Backendprogrammering, grupp 1

Projekt struktur och funktionalitet

Arbetsfördelning mellan gruppmedlemmar

Vi utförde beslut angående projektets design och tema tillsammans. Överlag delades de 3 domän klasserna upp till varsin person, med de tillhörande klasserna på service- och dataaccess-nivå. Detta var varje gruppmedlems huvudsakliga ansvar. I övrigt arbetade vi gemensamt på alla överlappande delar. Tog stort ansvar i gruppen för att tester, kompatibilitet och funktionalitet skulle vara robusta samt att kommunikationen var frekvent.

Översikt

Vårt projekt är ett CRM-system för bokningar som sker i ett gym. Medlemmar kan boka sig till olika pass, varje pass har en tilldelad ledare. Varje medlem, pass och instruktör är sin egna entitet med en egen tabell, mellan dessa tabeller finns olika relationskap som M:M mellan pass och medlem, 1:M mellan instruktör och pass.

Verktyg

Java, Spring, JPA, Maven, Fat-Jar mfl.

Struktur

Varje **entity** har en tillhörande **service**- och **dao**-klass med implementation, både service och dao är programmerat utifrån interface.

I servicelagret finns även en klass för att genomföra bokningar och andra operationer som rör fler än en entity. Exv boka specifik medlem till ett pass, eller byta instruktör på ett pass.

Utöver dessa 3 lager finns även **advice** för att föra tidmätning och se vilka metoder som anropas när man kör tester.

Alla lager ligger indelade i respektive paket (**client**, **services**, **dataaccess**, **domain**, **advice**) under **src.main.java.se.yrgo**.

Konfigurationsfiler är uppdelade då projektet beter sig olika vid körning och testning, all konfiguration finns under **src.main.resources**.

För att bygga, köra & testa finns **3 stycken Bash-skript** i rotkatalogen, läs **README.md**

Huvudsakliga komponenter

Entities: domän-klasserna som **Member**, **GymClass**, och **Instructor**. Member har **många-till-många** relation med GymClass & GymClass har **många-till-en** relation med Instructor och tvärtom.

Data Access Layer: DAO klasserna tar en JPQL-query med entitymanager som skapar query i koden för att kommunicera med databasen för varje entity.

Service Layer: Affärslogik för operationer gentemot objekt som skickas vidare till DAO och därefter lagras i databasen.

Client Layer: All interaktion mellan användare och backend, användaren kan beroende på nivå (**member** eller **SysAdmin**), läsa, uppdatera, lägga till och ta bort data, samt de nödvändiga operationerna som krävs för att det faktiskt ska vara ett bokningssystem för ett gym.

Funktionalitet

Viktiga delar:

- Att kunna addera, uppdatera och ta bort i medlems perspektiv för Member och i administrations perspektiv för Member, Instructor och GymClass.
- Booking management Boknings managements kontroll över bokning av pass och dess avbokning.
- Integrationstest för varje service implementation
- Springframework med JPA integrerad