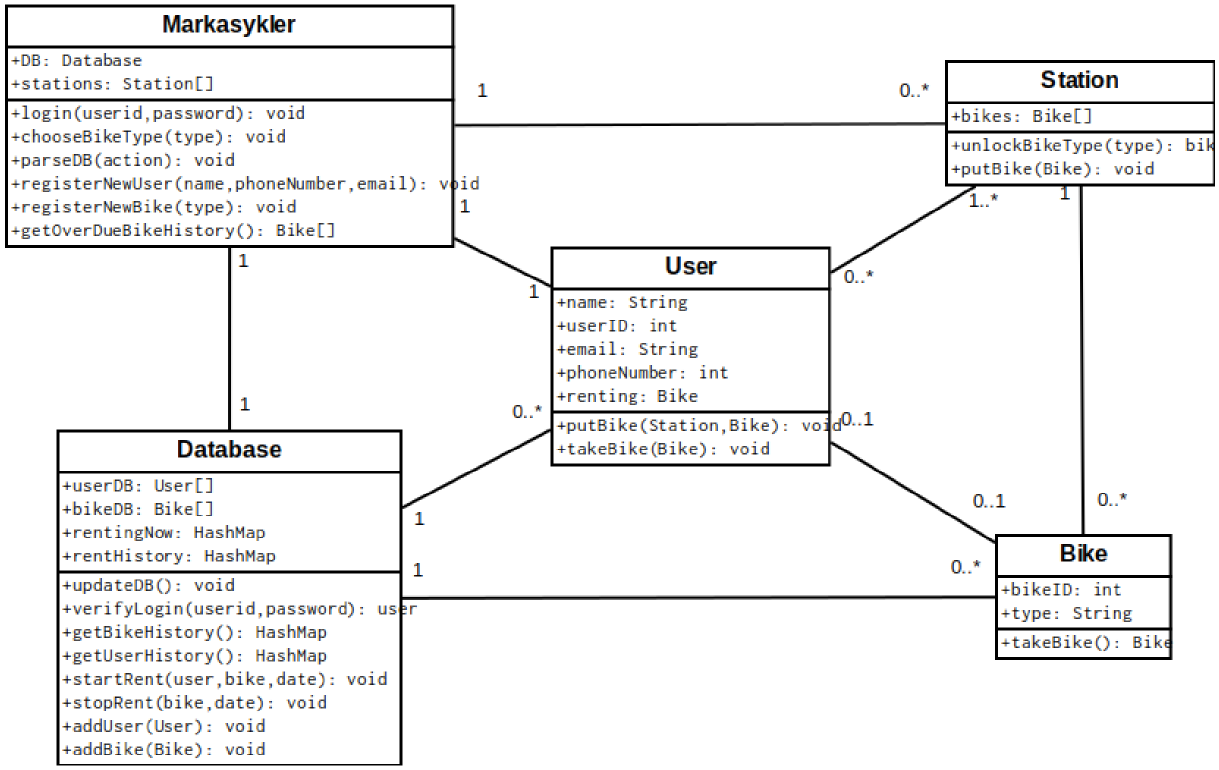
# Modellering av krav

Jan Inge Lamo

08.04.16

## 1 Klassediagram

Markasykler class diagram:



In the class diagram I opted for a more traditional system structure with a database instead of storing information inside the Bike class. While some may say this data structure increase the code complexity without any real benefit, but there is no harm in such a small system, and a detour from a provided object-oriented structure is a fresh breath.

## 2 Sekvensdiagram

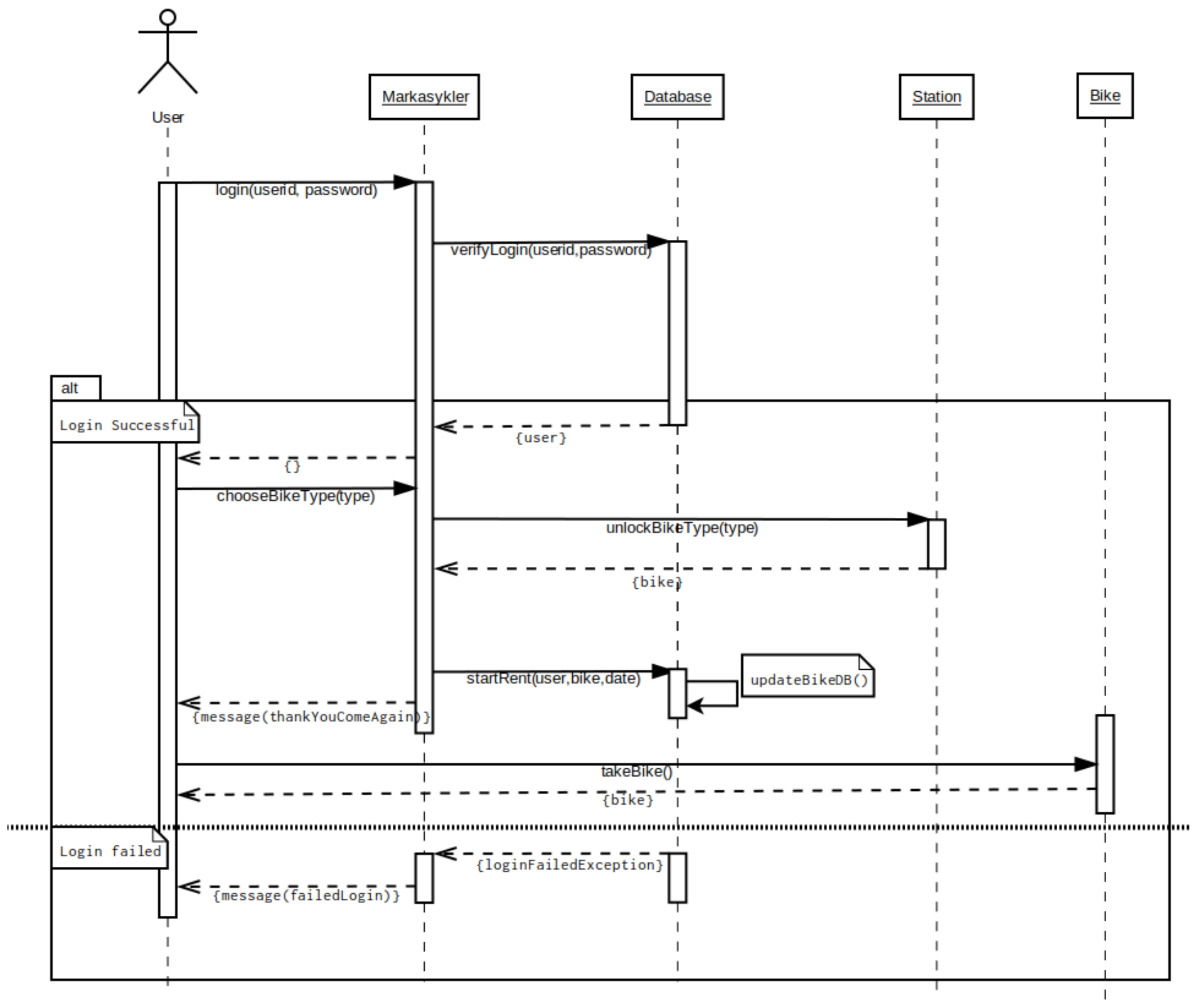
a)

1. User send userid and password to Markasykler.login()
2. Markasykler send userid and password to Database.verifyLogin()
3. Database return User object to Markesystem
4. User specify bike type to Markasykler
5. Markasykler unlocks the bike at this instance on the Station
6. Station return the bike
7. Markasykler send user- and bike object, and time/date to Database
8. Database updates
9. User takes the unlocked bike
10. Bike return itself to user (user is in possession of the bike)

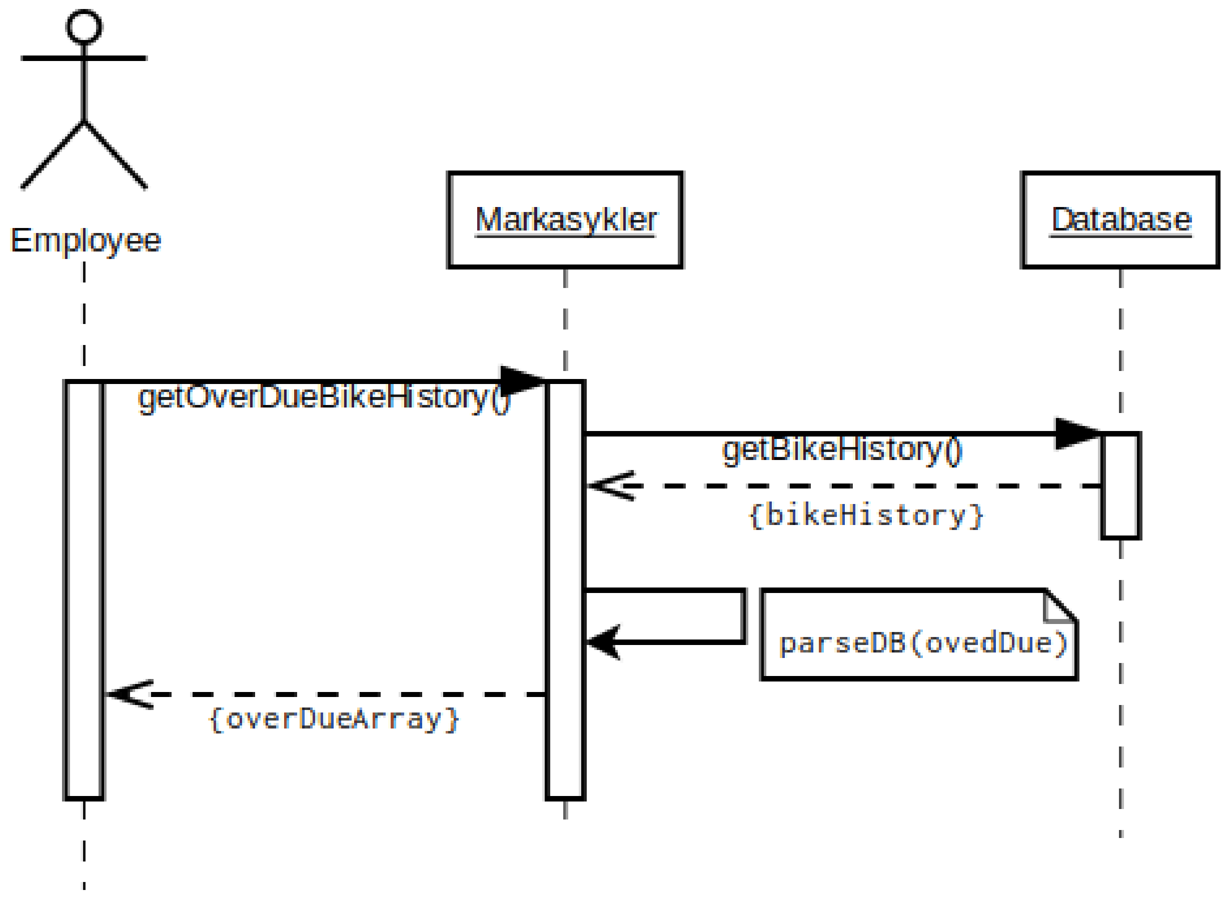
Alternate flow 1, step 3: Invalid login credentials.
A1.1: Database returns loginFailedExeption to Markasykler
A1.2: Markasykler give user an error message.

Precondition: The bike type must be available on the station the user is interacting with the Markasykler system.

b) Sequence diagram: ”rent a bike”



c) Sequence diagram: "View over due bikes"



## 3 Prosjektplanlegging

a)
Activities included in the development of the Markasykler system:

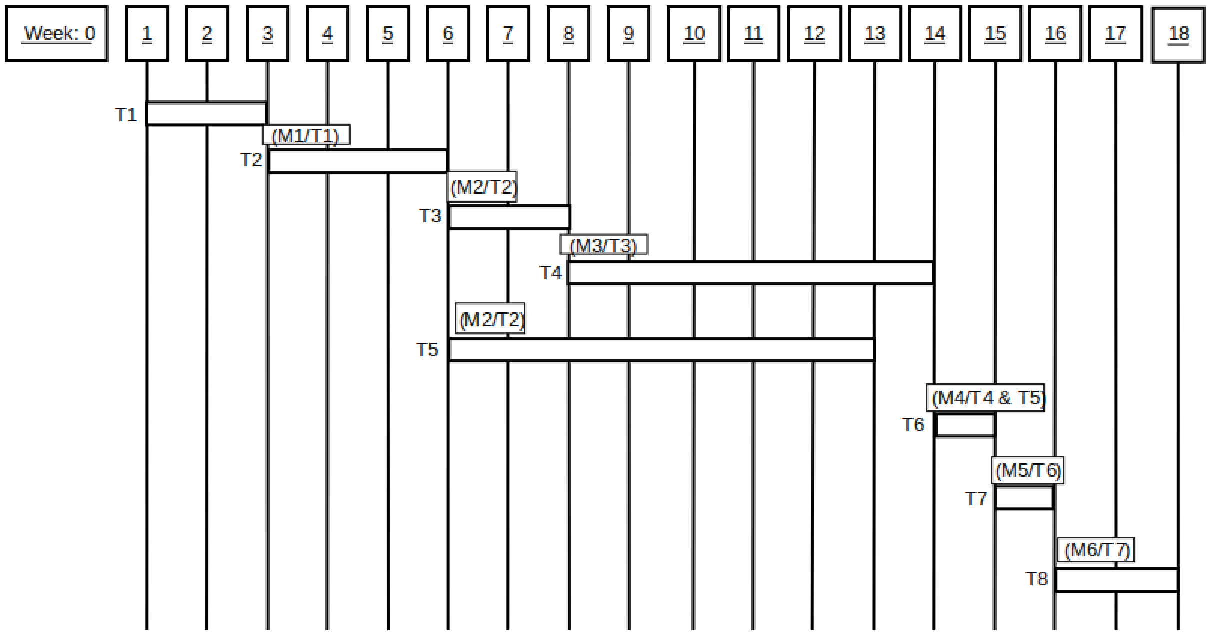
- *System specification* Map out all the necessary specifications for the system.
- *System design* Design the system based on system specifications/requirements and use cases for the system.
- *Code recycling* If possible find old code for similar systems and use the old code base for cheap reduction in development time.
- *Software development* Write the software the system, including the database and the website.
- *Hardware development* Design and build the bike stations.
- *Unit testing* Test each component in the system in an isolated environment.
- *Interface testing* Test that all components function and communicate properly with each other, this includes the Ruter payment system.
- *System testing* Test that the system works as a whole.

b)

| | Duration(weeks) | Name | Dependecy |
|---|---|---|---|
| T1 | 2 | *System specification* | |
| T2 | 3 | *System design* | T1(M1) |
| T3 | 2 | *Code recycling* | T2(M2) |
| T4 | 6 | *Software development* | T3(M3) |
| T5 | 7 | *Hardware development* | T2(M2) |
| T6 | 1 | *Unit testing* | T4,T5(M4) |
| T7 | 1 | *Interface testing* | T6(M5) |
| T8 | 2 | *System testing* | T7(M6) |

-

c)



d) Risk managment

| Risk | Probability | Consequence | Responsible |
|---|---|---|---|
| Key developers are sick or just absent during critical moments in the development | Moderate | Serious | Human resources. Spread the workforce and multiple groups. |
| Break down in monetary negotiations with Ruter | Low | Disastrous | PR department. Produce bad press about Ruter if they do not accept our conditions |
| Somebody breaks in to the database | Moderate | Monstrosity | Senior development team. Do extensive penetration testing. |
| Delays in the hardware shipment, causing the | Repair broken bikes. | Gain steady income source by repairing broken bikes from the system. | |