STLDD - Software Top Level Design Document: NewPussSystem

Systemarkitektgruppen
Nina Khayyami | Johan Rönnåker | Martin Lichota | Marcel Tovar Rascon

Innehåll

Dokumenthistorik

Ver.	Datum	Ansv.	Beskrivning
0.1	30 september 2014	SG	Struktur för dokumentet
0.2	2 oktober 2014	SG	Lagt in klass-diagram, ER-diagram, sekvens-diagram samt lagt in all text. Färdigt för informell granskning.
0.3	7 oktober 2014	SG	Uppdaterat klassbeskrivningar, klassmetoder, klassdiagram, SQL-frågor, sessionsinformation, sekvensdiagram samt figurbeskrivningar.
0.4	7 oktober 2014	SG	Uppdaterat enligt informella granskningsprotokollet.
0.5	7 oktober 2014	SG	Uppdaterat klassdiagram, uppdaterat sekvensdiagram för klassen ProjectGroupAdmin, rättat stavfel och syntaxfel.
0.6	7 oktober 2014	SG	Uppdaterat klassdiagram, sekvensdiagram och ett litet fel i metodbeskrivningen.
0.7	15 oktober 2014	SG	Uppdaterat klassdiagram med två nya klasser och fixat småfel i ER-diagram.
0.8	15 oktoboer 2014	SG	Uppdaterat sessionsvariabler.
0.9	16 oktober 2014	SG	Lade till groupID som sessionsvariabel.
1.0	17 oktober 2014	SG	Ändringar i klasserna och metodbeskrivningarna samt klassdiagrammet. Färdig för baseline.
1.1	22 oktober 2014	SG	Tagit bort privata metoder, samt uppdaterat klassdiagrammet. Även lagt till en saknad metodbeskrivning i ReportGenerator.
1.2	23 oktober 2014	SG	Lag till klassen changePassword, uppdaterat klassdiagrammet och klassmetoder enligt problemrapporter.
1.3	24 oktober 2014	SG	Uppdaterat sekvensdiagrammen för då man lyckas lägga till och ta bort en tidrapport, samt tagit bort sekvensdiagram för då man misslyckas med dessa operationer. Uppdaterat sekvensdiagrammen för då man lyckas och misslyckas lista tidsrapporter.
1.4	26 oktober 2014	SG	Åtgärdat PR21. Uppdaterat sekvensdiagrammen enligt PR22.
1.5	30 oktober 2014	SG	Uppdaterat sekvensdiagrammen för klassen ProjectGroupHandling enligt PR27.

1 Inledning

Dokumentet beskriver högnivådesignen för NewPussSystem tidrapporteringssystem.

2 Referensdokument

1. SRS - Software Requirements Specification (Dokumentnummer PUSS144401, version 1.2)
2. STLDD - Software Top Level Design Document: BaseBlockSystem (Dokumentnummer PUSS12004, version 1.0)

3 Sammanfattning

Systemet är implementerat i en Tomcat server med följande klasser och klassmetoder:

3.1 Klasser

class servletBase Den här klassen är superklass åt alla servlets i systemet. För beskrivning se sidan 2 i referensdokument 2.

class Administration Se sida 2 i referensdokument 2. En ändring har gjorts där metoden deleteUser ska returnera en boolean och inparametern ändras till (int id).

class LogIn Se sida 2 i referensdokument 2.

class Access Den här klassen använder sig av tabellen log och users i databasen för att hålla koll på vilka som är inloggade och aktiva. Har en användare varit inaktiv mer än 20 minuter, se dokumentreferens 1 krav 6.2.9, så nekar den åtkomst och loggar ut användaren.

class ProjectGroupAdmin Den här servleten ärver från ServletBase. Först kontrollerar den att användaren har behörighet att besöka sidan. Om så är fallet presenteras en tabell med information om varje projektgrupp (namn, projektledare). I tabellen kan administratören välja att radera gruppen och att lägga till/radera en projektledare. Projektgruppsnamnet i tabellen länkar administratören till projektledarens ändringsmeny. Nedanför tabellen finns en meny där administratören kan välja att lägga till nya projektgrupper, gå tillbaka till startsidan för administration, gå tillbaka till den gemensamma första inloggningssidan, samt logga ut.

class GroupHandling Den här servleten ärver från ServletBase och hanterar användarna i en projektgrupp.

class TimeReporting Den här servleten ärver från ServletBase och presenterar en undermeny med menyval relaterade till tidrapportering. Förutom undermenyn kommer första sidan att

vara tom. När ett val i undermenyn görs, kommer klassen att hantera den valda funktionen. Funktionerna som finns inkluderar att lista tidrapporter, visa och uppdatera tidrapporter samt skapa nya tidrapporter.

class ReportGenerator Den här statiska klassen är en nästlad klass som har metoder som tar emot referenser till databasen och ritar upp en tidrapport med relevanta data. De olika tidrapporter som kan ritas upp är: en redigerbar, ny och tom tidrapport, en redigerbar existerande tidrapport och en icke redigerbar existerande tidrapport.

class ProjectLeader Den här servleten ärver från ServletBase och visar en lista över alla användare. Härifrån kan projektledaren ändra projektmedlemmarnas roller.

class ReportHandling Den här servleten ärver från ServletBase och visar två listor; en lista med signerade och en lista med osignerade tidrapporter om man väljer att hantera tidrapporter. Projektledaren kan här signera eller avsignera rapporter.

class Statistics Den här servleten ärver från ServletBase och visar vissa funktioner om man väljer att generera statistik. Funktioner som kan väljas är t.ex. att visa vilken vecka som det har tidrapporterats mest tid.

class Start Den här servleten ärver från ServletBase och skapar den statiska menyn som finns tillgänglig för användaren på alla sidor i systemet.

class ChangePassword Den här servleten ärver från ServletBase och innehåller en html form för att användare ska kunna byta sitt lösenord.

3.2 Klassmetoder

Nedan beskrivs alla klassmetoder till klasserna som presenterats under rubriken 3.1 Klasser.

3.2.1 class Access

public boolean updateLog(int userID, String session)
Updates the log with a new timestamps for the given user and session.
@param userID: The requesting users id.
@param session: The requesting users session id.
@return boolean: True if the user has not been inactive for too long.

public boolean logInUser(int userID, String session)
Updates the database, sets the user as logged in, stores the current session id and current timestamp for the requesting user.
@param userID: The requesting users id.

@param session: The requesting users session id.
@return boolean: True if the user is not already logged in.

public boolean logOutUser(int userID, String session)
Updates the database, sets the user as logged out, removes the current session id and current timestamp for the requesting user.
@param userID: The requesting users id.
@param session: The requesting users session id.
@return boolean - True if the user is not already logged out.

3.2.2 class ReportGenerator

public static String viewReport(ResultSet data)
Prints out a time report in the right format and with the data specified by the ResultSet parameter.
@param data: Specifies which data to print in the time report.
@return String: A string containing html code.
public static String viewReport(HashMap<String, int> data)
Prints out a time report in the right format and with the data specified by the HashMap parameter.
@param data: Specifies which data to print in the time report.
@return String: A string containing html code.

public static String updateReport(ResultSet data)
Prints out a html-form in the right format and with the data specified by the ResultSet parameter.
@param data: Specifies which data to print in the html-form.
@return String: A string containing html code.

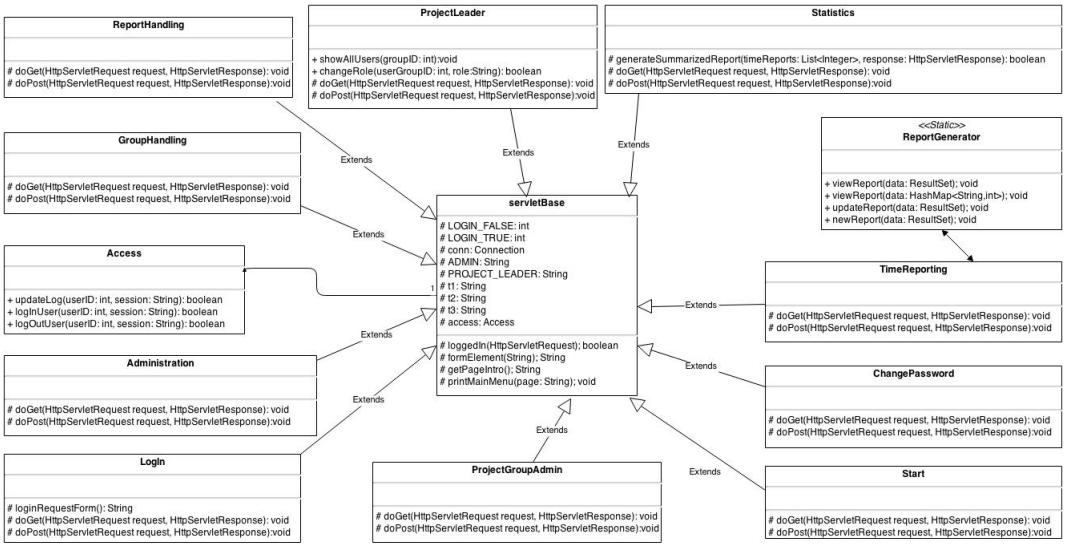
public static String newReport(int weekNumber)
Prints out a time report html-form prefilled with data.
@param weekNumber: Specifies for which week the data should be shown.
@return String: A string containing html code.

4 Klassdiagram

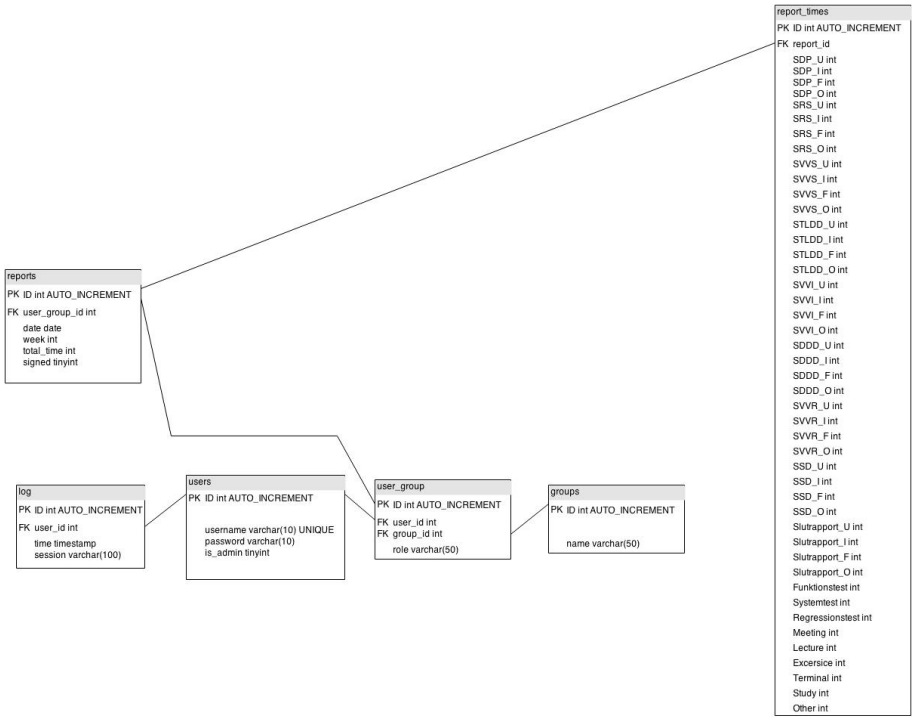
Ett klassdiagram med alla klasser visas i Figur 1.

5 Databas

ER-diagram över databasen för systemet visas i figur 2. Databasen kan skapas från scratch med följande SQL kommandon:



Figur 1: Klassdiagram över alla klasser i systemet.



Figur 2: ER-diagram över databasen för systemet

```
mysql> create database puss1404;
```

```
mysql> use puss1404;

mysql> CREATE TABLE IF NOT EXISTS `groups` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `name` varchar(20) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE KEY `name` (`name`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> CREATE TABLE IF NOT EXISTS `log` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `user_id` int(11) NOT NULL,
    ->   `time` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE
    ->     CURRENT_TIMESTAMP,
    ->   `session` varchar(100) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   KEY `user_id` (`user_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> CREATE TABLE IF NOT EXISTS `reports` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `user_group_id` int(11) NOT NULL,
    ->   `date` date NOT NULL,
    ->   `week` int(11) NOT NULL,
    ->   `total_time` int(11) NOT NULL,
    ->   `signed` tinyint(4) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   KEY `user_group_id` (`user_group_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> CREATE TABLE IF NOT EXISTS `report_times` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `report_id` int(11) NOT NULL,
    ->   `SDP_U` int(11) NOT NULL,
    ->   `SDP_I` int(11) NOT NULL,
    ->   `SDP_F` int(11) NOT NULL,
    ->   `SDP_O` int(11) NOT NULL,
    ->   `SRS_U` int(11) NOT NULL,
    ->   `SRS_I` int(11) NOT NULL,
    ->   `SRS_F` int(11) NOT NULL,
    ->   `SRS_O` int(11) NOT NULL,
    ->   `SVVS_U` int(11) NOT NULL,
```

```
    ->   `SVVS_I` int(11) NOT NULL,
    ->   `SVVS_F` int(11) NOT NULL,
    ->   `SVVS_O` int(11) NOT NULL,
    ->   `STLDD_U` int(11) NOT NULL,
    ->   `STLDD_I` int(11) NOT NULL,
    ->   `STLDD_F` int(11) NOT NULL,
    ->   `STLDD_O` int(11) NOT NULL,
    ->   `SVVI_U` int(11) NOT NULL,
    ->   `SVVI_I` int(11) NOT NULL,
    ->   `SVVI_F` int(11) NOT NULL,
    ->   `SVVI_O` int(11) NOT NULL,
    ->   `SDDD_U` int(11) NOT NULL,
    ->   `SDDD_I` int(11) NOT NULL,
    ->   `SDDD_F` int(11) NOT NULL,
    ->   `SDDD_O` int(11) NOT NULL,
    ->   `SVVR_U` int(11) NOT NULL,
    ->   `SVVR_I` int(11) NOT NULL,
    ->   `SVVR_F` int(11) NOT NULL,
    ->   `SVVR_O` int(11) NOT NULL,
    ->   `SSD_U` int(11) NOT NULL,
    ->   `SSD_I` int(11) NOT NULL,
    ->   `SSD_F` int(11) NOT NULL,
    ->   `SSD_O` int(11) NOT NULL,
    ->   `slutrapport_U` int(11) NOT NULL,
    ->   `slutrapport_I` int(11) NOT NULL,
    ->   `slutrapport_F` int(11) NOT NULL,
    ->   `slutrapport_O` int(11) NOT NULL,
    ->   `funktionstest` int(11) NOT NULL,
    ->   `systemtest` int(11) NOT NULL,
    ->   `regressionstest` int(11) NOT NULL,
    ->   `meeting` int(11) NOT NULL,
    ->   `lecture` int(11) NOT NULL,
    ->   `excersice` int(11) NOT NULL,
    ->   `terminal` int(11) NOT NULL,
    ->   `study` int(11) NOT NULL,
    ->   `other` int(11) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   KEY `report_id` (`report_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> CREATE TABLE IF NOT EXISTS `users` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
```

```
    ->   `username` varchar(10) NOT NULL,
    ->   `password` varchar(10) NOT NULL,
    ->   `is_admin` tinyint(4) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   UNIQUE KEY `username` (`username`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> CREATE TABLE IF NOT EXISTS `user_group` (
    ->   `id` int(11) NOT NULL AUTO_INCREMENT,
    ->   `user_id` int(11) NOT NULL,
    ->   `group_id` int(11) NOT NULL,
    ->   `role` varchar(20) NOT NULL,
    ->   PRIMARY KEY (`id`),
    ->   KEY `user_id` (`user_id`),
    ->   KEY `group_id` (`group_id`)
    -> ) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=1 ;

mysql> ALTER TABLE `log`
    ->   ADD CONSTRAINT `log_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users`
    ->   (`id`);

mysql> ALTER TABLE `reports`
    ->   ADD CONSTRAINT `reports_ibfk_1` FOREIGN KEY (`user_group_id`) REFERENCES
    ->   `user_group` (`id`);

mysql> ALTER TABLE `report_times`
    ->   ADD CONSTRAINT `report_times_ibfk_1` FOREIGN KEY (`report_id`) REFERENCES
    ->   `reports` (`id`);

mysql> ALTER TABLE `user_group`
    ->   ADD CONSTRAINT `user_group_ibfk_2` FOREIGN KEY (`group_id`) REFERENCES
    ->    `groups` (`id`),
    ->  ADD CONSTRAINT `user_group_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES
    ->    `users` (`id`);
```

6 Information lagrad i sessioner

I en pågående session sparas följande attribut i sessionen:

Integer state: Används för att beskriva om användaren är inloggad eller inte. Följande två tillstånd finns:

0: Ej inloggad
1: Inloggad

Session session: Användarens sessions-id.

String name: Användarens användarnamn, e.g. 'admin'.

int userID: Användarens id.

int groupID: Id till gruppen som användaren är inloggad på.

int userGroupID: Id från tabellen user_group som är ett id som sammanlänkar användare och grupp.

String role: Användarens roll i den aktuella gruppen.

7 Sekvensdiagram

Observera att inte alla meddelanden visas i följande sekvensdiagram, utan endast de meddelanden som krävs för att förstå sekvensen.

7.1 class Administration

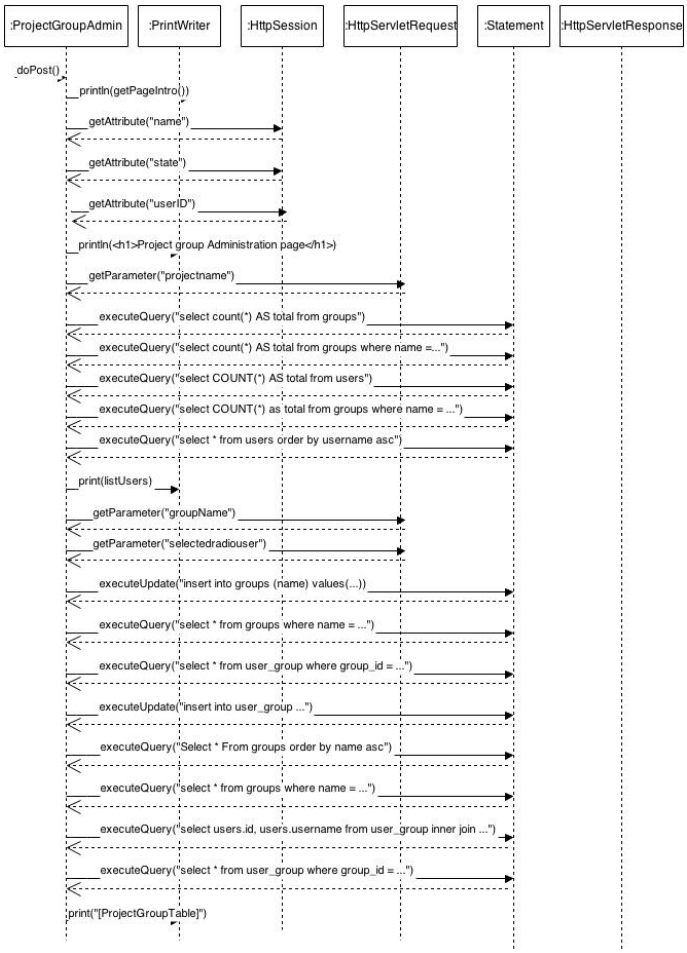
Figur 2 i dokumentreferens 2 visar sekvensen för hur servleten Administration hanterar att administratören lägger till en ny användare.

7.2 class ProjectGroupAdmin

Figur 3 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att lägga till en ny projektgrupp och Figur 4 visar hur den hanteras när den misslyckas.
Figur 5 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp.

7.3 class GroupHandling

Figur 6 visar hur servleten GroupHandling hanterar en förfrågan om att lägga till en användare i en projektgrupp, medan Figur 7 visar hur det hanteras när den misslyckas.
Figur 8 visar hur servleten GroupHandling hanterar en förfrågan om att ta bort en användare från en projektgrupp, medan Figur 9 visar hur det hanteras när den misslyckas.



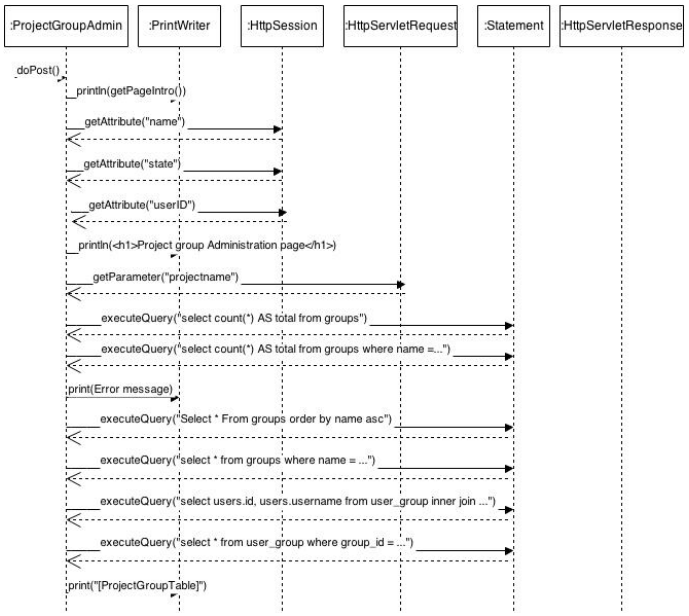
Figur 3: Sekvensdiagram som visar hur en lyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin.

7.4 TimeReporting

Figur 10 visar hur servleten TimeReporting hanterar en förfrågan om att skapa en ny tidrapport som läggs till i databasen.
Figur 11 visar hur servleten TimeReporting hanterar en förfrågan om att ta bort en tidrapport.
Figur 12 visar hur en förfrågan om att uppdatera en tidrapport som ännu inte blivit signerad hanteras.
Figur 13 visar hur servleten TimeReporting hanterar en förfrågan om att skriva ut en användares alla tidrapporter, Figur 14 visar hur den hanteras när den misslyckas.

7.5 ProjectLeader

Figur 15 visar hur servleten ProjectLeader hanterar en förfrågan om att ändra en projektroll för en användare.



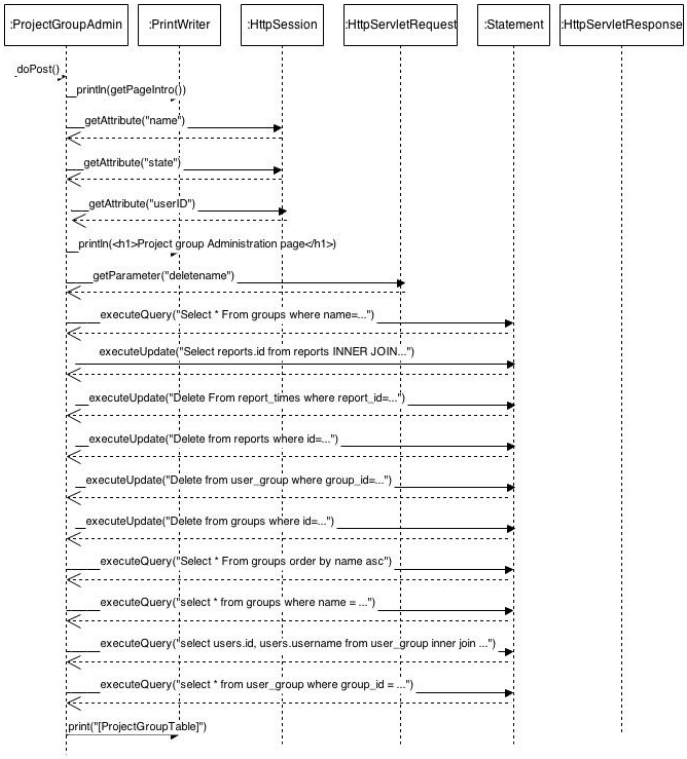
Figur 4: Sekvensdiagram som visar hur en misslyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin.

7.6 ReportHandling

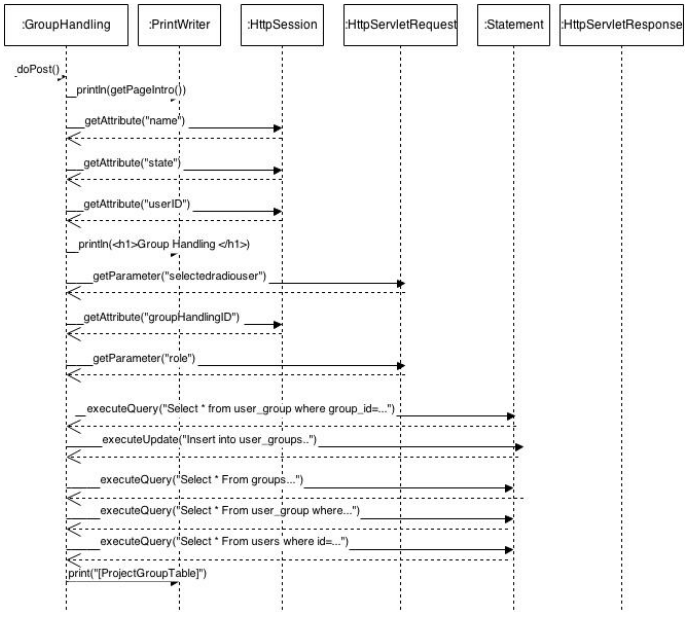
Figur 16 visar hur servleten ReportHandling hanterar en förfrågan om att visa alla tidrapporter i projektgruppen, signerade samt osignerade. Figur 17 visar hur servleten ProjectLeader hanterar en förfrågan om att signera en rapport.

7.7 Statistics

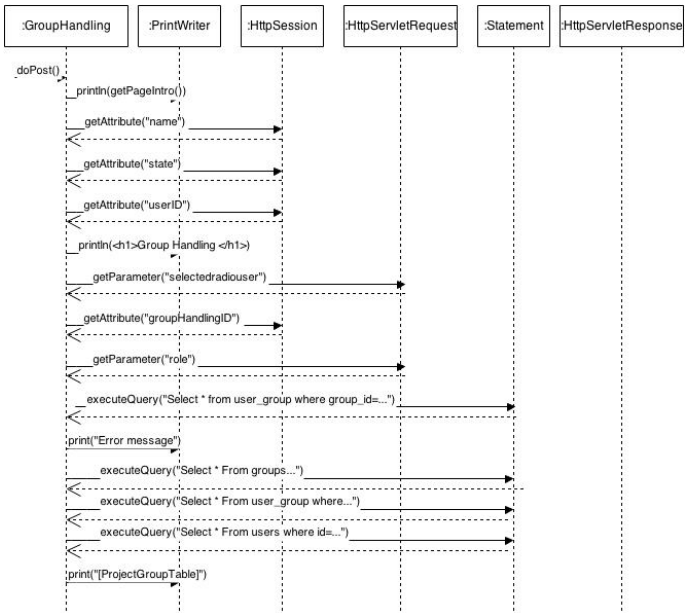
Figur 18 visar hur servleten Statistics hanterar en förfrågan om att visa en tidrapport som sammanfattar arbetet utfört i en viss aktivitet av en viss grupp/undergrupp under en specifik tid.



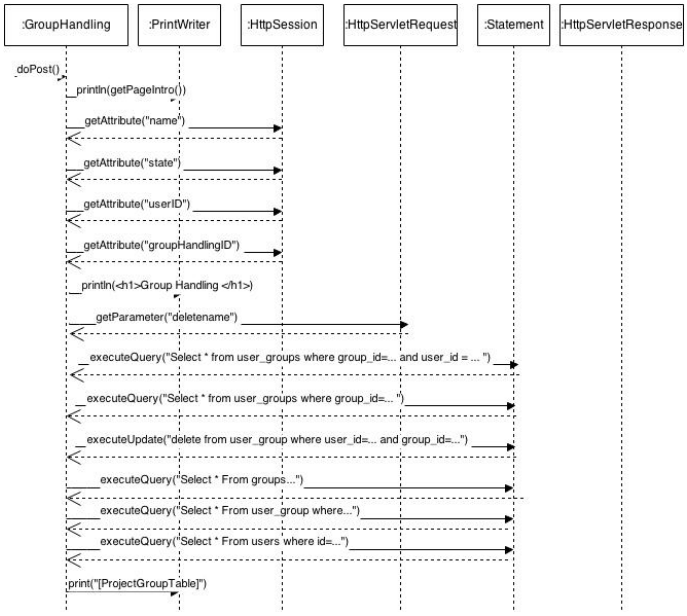
Figur 5: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp.



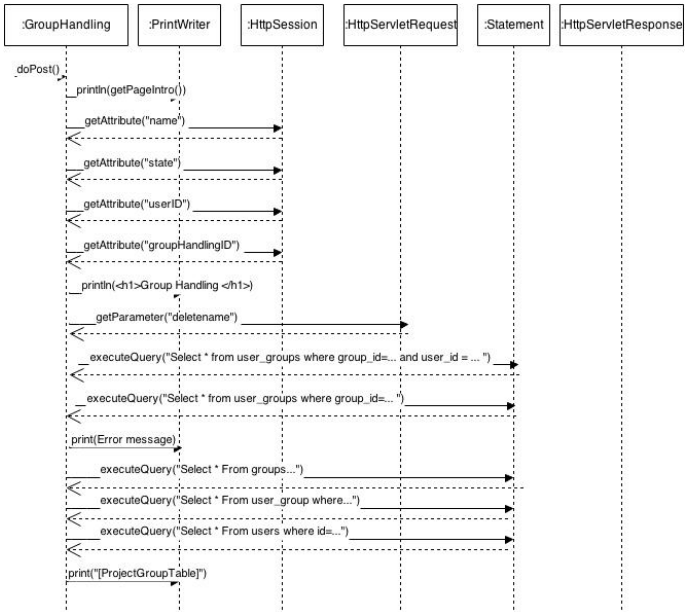
Figur 6: Sekvensdiagram som visar hur klassen GroupHandling hanterar en förfrågan om att lägga till en användare i en projektgrupp.



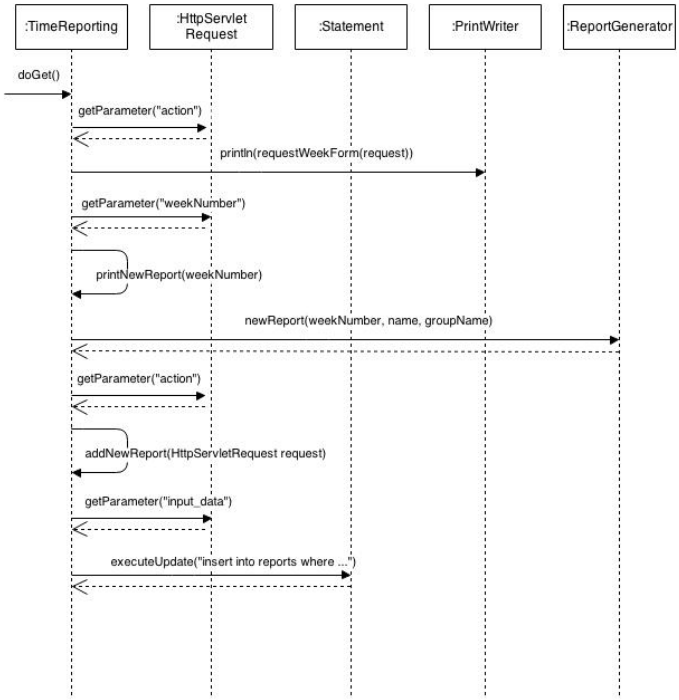
Figur 7: Sekvensdiagram som visar hur klassen GroupHandling hanterar en misslyckad förfrågan om att lägga till en användare i en projektgrupp.



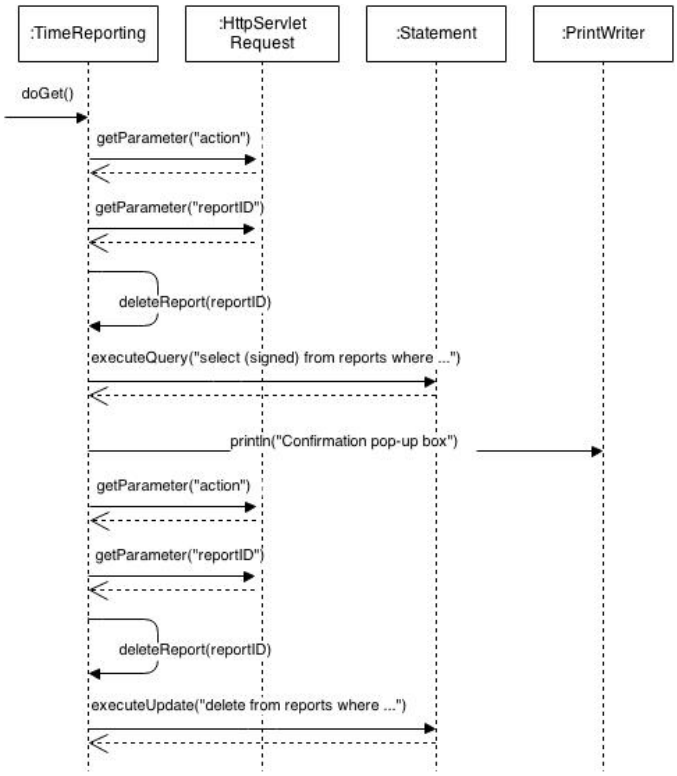
Figur 8: Sekvensdiagram som visar hur klassen GroupHandling hanterar en förfrågan om att ta bort en användare från en projektgrupp.



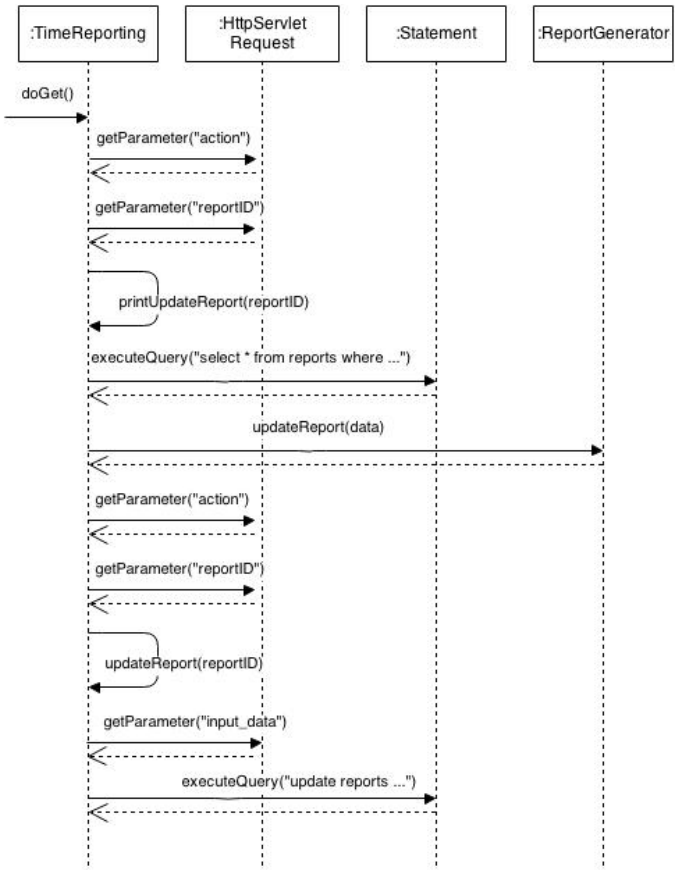
Figur 9: Sekvensdiagram som visar hur klassen GroupHandling hanterar en misslyckad förfrågan om att ta bort en användare från en projektgrupp.



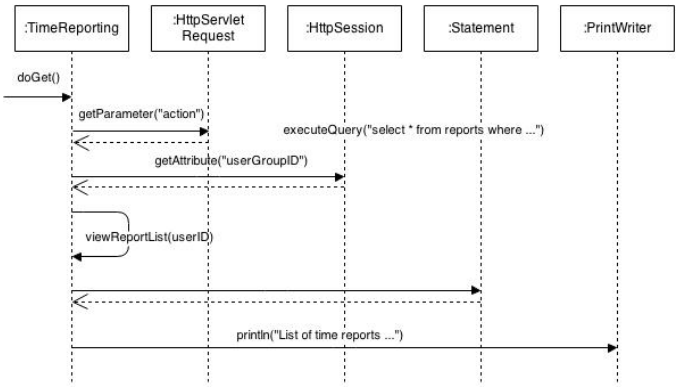
Figur 10: Visar sekvensen av de grundläggande metodanrop som sker då användaren framgångsrikt skapar en ny tidrapport i klassen TimeReporting.



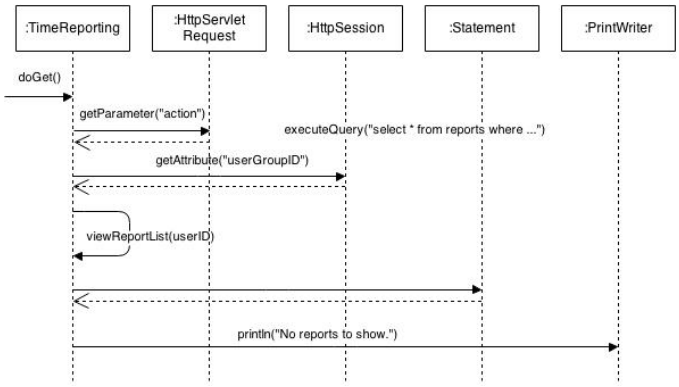
Figur 11: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt tar bort en tidrapport som inte är signerad.



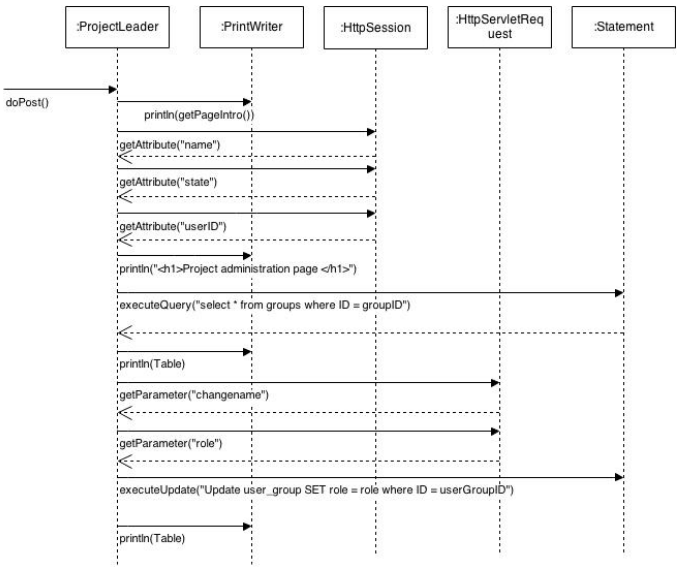
Figur 12: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt uppdaterar en tidrapport som inte är signerad.



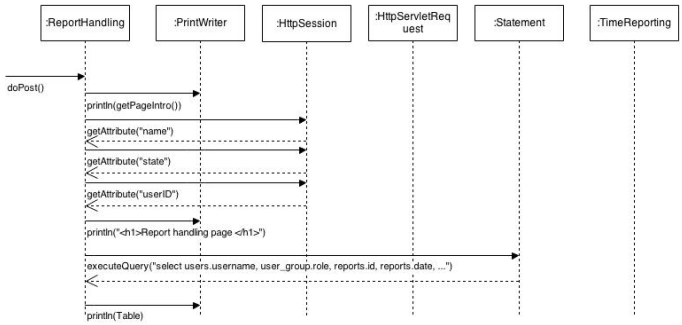
Figur 13: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt listar alla sina befintliga tidrapporter.



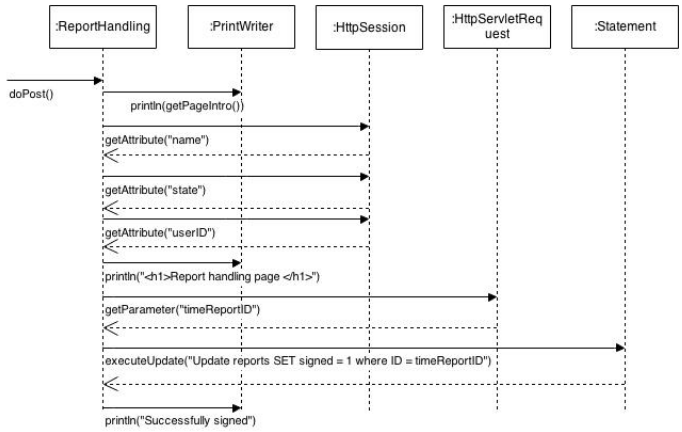
Figur 14: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker lista alla sina befintliga tidrapporter, då denne inte har några befintliga tidrapporter.



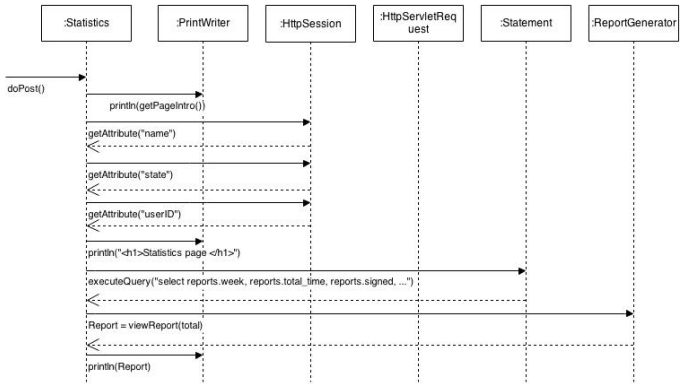
Figur 15: Sekvensdiagram som beskriver hur klassen ProjectLeader hanterar en förfrågan om att ändra en projektroll för en användare.



Figur 16: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att visa alla tidrapporter i en projektgrupp.



Figur 17: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att signera en rapport.



Figur 18: Sekvensdiagram som visar hur klassen Statistics hanterar en förfrågan om att visa en tidrapport som sammanfattar arbetet utfört av en viss grupp/undergrupp för valda veckor.