# Arkitektur

## Logisk arkitektur (Karsten)

Softwarearkitekturen udgør sættet af vigtige beslutninger om systemets organisering, valget af de strukturelle elementer og deres interfaces, og deres adfærd i form af kommunikationen mellem dem. [Larman, kap. 13]

Vi har valgt en lagdelt arkitektur med udgangspunkt i trelagsmodellen der overordnet består af tre lag, hhv. brugerflade, logik, og data access. Vores arkitektur er *relaxed* da et lag kan kalde ned til ethvert lag under sig—en udbredt arkitektur i informationssystemer.

Lagdeling af arkitekturen er en anvendelse af Layers Pattern og har en række fordele [Larman, kap. 13]:

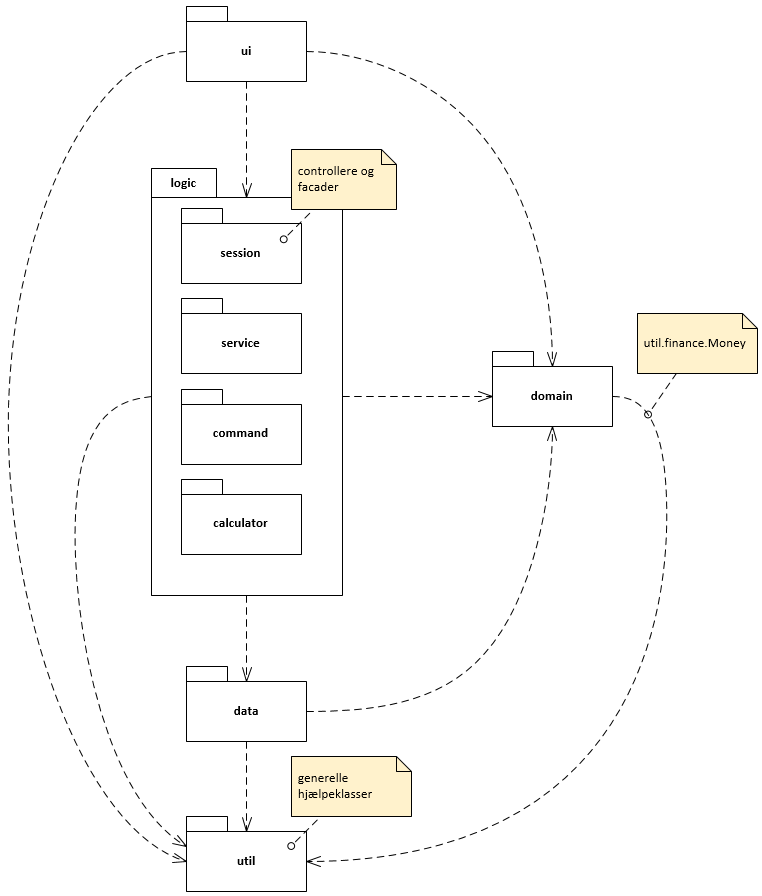
* Separation of concerns
* Adskillelse af high level fra low level services
* Adskillelse af applikationsspecifikke fra generelle services
* Reduceret kobling
* Øget potentialet for genbrug
* Øget gennemskuelighed

Det enkelte lags niveau afspejler dets stabilitet i forhold til forandringer. De nederste lag bør være de mest stabile da forandringer her vil propagere op igennem hele arkitekturen. Modsat er brugerfladen øverst fri fra koblinger fra de øvrige lag, og kan med større lethed udsættes for forandringer eller helt skiftes ud efter behov.

Pakkediagrammet i figur NNN viser vores logiske arkitektur. Entitetklasser har vi valgt at trække ud af logiklaget til en separat pakke: domain. Denne pakke er speciel da vi i OO typisk vælger at lade brugerflade og data access kende entiteterne frem for at gøre disse lag dataagnostiske. Dermed skaber vi en kobling som er i strid med kravet om udelukkende nedadgående kommunikation i arkitekturen. Det er en undtagelse vi kan leve med idet vi holder entiteterne som simple dataobjekter.

I logiklaget findes applikationens forretningsregler og service-klasser til entiteterne i domænet. Brugerfladen (ui) indeholder ”dumme” vinduer og paneler som blot er tynde klienter med ansvar for at præsentere output og delegere input videre ned til logiklaget. Dermed ingen logik i brugerfladen, jf. Model-View-separation. Endelig har datalaget ansvar for at oversætte forespørgsler fra serviceklasser i logik til CRUD-operationer i databasen.

Allernederst har vi en util-pakke med diverse generelle hjælpeklasser, fx en Money-klasse til repræsentation af pengebeløb.



*Figur NNN: Logisk arkitektur*