**Order**

Enligt våran kravanalys så bestämde vi att vi ska ha med orderhantering och att en order även agerar som ett kvitto. Men innan vi började med den så gick vi igenom med alla i gruppen om vad som skulle vara med på en order. I våran order modell har vi valt att använda oss av bygg mönstret som främst i vårat program bidrar till att kod för att skapa en order är både enklare att skriva och lättare att läsa. Så vi har suttit ner och läst på om bygg mönstret, gjort enkelt test projekt och även implementerat det manuellt i våran create auction så att när vi skulle implementera det till order så valde vi att göra det med Lombok då det kortar ner koden avsevärt. Men när vi skulle använda builder metoden i order service kunde vi inte det för att den inte var publik så vi fick ändå ange bygg klassen i order klassen med en tom konstruktor så den kunde nås.

Vi gjorde dock aktivitetsdiagrammet till skapa order metoden efteråt och det för att vi redan hade en gammal metod sen tidigare som vi valde att anpassa till den nya modellen samt bygg mönstret.

**Timer**

När order var klar och fungerade så var nästa steg att ta itu med när en order ska skapas. Och det vi kom fram till var att det borde ske automatiskt när en auktions tid har tagit slut och att auktionen har bud.

För att komma fram till en lämplig lösning så kollade vi på timer med TimerTask och ScheduledExecutorService och testade lite med det, men i och med att vi använder oss av Spring Boot så kunde man använda annoteringen @Scheduled som man enkelt kunde lägga till till en metod som man ville skulle köras en gång eller i intervall så slapp man en del kod även där.

Så vi har nu en funktion som startas när Celemo backend körs igång som på en bestämd intervall på 20 sekunder emellan kollar auktionerna om det är någon som gått ut. Ifall det är så och att auktionen då har bud lagda så kommer en order skapas och i samband med det så skickar vi även ut en notifiering till berörda och ifall säljaren inte är online så får denne ett mail.

Detta funkar bra i denna lilla skala som vi har nu men skulle det vara så att vi har extremt mycket auktioner som ska kollas igenom så kommer vi nog få problem med prestandan. Man märker att om det är ex 20 auktioner ska ska hanteras på raken så är det lite fördröjning mellan varje auktion i funktionen vilket innebär att funktionen kommer ta sin tid.  
Att åtgärda detta kanske man kan göra genom att dela upp auktionerna mer med olika metoder som kollar och där kommer antagligen multi threading att spela stor roll.   
Ett sätt att uppnå multi threading för metoder som använder sig av annoteringen @Scheduled är att lägga till en specifik rad i application.properties där man enkelt kan ange hur många trådar programmet ska ha tillgång till.