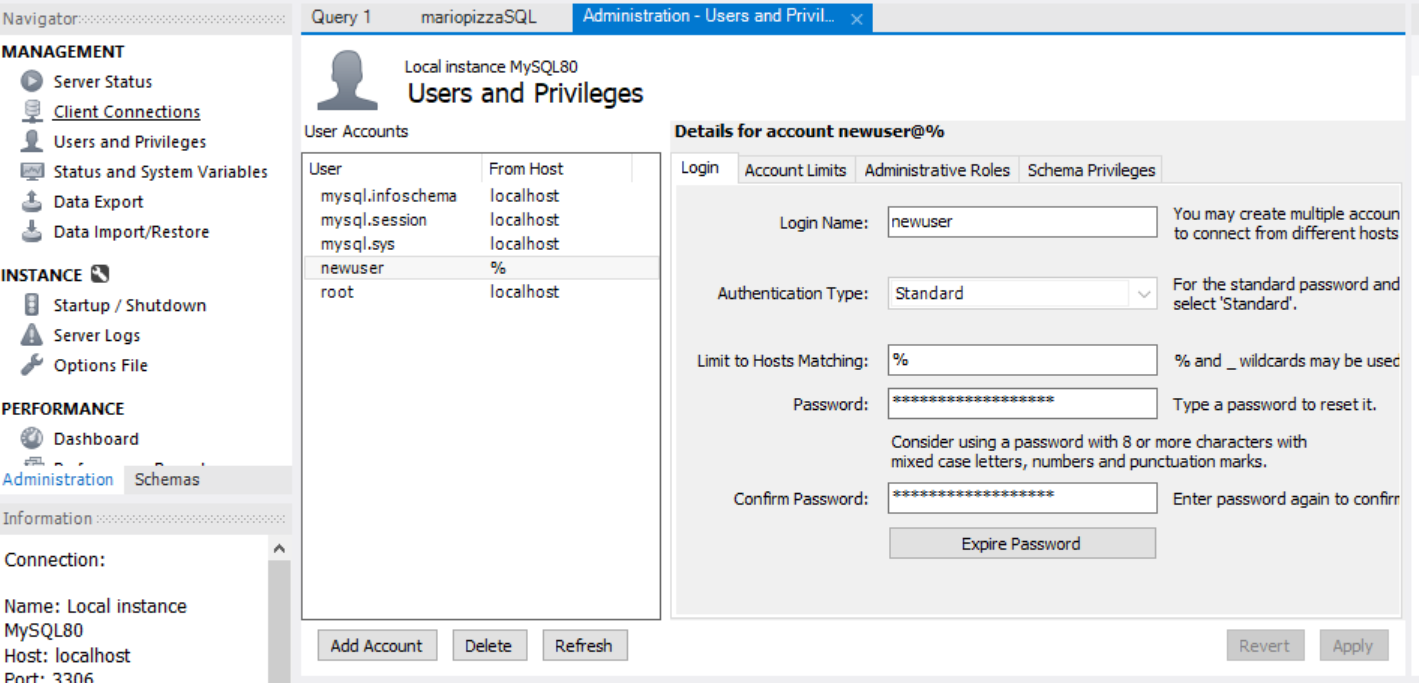


**Marios Pizzabar, opgave 3, gruppe 7**

Hermed en oversigt, over projektet Marios Pizzabar af gruppe 7, opgave 3.

**Vigtigt! Database forbindelse.**

Vores kode forventer at du forbinder til en lokal database. Det er derfor hardcoded at IP er localhost. Derudover forventer koden at databasen kan forbindes til med brugernavn: newuser og password ”Aa12345678”. Denne bruger skal derfor tilføjes til databasen før at koden virker. Husk den skal have ”DBA” afkrydset inde i adminstrative roles. Evt. kan man ændre source code så user og password er root og det tilhørende password som blev lavet ved database oprettelsen.



**Krav og udførsel**

Kravene for denne del af opgaven er følgende:

1. Validering af brugerinput
2. Mulighed for at tilføje flere pizzaer på en ordre
3. Tilføjelse af afhentningstidspunkt
4. Statistik på pizzaer
5. Generel program-robusthed med fejlhåndtering (exception handling)

Vi har udført bruger validering af alle bruger inputs. Der er lidt forskellige måder vi opnår dette på. Men primært tjekker vi inputtet enten ved hjælp af try/catch eller en simpel if statement, som ligger inde i en while løkke. While løkken bliver ved at med scanne efter et nyt input indtil vi har godkendt et input. Vi har også tilføjet flere steder menu systemet at man kan returnere ved skrive -1, hvis man fortryder sit valg. (fx hvis man trykker på fjern bestilling men ikke vil fjerne en bestilling alligevel)

Vi har ændret vores bestillings klasse så den nu har et bestillingstidspunkt som ved bruger input bliver valideret til at være i ”HH:MM” format (altså time og minutter). Samtidig har vi gjort så der er mulighed for at bestille flere pizzaer på samme ordre. De bliver gemt i et Int Array i bestillings objektet.

Vi laver vores statistik med en fileWriter og statistikken bliver outputtet til en txt fil. Klassen FileWriterHistorik holder en arrayliste med alle de færdige bestillinger (altså bestillinger der er fjernet fra den aktiveordre Database). Ved program afslutning (når man taster 9 i hovedmenuen) udskriver fileWriter alle ordrerne, samt en statistik på hvor gange de forskellige pizzaer er blevet solgt og den samlede omsætning.

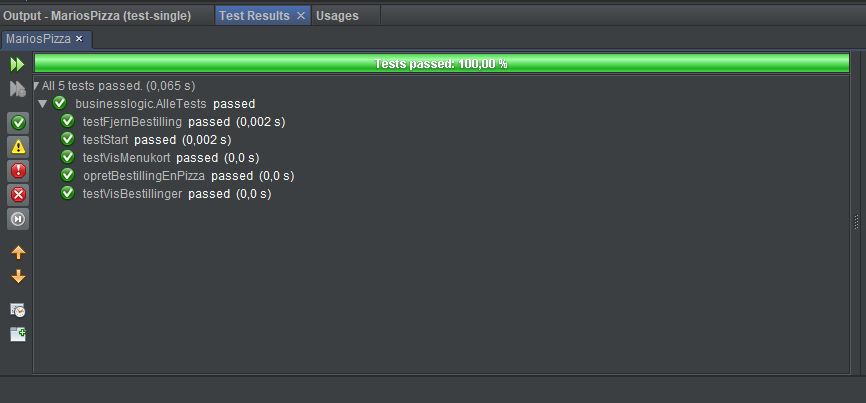


Exception handling er blevet lavet i de enkelte metoder og klasser hvor de kan opstå. Således at ingen af metoderne bare thrower exceptions videre til andre metoder.

Vi nåede ikke at implementere en storage mulighed der brugte filer i stedet for databasen. Dog er der et Storage Interface som man kan implementere til en klasse der kan skrive til filer og på den måde opnå den funktionalitet. Men på nuværende tidspunkt er der ikke implementeret nogen fileWriter til bestillinger eller menukort. Det er kun statistik der bruger en filewriter.

**Test**

Her er et screendump af resultatet fra test:



**Reference til Opgave 2 dokumentationen**

*Vi har først lavet et use-case diagram, som repræsentation for de 5 use-cases, vi har fundet relevante for at løse opgaven.*

*Ud fra hvad der i opgaven er beskrevet, er vi kommet frem til følgende prioritering af use cases, set fra kundens perspektiv:*

1. *Se bestillinger*
2. *Opret bestilling*
3. *Fjern bestilling*
4. *Se menukort*
5. *Se historik*

*Vi har for hver use-case startet med en test, da vi arbejder i et Test-driven-development miljø. Programmet kører udelukkende i terminalen, og har ikke noget grafisk user-interface. Den primære kommunikation med brugeren foregår med input af tal.*

*Vi har ikke implementeret nogen form for sortering af bestillingslisten, da Mario ikke var specifik i sine krav. Derfor er bestilingslisten sorteret i den rækkefølge ordrerne er tastet ind i.*

*Da Mario gerne vil have gemt historikken over salg af pizzaer, har vi valgt at eksportere færdige ordrer til en ekstern fil når programmet lukkes, som Mario kan tilgå, når han vil have et overblik. Denne fil over skrives hver gang programmet lukkes med user input(Der bliver indtastet 9). Der bliver ikke gemt nogen fil ved program crash eller anden uventet program afslutning.*