

Projekt Meddelandecentralen: Lägesrapport

Inlämningsdatum 21 okt av 23.59 **Poäng** 100 **Lämnar in** en länk till webbplats
Tillgänglig 17 okt kl 0:00–21 okt kl 23.59

Den här uppgiften låstes 21 okt kl 23.59.



Du och Viktor har bestämt att stämma av projektets status och se till att samtliga projektmål kommer uppfyllas. Detta är en mycket mindre formell typ av lägesrapport än den som du gjorde i *projekt SmartHuT*. Istället för en rapport på *projektets status* så syftar ni här att stämma av **produktens status**. I ett utvecklingsprojekt med flera utvecklare har ofta de mer seniora utvecklarna ett ansvar att en produkt byggs enligt den överenskomna designen. Detta görs oftast löpande via *Pull Request* på Github - eller gemensam kodgranskningar.


I detta projekt så ska du dels lägga fram en design på hur applikationen ska implementeras via ett **klassdiagram** - se övningen i denna veckas modul för steg-för-steg instruktioner på hur du gör detta. Ett klassdiagram är ett effektivt, och ofta använt verktyg för att dela upp en system - exempelvis en webbapplikation i flera delsystem som är enklare att hantera, kom ihåg vad du lärt dig om de *objektorienterade principerna*!

Precis som du lärt dig om att designa i förväg - kontra att dokumentera ditt system i efterhand så blir detta ett *mellanting* mellan de båda. Börja med att skapa ett diagram med de klasser som finns i ditt projekt, och lägg sedan till de klasser som du ämnar skapa i projektet. Tänk på att *klasser/entiter* i ditt diagram även kan representera vad som i Javascripts skrivs som funktioner - i.e. React komponenter. I Javascript används ofta funktioner för att representera mallen för objekt som ett alternativ till *class* semantiken.

Du har antagligen en vanlig, och övergripande uppdelning i ditt projekt - *backendkoden* i C# samt *frontendkoden* i Javascript; både dessa delar innehåller säkerligen olika klasser (eller motsvarande). Du måste i denna uppgiften dokumentera ditt klientgränssnitt (frontend). Men du

väljer själv om du vill avgränsa dig från att dokumentera backend/server koden.

Inlämning

I denna inlämning ska du skapa ovannämnda *klassdiagram*, samt lämna in detta som en bild i ditt git-repo från tidigare projektinlämning. Du ska också ta en ny bild över ditt kanban-bräde och bifoga i git-repot samt **tagga** den sista commit:en innan inlämning i ditt git-repo. Se [denna guide](https://stackoverflow.com/questions/18216991/create-a-tag-in-a-github-repository)  (<https://stackoverflow.com/questions/18216991/create-a-tag-in-a-github-repository>) på hur man skapar git-taggar från GitHub eller kommandotolken, du kan också skapa en *release* på GitHub ~ vilket i sin tur skapar en tagg.

- Taggen ska vara **lagesrapport**.

På Canvas lämnar du in en länk till ditt git-repo, för att signalera att du är färdig med uppgiften. Det ska såklart vara samma git-repo som tidigare. Genom att genomföra följande uppdatering av ditt git-repo så visar du förmåga att genomföra och rapportera på programmeringsuppgifter (Läranderesultat 7) och att driva utvecklingsprojekt med kontinuerlig rapportering (Läranderesultat 5).

Nummer Poäng Kriterium Beskrivning			Läranderesultat
1	5		Korrekt länk till projektets git-repo är inlämnad på Canvas 5
2	10	1	Projektets git-repo innehåller en commit som är taggad <i>lagesrapport</i> 5, 7
3	20	2	Projektets git-repo är uppdaterat med en klassdiagram på denna commit 5, 7
4	20	3	Från klassdiagram går att förstå klientgränsnittets övergripande struktur 7
5	15	3	Klassdiagrammet följer UML standarden för klassdiagram 7
6	20	2	Projektets git-repo är uppdaterat med en bild över projektets kanban-bräde 5, 7
7	10	6	Kanban-brädet är lämpligt uppdaterat med nuvarande status på projektet 5