CDIO-1

Gruppe 11

14/3-2020

02324 - Videregående Programmering

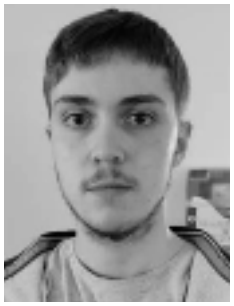
Mikkel Danielsen
s183913

Frederik Koefoed
s195463

Muhammad Talha Butt
s195475

Volkan Isik
s180103

Lasse Strunge
s195486



Mark Rune Mortensen
s174881

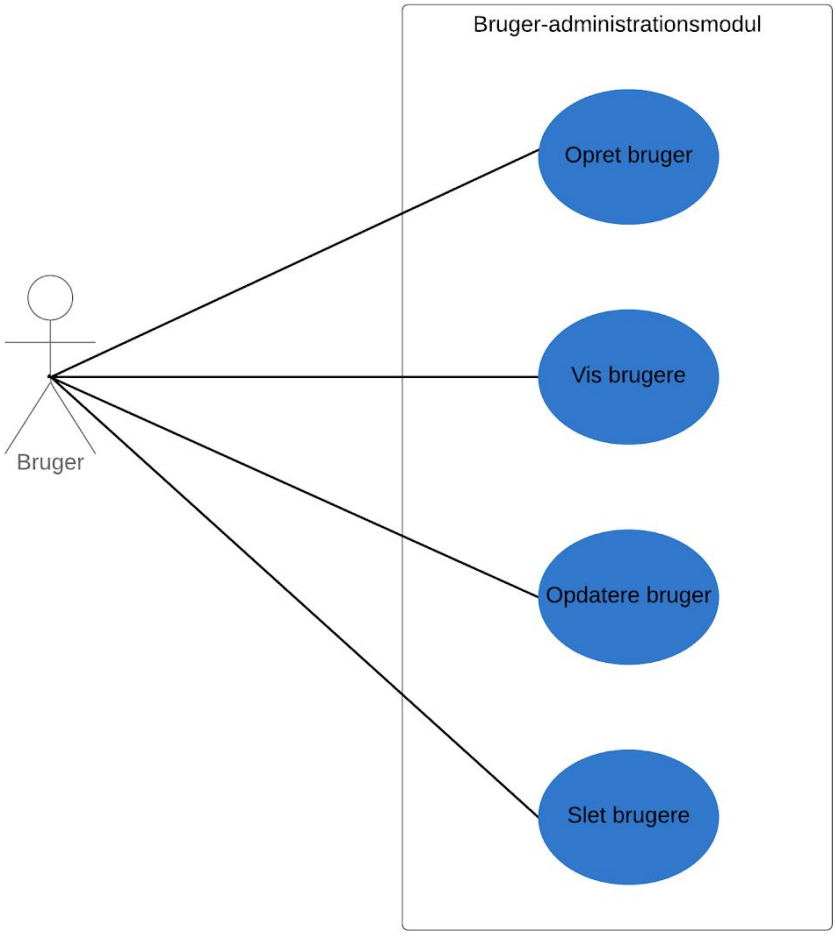
Indholdsfortegnelse

Kravliste

K1 - Programmet skal styres af en TUI (Tekstbaseret User Interface)
K2 - Det skal være muligt at oprette en bruger
K3 - Når man opretter en bruger skal man kunne angive deres navn, initialer, cpr-nummer samt deres roller
K4 - Man skal kunne se data på en bruger, hvis man har brugeren ID
K5 - Brugerens ID skal være mellem 11 og 99
K6 - Man skal kunne ændre i brugerens oplysninger (Navn, initialer, cpr-nummer, password)
K7 - Man skal kunne slette brugere
K8 - Der er 4 roller (Admin, Pharmacist, Foreman, Operator)
K9 - Brugernavnet skal være mellem 2 og 20 karakterer
K10 - Initialerne skal være mellem 2 og 4 tegn
K11 - Passwordet skal indeholde mellem 6 og 50 tegn
K12 - 3 af 4 følgende kategorier skal opfyldes for passwordet: der skal være 1 småt bogstav, der skal være 1 stort password, der skal være 1 cifer fra 0-9, der skal være et specialtegn ({'.', '-', '_', '+', '!', '?', '='}
K13 - Alle metoderne i interfacet IUserDAO skal implementeres
K14 - Når en ny bruger oprettes skal der genereres et password til dem
K15 - Når en ny bruger oprettes skal de have det laveste ledige ID
K16 - Programmet skal kunne afsluttes fra menuen

Analyse

UseCase Diagram



Use case beskrivelse

Fully dressed

Use Case Section	Comment
Navn	Opret bruger
Scope	CDIO 1
Level	User-goal
Primær aktør	Bruger
Preconditions:	Programet startet og en database valgt
Postconditions:	En ny bruger er oprettet
Hoved succes scenarie:	1) Brugeren trykker på 1. Opret Bruger 2) Navn bliver indtastet efterfulgt af enter. 3) Initial bliver indtastet efterfulgt af enter. 4) CPR-nummer bliver indtastet efterfulgt af enter. 5) En rolle eller flere roller vælges fra menuen. 6) Brugeren bliver oprettet efter valg af roller.
Alternative scenarier:	3.a: Den indtastet initial er for lang eller for kort 1. Systemet melder fejlkode 2. Beder brugeren om at indtaste initial igen 4.a: Den indtastet CPR-nummer er i forkert format 1. Systemet melder fejlkode 2. Beder brugeren om at indtaste CPR-nummer igen 5.a: Brugeren indtaster en rolle som ikke findes 1. Systemet melder fejlkode 2. Beder brugeren om at indtaste en rolle eller roller igen

Use Case Section	Comment
Navn	Vis brugere
Scope	CDIO 1
Level	User-goal
Primær aktør	Bruger
Preconditions:	Programet startet og en database valgt
Postconditions:	Liste af brugere bliver vist
Hoved succes scenarie:	1) Brugeren trykker på 2. List brugere efterfulgt af enter

Fully dressed

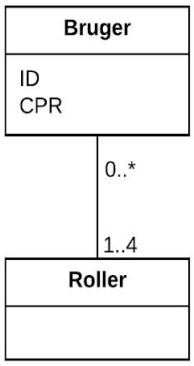
Use Case Section	Comment
Navn	Opdater bruger
Scope	CDIO 1
Level	User-goal
Primær aktør	Bruger
Preconditions:	Programet startet og en database valgt
Postconditions:	En brugere bliver ændret
Hoved succes scenarie:	1) Brugeren trykker på 3. Ret bruger efterfulgt af enter 2) Brugeren indtaster en ID for den bruger som skal ændres. 3) En af menupunkt vælges(1.Navn , 2. Brugernavn , 3.Kodeord , 4.Roller) 4) Brugeren indtaster ændringen i forhold til den valgte menupunkt.
Alternative scenarier:	2.a: Bruger Indtaster forkert ID 1. Systemet melder fejl 2. Bruger indstaster en ny ID 3.a: Brugeren vælger 1.Navn

	1. Systemet beder om at indtaste navn 2. Brugere indtaster den nye navn 3.b: Brugeren vælger 2. Initial 1. Bruger indtaster den nye initial 2. Den indtastet initial er for kort eller for lang 3. Systemet melder fejl 4. Bruger indtaster igen 3.c: Brugeren indtaster en rolle som ikke findes 1. Systemet melder fejlkode 2. Beder brugeren om at indtaste en rolle eller roller igen

Fully dressed

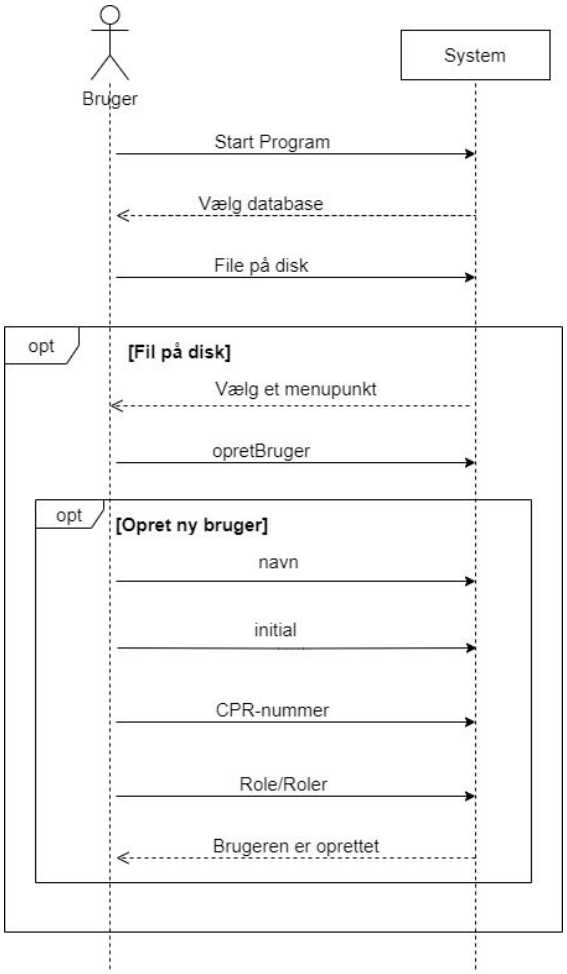
Use Case Section	Comment
Navn	Slet bruger
Scope	CDIO 1
Level	User-goal
Primær aktør	Bruger
Preconditions:	Programet startet og en database valgt
Postconditions:	En brugere bliver ændret
Hoved succes scenarie:	1. Brugeren trykker på 4. Slet bruger efterfulgt af enter 2. Brugeren indtaster en ID for den bruger som skal slettes. 3. Den indtastet bruger bliver slettet fra systemet

Domæne model



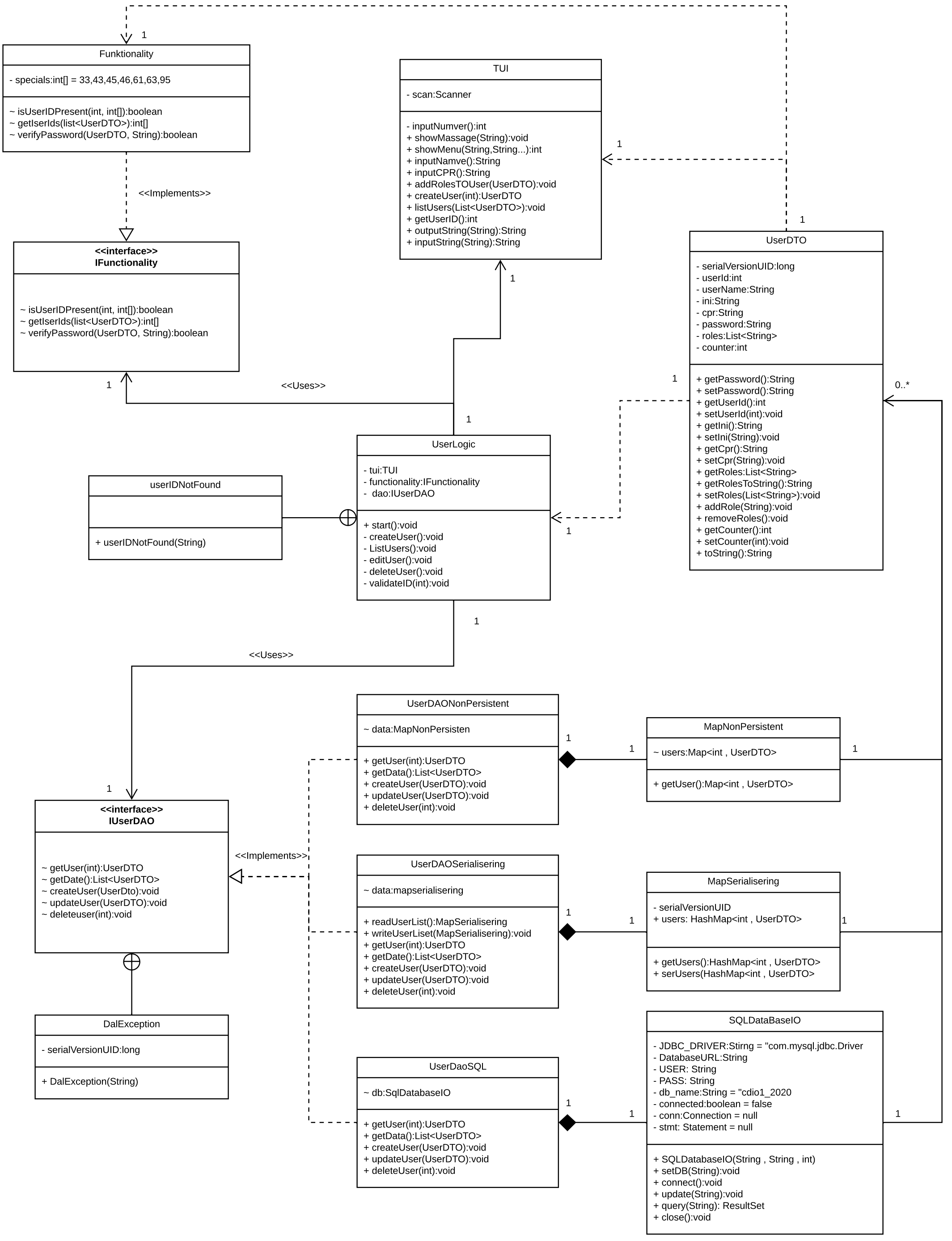
Systemsekvensdiagram

Nedenfor ses et systemsekvensdiagram over use casen “opret bruger”:



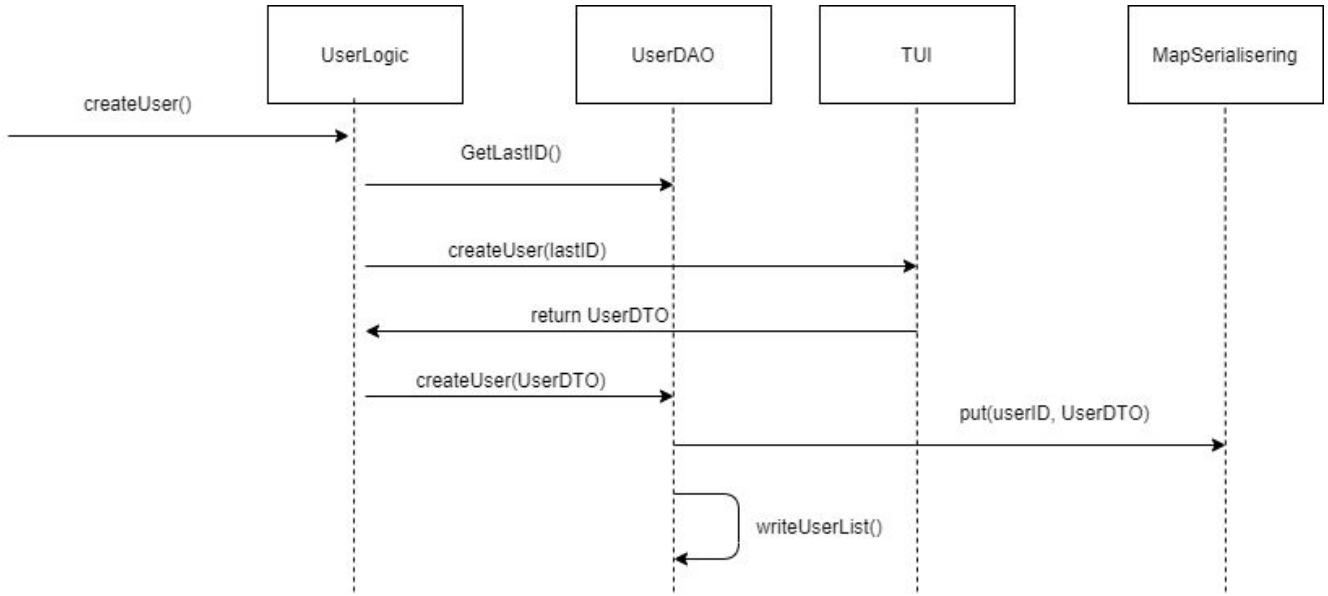
Design

Klassediagram - CDIO 1



Sekvens Diagram

Sekvens-diagrammet herunder viser hvad der sker i programmet når man vælger “Opret ny bruger” i interfacet.



Implementering

Overvejelser

3-lags modellen

Vi har i vores projekt valgt at benytte os af trelagsmodellen, for at gøre systemets struktur mere overskuelig og de enkelte moduler nemmere at forstå. Samtidig bidrager modulariseringen til et system der er mere genanvendeligt, nemmere at vedligeholde samt lettere at udvide i fremtiden, da de enkelte delelementer kan betragtes isoleret.[1]

Effekten af modulariseringen er kun yderligere forstærket af, at der i hver af de 3 lag (PL, BLL og DAL) forefindes interfaces der beskriver præcis hvilke metoder der ønskes implementeret. På den måde var det nemt at implementere tre forskellige versioner af datalaget, navnligt en ikke-persistent HashMap baseret, en persistent serialiserings baseret og en persistent SQL baset. Vores Main-klasse er således indrettet til at bestemme systemet aktuelle konfiguration. På samme måde vil det i fremtiden være nemt at implementere f.eks. et andet UI.

Vi har i vores løsning valgt at benytte os af en controller, i et forsøg på at gøre grænsefaldelaget så uafhængigt og på linje med ovenstående som muligt. Alt logik om hvad der skal ske hvornår i programmet er således blevet overført til vores controller.

Det er værd at bemærke, at vores controller også kender til DAL. Vi har valgt denne løsning, da der i vores program forekommer særligt mange rene datamanipulationsopgaver, der ikke har behov for at gå gennem funktionalitetslaget; tag f.eks. en metode som .createUser(). Bemærk også, hvordan vi til at overføre information mellem de forskellige lag benytter en såkaldt DTO (data transfer objekt), til at overskuelige processen.

Bruger ID

Den måde, vi opretter bruger-id'en på, er ved at se på listen over brugere, hvis der ikke er nogen bruger på listen, så bliver der først oprettet i TUI ved at kalde på funktion creatUser som tager bruger id 10 som et argument, hvilket vil sige at den id som er givet ved argument vil blive forøges med 1, når vi kalder funktionen. Så opretter vi også brugeren med den id ned på vores data. Det betyder, at den første bruger kommer til at få bruger-id 11.
Ellers, hvis der allerede er brugere på listen, kalder vi en funktion setCounter, hvor vi opdaterer den statiske variabel ’’counter’’ i UserDtu til den sidste brugers id, som bruges til creatUser funktion, som så igen bliver forøget i TUI’en funktion createUser, derefter bliver useren oprettet ned på data med næste id afhængig af den tidligere brugers id.

[1] “3 lags modellen” af Mads Nyborg og Uffe Kofoed, 2013, s. 1-2.

Database server

Vi har i vores løsning valgt at benytte en online MYSQL server som database, for at undgå at brugeren skal have en lokal database server liggende. Databasen er opsat med tre tabeller: userdto (brugeren), roles (roller) og userdto_roles (Hvilke userdto brugere har hvilke roller).

userdto

userID	userName	ini	cpr	password

roles

role_name

userdto_roles

userID	role_name

Serialisering (fil på disk)

En af database implementering i vores system har været en fil på disken. I denne implementering har vi brugt serialiserings funktionen hos Java programmeringssprog. Filen lægger i data mappen og hedder “users.data”. Denne fil bliver læst hver gang, når man vælger, at programmeret skal køre med en fil på disk database. Der bliver indlæst in objekt af klassen UserDAOSerialisering. Denne fil bliver overskrevet hver gang når man opretter en ny bruger, ændrer en bruger i databasen og når der bliver slettet en bruger. I tilfælde, at hvis der ikke eksisterer en fil i data mappen som hedder “users.data”, så vil der blive oprettet en ny fil.

Password standarder

Hver brugers password skal følge nogle specifikke regler[2]. Der er derfor blevet implementeret en metode i Functionality klassen. Først bliver længden af passwordet verificeret. Derefter gennemgås hvert bogstav i passwordet, og validere om bogstavet er tilladt. Samtidig med dette holdes der styr på om kriterierne for de 4 forskellige grupper af bogstaver (lowercase, uppercase, tal og specialtegn) bliver brugt. Derefter bliver passwordet søgt igennem for om det indeholder brugerens navn eller ID, og til sidst bliver der talt op om mindst 3 kategorier er brugt. Hvis fejler undervejs, kaster koden en exception, der beskriver fejlen i dens besked. Hvis passwordet til gengæld går fri i alle undersøgelserne, returnere metoden true.

[2] https://password.dtu.dk/

Demodata

Nedenfor ses den demodata som alle systemets databaser er blevet initialiseret med

ID	Navn	Ini.	CPR	Kodeord	Roller
11	Volkan	VOL	0912941235	SkiftMig!	Admin
12	Mikkel	MIK	3007902345	SkiftMig!	Pharmacist
13	Mark	MAR	1603783455	SkiftMig!	Foreman
14	Talha	TAL	2301974565	SkiftMig!	Operator
15	Frederik	FRE	0501025675	SkiftMig!	Pharmacist, Operator
16	Lasse	LAS	1104966785	SkiftMig!	Admin, Pharmacist, Foreman, Operator

Brugervejledning

Installation og kørsel af CDIO1.jar

- Programmet kræver Java 8
- Programmet er bygget til at køre på en Windows maskine.
- Udpak 11_del1.zip
- Åben mappen og kør run.bat

Kompilering, installation og afvikling af kildekode

- Start med at klone vores git repository
    - *git clone https://github.com/Kameldrengene/CDIO1*
- Åben projektet i IntelliJ
- Gå ind i File->Project Structure
- Gå ind under Artifacts og add JAR from modules with dependencies
- Vælg Main Class og sæt den til Main
- Tryk ok og ok til Project Structure
- Udvid *Available Elements* og dobbelt klik på *mysql-connector-java* og *UserDAODiscImplementation*
- Gå ind i Build->Build Artifacts->CDIO1.jar>Build
    - Programmet bliver nu bygget
- Gå ind i CDIO1\out\artifacts\CDIO1_jar\ og lav folderen CDIO1\src\data
- Indsæt CDIO1\src\data\users.data filerne i CDIO1\out\artifacts\CDIO1_jar\CDIO1\src\data
- Gå ind i CDIO1\out\artifacts\CDIO1_jar\ og åben en cmd og kør kommandoen:
    - *java -jar CDIO1.jar*

Test

BLL

Test case ID	TC01
Class	BLL - Functionality
Method	.isUserIDPresent(ind ID, int[] IDs)
Summary	It is tested whether or not the method is capable of properly assessing if a given int is present in a given array
Type	Junit 4
Procedure	An ID and an array is given the methods and the result is evaluated
Data	Following the format (ID, array): (3, {1,2,3,4,5}), (3, {1,2,4,5}), (11, {1,2,4,5}), (2, {10,6,4,2}), (3, {}), (-1, {1,2,4,5})
Expected	True, false, false, false, true, false, false
Status	Passed
Created by	Mikkel Danielsen
Creation date	11/03-2020
Last tested	11/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418

Test case ID	TC02
Class	BLL - Functionality
Method	.getUserIDs(List<UserDTO> users)
Summary	It is tested whether or not the method is capable of correctly identifying the user IDs in a given UserDTO-list
Type	Junit 4

Procedure	An UserDTO is created, given an ID and inserted into a UserDTO-list. The methods output is then asserted. This is done four times.
Data	Users with IDs 13, 100, -1 and 0.
Expected	13 13, 100 13, 100, -1 13, 100, -1, 0
Status	Passed
Created by	Mikkel Danielsen
Creation date	11/03-2020
Last tested	11/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418

Test case ID	TC03
Class	BLL - Functionality
Method	verifyPassword()
Summary	Der bliver testet for om metoden smider de rigtige exceptions ved fejl og om korrekte passwords går igennem.
Type	JUnit 4
Procedure	Metoden verifyPassword() bliver flere gange kaldt for en test bruger. Hver gang ændres Passwordet, der bliver testet, for at se om der bliver smidt de korrekte exceptions. Hver gang en test går som planlagt bliver 1 lagt til count. Count starter på 0.
Data	Count = 0 testUser = [userId=11, userName=Volkan] Passwords testet: 1. Aa1 2. AAAAAAAAAbbbbbbbbbbCCCCCCCCCCCddddddddddEEEEEEEEEE! 3. Aa123aA¤ 4. Aa123aA@ 5. Aa123aA& 6. Volkan123

	7. Aa11!23 8. AAAAAAAAAA 9. AAAAAbbbbb 10. AAAAA!!!!! 11. AAAbbb123 12. AAAbbb123!. 13. !+-.=?_aA
Expected	13
Status	Passed
Created by	Frederik Koefoed
Creation date	12/03-2020
Last tested	12/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418

DAL

Test case ID	TC04
Class	DAL - UserDAOSQL
Method	.getUser(), .getData(), .createUser(), .updateUser(), .deleteUser()
Summary	It is tested whether or not the methods are capable of connecting to the database, and modifying the columns.
Type	Junit 4
Procedure	A new user is created and tested if the user data is in the database.
Data	UserDTO.setCpr("1122334455"); UserDTO.setIni("MRM"); UserDTO.setUserId(1); UserDTO.setUserName("Mark");
Expected	True

Status	Passed
Created by	Mark Rune Mortensen
Creation date	11/03-2020
Last tested	11/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418

Test case ID	TC05
Class	DAL - UserDAOSerialisering
Method	getData()
Summary	Der bliver testet om at databasen returer korrekte antal af bruger som er allerede oprettet i systemet.
Type	Junit 4
Procedure	Metoden getData() bliver kaldt og antallet af oprettede bruger bliver kontrolleret
Data	[userId=11, userName=Volkan, ini=VOL, roles=[Admin]] [userId=12, userName=Mikkel, ini=MIK, roles=[Pharmacist]] [userId=13, userName=Mark, ini=MAR, roles=[Foreman]] [userId=14, userName=Talha, ini=TAL, roles=[Operator]] [userId=15, userName=Frederik, ini=FRE, roles=[Pharmacist, Operator]] [userId=16, userName=Lasse, ini=LAS, roles=[Admin, Pharmacist, Foreman, Operator]]
Expected	6
Status	Passed
Created by	Volkan Isik
Creation date	11/03-2020
Last tested	11/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418

Test case ID	TC06
Class	DAL - UserDAOSerialisering
Method	createUser()
Summary	En test bruger oprettes i systemet og gemmes på disken
Type	Junit 4
Procedure	En bruger oprettes og resultat kontrolleres ved hjælp af getData() metoden
Data	UserDTO [userId=17, userName=Anders, ini=AND, roles=[Admin]]
Expected	AND
Status	Passed
Created by	Vokan Isik
Creation date	11/03-2020
Last tested	11/03-2020
Environment	IntelliJ IDEA 2019.2.3 (Ultimate Edition) Build #IU-192.6817.14, built on September 24, 2019 on Windows 10 Enterprise Version 1903 OS build 18362.418