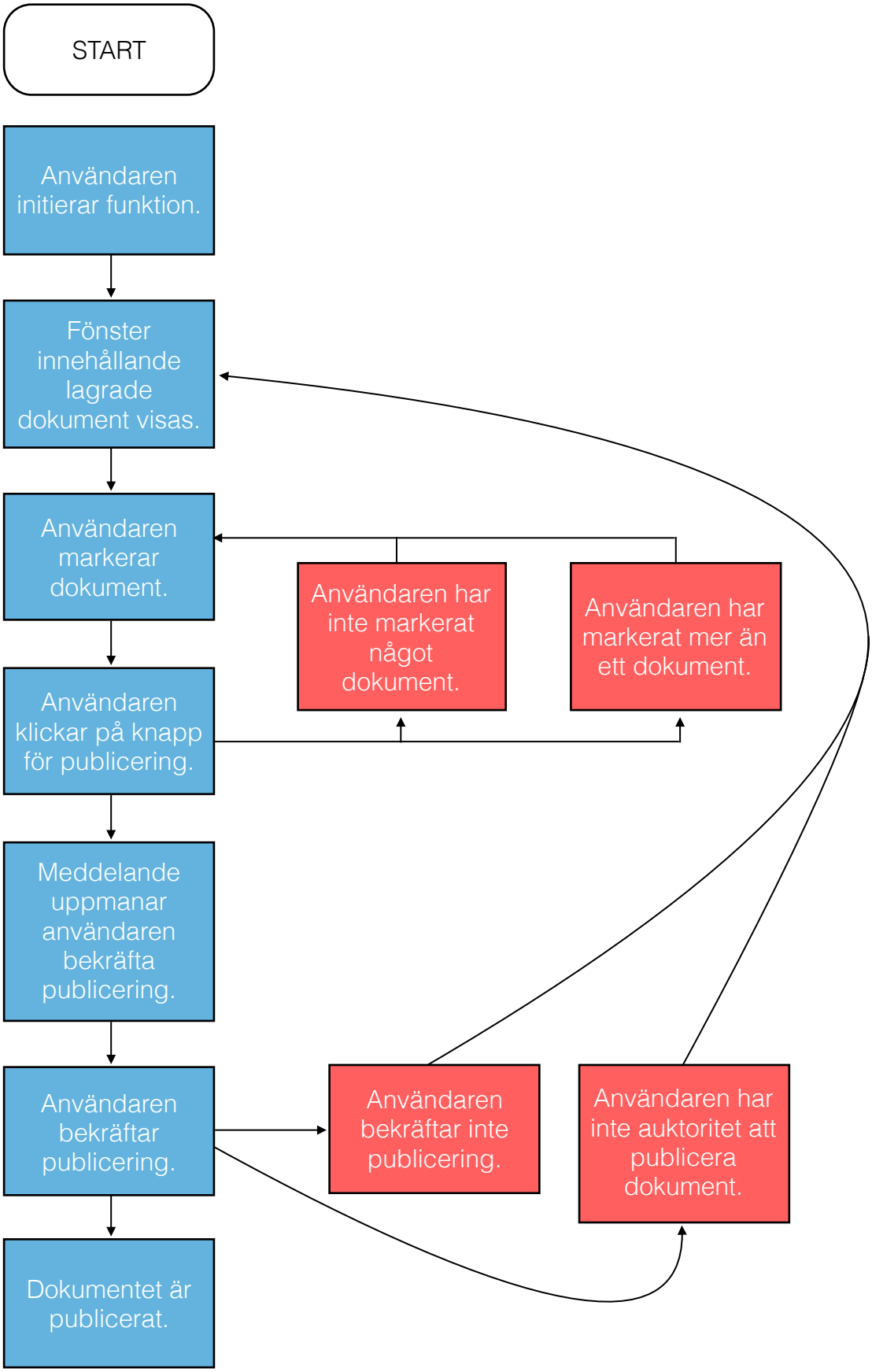
PRIMÄRT FLÖDE

1. Användaren navigerar sig till och initierar funktion för publicering.
2. Fönster innehållande lagrade dokument visas i listvy för användaren.
3. Användaren markerar det dokument denne vill publicera.
4. Användaren klickar på knapp för publicering.
5. Meddelande som uppmanar användaren bekräfta publicering visas.
6. Användaren bekräftar publicering.
7. Dokumentet är publicerat.

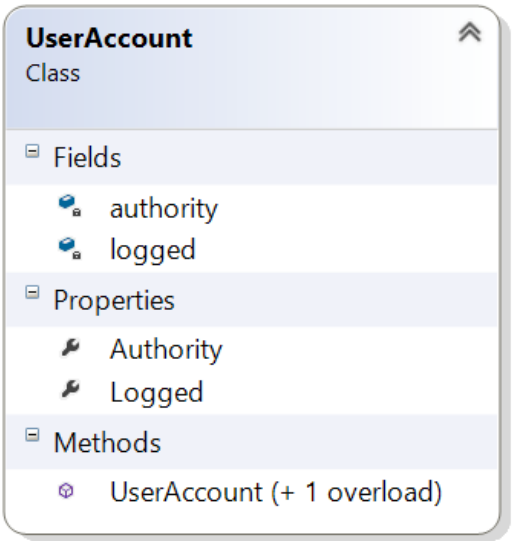
SEKUNDÄRT FLÖDE

4A. Användaren har inte markerat något dokument.
6B. Användaren har markerat mer än ett dokument.
6A. Användaren bekräftar inte publicering.
6B. Användaren har inte auktoritet att publicera dokument.

FLÖDESDIAGRAM



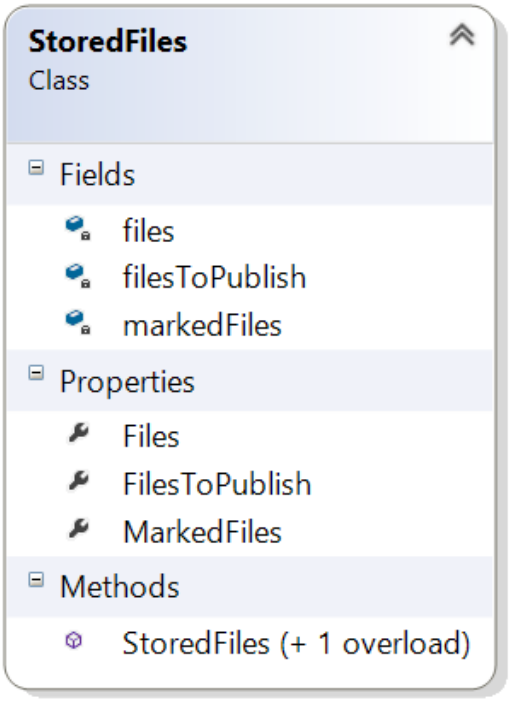
KLASSER MED BESKRIVNINGAR



UserAccount

Klassens uppgift är att validera huruvida användaren är inloggad i systemet och således tillåten att initiera funktionen. Beroende på användarens status skrivs lämpliga meddelanden ut.

Klassens andra uppgift är att validera huruvida användaren har auktoritet att publicera ett eller flera dokument. Vid händelse av att så inte är fallet kastas ett undantag med passande felmeddelande.



StoredFiles

Klassens uppgift är att validera huruvida det finns dokument i systemet att publicera samt antalet av dessa. Vid händelse av att inga dokument finns lagrade kastas ett undantag efter initiering av funktion med beskrivande felmeddelande.

Klassen skall även kontrollera att användaren markerat det dokument som skall publiceras när denne klickar på knapp för publicering. Vid händelse av att användaren inte markerat någon fil alternativt markerat mer än ett dokument kastas undantag med beskrivande felmeddelande beroende på situation.

Fältet *files* med dess tillhörande egenskap *Files* nyttjas inte i någon av testfallen till denna iteration.

ENHETSTESTNING

TESTSVIT

Enhetstest för metoderna i klassen UserAccount

Testfall	Vad skall testas	Testdata	Förväntat resultat	Resultat
Lyckad initiering av funktion.	Kontrollera om användaren är inloggad.	logged = true	Initiering lyckas.	Initiering lyckas.
Misslyckad initiering av funktion.	Kontrollera om användaren är inloggad.	logged = false	Initiering misslyckas.	Initiering misslyckas.
Lyckad validering av auktoritet att publicera.	Kontrollera om användaren har auktoritet att publicera dokument.	authority = true	Validering lyckas. Publicering kan fortgå.	Validering lyckas. Publicering kan fortgå.
Misslyckad validering av auktoritet att publicera.	Kontrollera om användaren har auktoritet att publicera dokument.	authority = false	Validering misslyckas. Publicering misslyckas.	Validering misslyckas. Publicering misslyckas. Beskrivande felmeddelande kastas.

Enhetstest för metoderna i klassen StoredFiles

Testfall	Vad skall testas	Testdata	Förväntat resultat	Resultat
Lyckad validering av antal lagrade dokument.	Kontrollera att det finns åtminstone ett lagrat dokument.	filesToPublish = 5	Validering lyckas. Eventuell publicering kan fortgå.	Validering lyckas. Eventuell publicering kan fortgå.
Misslyckad validering av antal lagrade dokument.	Kontrollera att det finns åtminstone ett lagrat dokument.	filesToPublish = 0	Validering misslyckas. Eventuell publicering kan ej fortgå. Felmeddelande visas.	Validering misslyckas. Eventuell publicering kan ej fortgå. Felmeddelande kastas.
Lyckad validering av antal markerade dokument.	Kontrollera att användaren markerat ett dokument för publicering.	markedFiles = 1	Validering lyckas. Endast ett dokument är markerat. Användaren kan publicera dokument.	Validering lyckas. Endast ett dokument är markerat. Användaren kan publicera dokument. Antal markerade dokument skrivs ut.

Misslyckad validering av antal markerade dokument #1.	Kontrollera att användaren markerat ett dokument för publicering.	markedFiles = 0	Validering misslyckas. Inga dokument är markerade. Publicering kan inte fortgå.	Validering misslyckas. Inga dokument är markerade. Publicering kan inte fortgå. Beskrivande felmeddelande kastas.
Misslyckad validering av antal markerade dokument #2.	Kontrollera att användaren markerat ett dokument för publicering.	markedFiles = 3	Validering misslyckas. För många dokument är markerade. Endast ett tillåts.	Validering misslyckas. För många dokument är markerade. Endast ett tillåts. Felmeddelande kastas.

MOTIVERING

Testsviten är utformad på ett tydligt och överskådligt vis, som vid eventuell utökning kring kriterierna rörande publicering möjliggör för vidare påfyllning av ytterligare enhetstester och testfall. Testsviten innehåller två enhetstester med tillhörande testfall vilka verifierar och dokumenterar samtliga fält, egenskaper och metoder av mitt tillämpade användningsfall. Testsvitens samtliga testfall består av fem kolumner som innehåller beskrivande och förklarande information kring vad som skall utföras i varje enskilt test. Planerad testdata presenteras samt förväntade och slutgiltiga resultat. De dokumenterade testfallen anser jag kontrollerar samtliga tänkbara utfall av de fält, egenskaper och metoder som finns med i denna avslutande iterationens användningsfall.

IMPLEMENTERA TESTSVITEN OCH KÖR

DOKUMENTATION AV DE UTFÖRDA TESTERNA

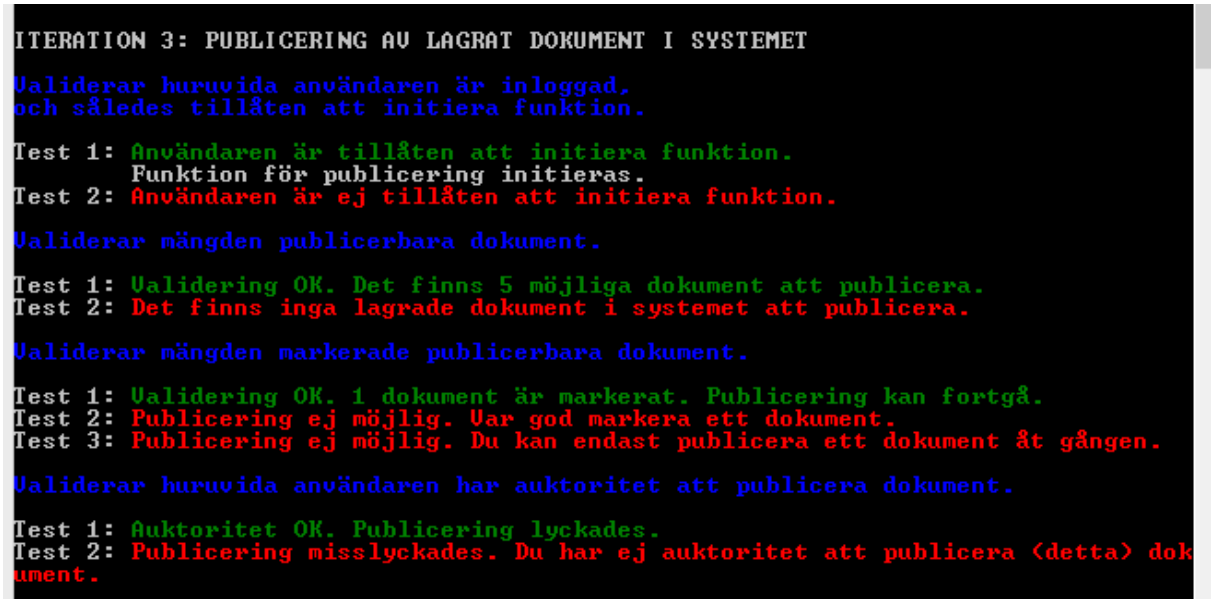
Bilden nedan presenterar mina kompilerade enhetstester med tillhörande testfall. Denna tredje och avslutande iteration testar i huvudsak systemets lagrade dokument samt huruvida en användare har auktoritet att publicera ett eller flera av dessa. Varje specifikt områdestest utgörs av två till tre tester där det ena resulterar i ett godkännande, och det andra (och eventuellt tredje) resulterar i ett ej godkänt utfall.

- Det inledande testet validerar huruvida användaren är inloggad eller ej, och därmed bestämmer om användaren är tillåten att initiera funktionen. I det fall användaren är inloggad kan initiering av funktion för publicering fortlöpa utan problem.
- Det andra testfallet kontrollerar mängden lagrade dokument i systemet som finns till förfogande att eventuellt publicera. Av uppenbara orsaker måste det åtminstone finnas ett tillgängligt dokument för att en publicering skall kunna ske. Skulle det inte finnas några lagrade filer visas ett meddelande som förklarar detta.
- Det tredje testet kontrollerar huruvida användaren markerat ett dokument när denne klickar på knapp för publicering. För att en publicering skall kunna genomföras krävs det att användaren har markerat ett (och endast ETT) dokument. Vid händelse av att inget alternativt fler en ett dokument markerats visas ett felmeddelande som förklarar situationen för användaren.
- Det fjärde och avslutande testfallet för denna tredje iteration validerar huruvida användaren som försöker publicera ett eller flera dokument har auktoritet att göra detta. Alternativt har auktoritet för en publicering av dokumentet i fråga. Vid händelse av att en användare inte har auktoritet skrivs detta ut och publiceringen avbryts.

ANALYSERA OCH FÖRESLÅ FÖRBÄTTRINGAR

Överlag är jag nöjd med de olika testfallen för denna iteration. Möjligtvis kan mina testfall tyckas vara få till antalet, men de testar och kontrollerar det mest väsentligaste stegen och områdena i en funktion likt denna vilket jag värdesätter högt. Återigen kan min upprepning av det inledande testfallet (rörande huruvida en användare är inloggad eller ej) kritiseras, men eftersom detta är något som alltid skall kontrolleras oavsett initiering av funktion valde jag att innefatta det även här. Utöver tidigare nämnd kritik är jag nöjd med testfallen i sin helhet.

BILD TAGEN PÅ KOMPILATOR FÖR TESTFALLEN AV DEN TREDJE ITERATIONEN



REFLEKTION

I denna tredje och avslutande iteration har jag vidareutvecklat klasserna *StoredFiles* och *UserAccount* från mina tidigare iterationer. Dessa utvecklingar var nödvändiga för att kunna utföra de tester jag planerat för iterationen. I klassen *UserAccount* införde jag en metod för att kontrollera användarens auktoritet vid publicering av dokument. Detta var mitt avslutande test som ett sista kontrollerande steg innan publicering. Klassen *StoredFiles* tillförde ytterligare tester för att även kontrollera mängden markerade dokument som användaren vill publicera. I detta fall tillåts endast en fil markerad. Arbetsprocessen har överlag fortlöpt utan problem. Jag märkte dock relativt tidigt in i programmeringen att jag skulle bli klar väl tidigt, därför valde jag att implementera fler varierade tester för att utvidga min totala iteration. Även i detta användningsfall har jag i stora drag återanvänt mitt sedan tidigare dokumenterade flödesschema samt flödesdiagram. Grundstrukturen står i princip oförändrad, dock tillförde jag ett par extra implementationer för att fylla min planerade tid. Med dessa ändringar i åtanke är jag ändå nöjd med hur iterationen avslutades.