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.

1 Inledning

Dokumentet beskriver högnivådesignen för NewPussSystem tidrapporteringssystem.

2 Referensdokument

1. SRS - Software Requirements Specification (Dokumentnummer PUSS144401, version 0.21)
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 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 GroupHandling Den här servleten ärver från ServletBase och hanterar användarna i en projektgrupp.

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 ProjectGroupAdmin

private String addProjectForm()
Constructs a form for adding project groups.
@return String - The html-code for constructing the form.

private boolean addProject(String name)
Adds a new project group. Has to be followed by a successful call of addUserToGroup(userID, groupID, role) with role = project leader.
@param name: The name of the new group.
@return boolean: True if the project is added successfully.

private boolean deleteProject(int projectID)
Deletes a project group.
@param projectID: The id of the project group which will be deleted.
@return boolean: True if the group is deleted successfully.

private boolean addUserToGroup(int userID, int groupID, String role)
Adds a user with a role to a group.
@param userID: The id of the user who will be added to the group.
@param groupID: The id of the group to which the user will be added to.
@param role: The role the user will have in the group.
@ return boolean: True if the user was successfully added.

private boolean removeUserFromGroup(int userGroupID)
Removes the user from the group. Which user and group are found in the database's tabel user_group.
@param userGroupID: The ID of the row in the table user_group on which the user and group can be found.
@return boolean: True if the user was successfully removed.

3.2.3 class TimeReporting

private void viewReportList(int userID)
Prints out a list of the user's own reports.
@param userID: The id of the user.

private void viewReport(int reportID)

Fetches the data for the time report specified by the reportID parameter and passes the data on to the static method viewReport in the static ReportGenerator class.

@param reportID: The id of the time report.

private void printUpdateReport(int reportID)

Fetches the data for the time report specified by the reportID parameter and passes the data on to the static method updateReport in the static ReportGenerator class.

@param reportID: The id of the time report.

private void printNewReport(int weekNumber)

Prints out the new timereport html-form.

@param weekNumber: The weeknumber for the time report.

private void updateReport(int reportID)

Updates the data stored in the database for the report specified by the reportID parameter.

@param reportID: The id of the time report to update.

private void deleteReport(int reportID)

Deletes the report specified by the reportID parameter after confirmation and if and only if it is unsigned.

@param reportID: The id of the time report to delete.

private void addNewReport()

Inserts the data from the new report form into the database.

private void filterReportList(String filter)

Filters what time reports that should be shown in the list.

@param filter: The filter that should be applied.

3.2.4 class ReportGenerator

public static void 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.

public static void 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.

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

3.2.5 class ProjectLeader

public void showAllUsers(int groupID)
Shows a list of all the users in the system.
@param groupID: The group which members should be listed.

public boolean changeRole(int userGroupID, String role)
Assigns a role to a user in a project group.
@param userGroupID: The user to be assigned a role, and in which group.
@param role: Which role to assign.
@return boolean: True if the user was successfully added.

3.2.6 class ReportHandling

private boolean signTimeReport(int timeReportID)
Signs a time report.
@param timeReportID: The id of the report that will be signed.
@return boolean: True if the time report was successfully signed.

private boolean unsignTimeReport(int timeReportID)
Unsigns a time report.
@param timeReportID: The id of the report that will be unsigned.
@return boolean: True if the time report was succcessfully unsigned.

private void showAllReports(int group)
Shows a list of all the time reports, from the specified group with the given id.
@param group: The project leaders group id.

3.2.7 class Statistics

private boolean generateStatisticsReport(int groupID, int userID, String role, String weeks, String activity)
Shows a time report based on the parameters requested by the user.
@param groupID: The project leaders group id.
@param userID: The user for the time report.
@param role: The role of the user(s) for the time report.
@param weeks: The weeks for which the time report will be shown.
@param activity: The activity that will be included in the time report.
@return boolean: True if the report was successfully generated and shown.

private boolean generateSummarizedReport(List<String> timeReports, String activity, String subactivity)
Shows a time report with the summarized activity from several time reports.
@param timeReports: The time reports which it will summarize from.
@param activity: The activity which will be summarized.
@param subactivity: The subactivity which will be summarized.
@return boolean: True if the time report was successfully generated.

private String commonActivity(int groupID)
Finds the activity that has the most combined minutes reported by the users in the project group.
@param groupID: The project leaders group id.
@return string: Returns the activity.

private int busiestWeek(int groupID)
Finds the week with the most combined minutes reported by the users in the project group.
@param groupID: The project leaders group id.
@return int: Returns the busiest week.

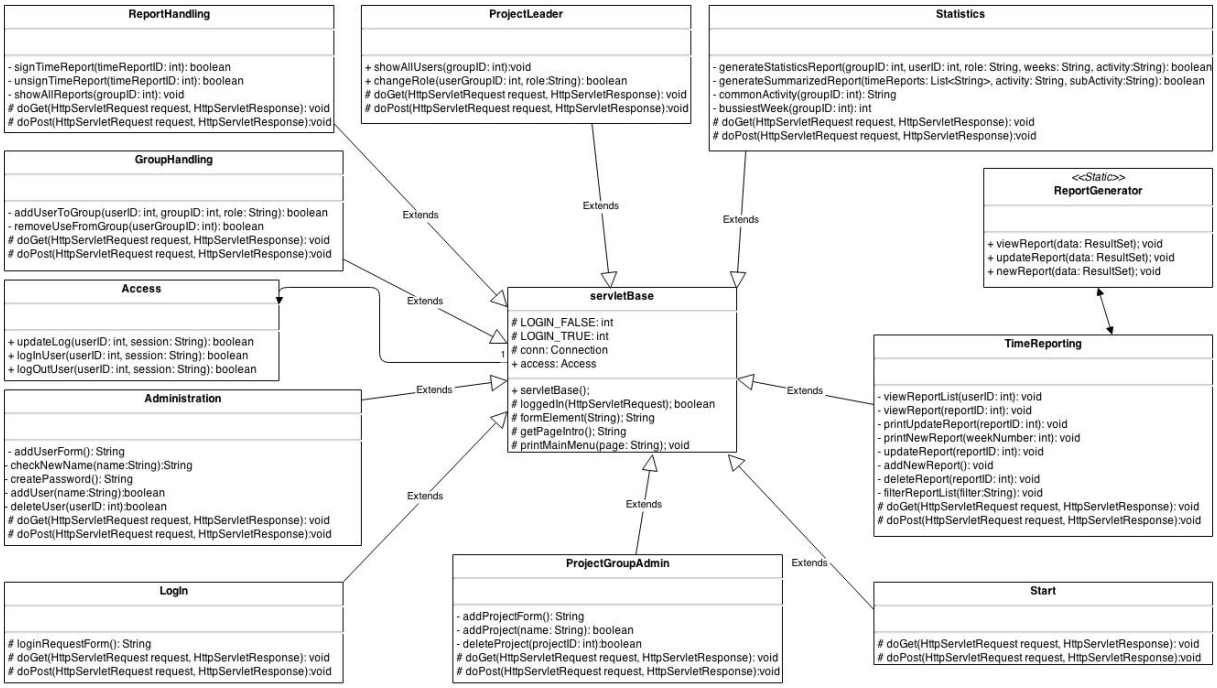
4 Klassdiagram

Ett klassdiagram med alla klasser visas i Figur 1.

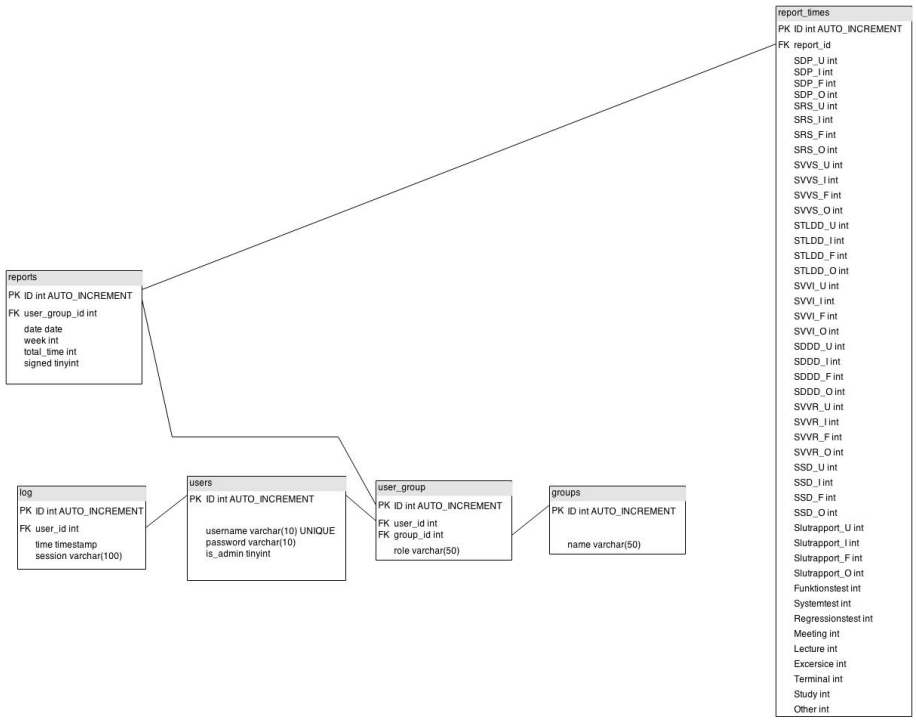
5 Databas

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

```
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,
```



Figur 1: Klassdiagram över alla klasser i systemet.



Figur 2: ER-diagram över databasen för systemet

```
    ->   `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:

Session session: Användarens sessions-id.

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

int userID: Användarens id.

int userGroupID: Id som säger vilken grupp användaren är inloggad för.

7 Sekvensdiagram

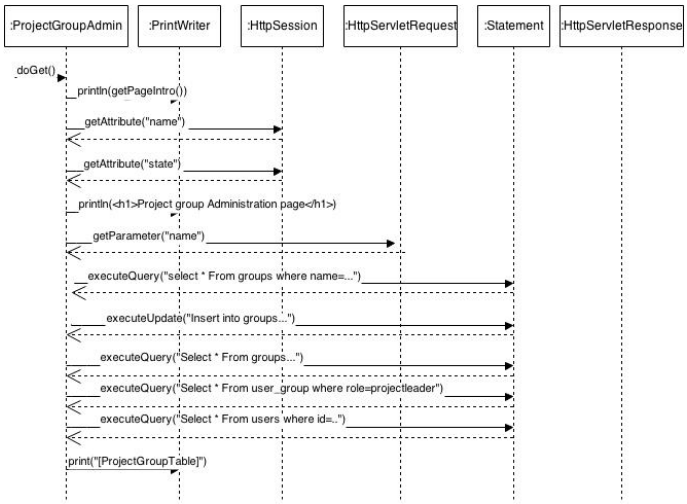
Observera att inte alla meddelanden visas i följande sekvensdiagram, utan endast de meddelanden som krävs för att förstå sekvensen.

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.

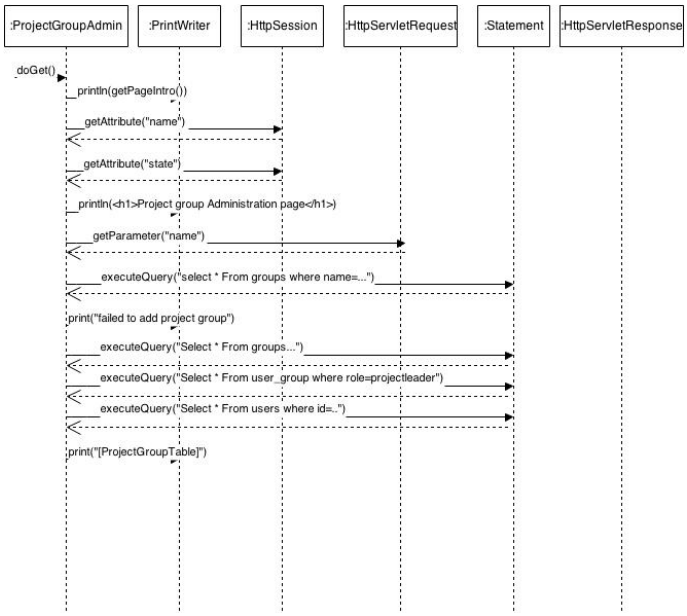


Figur 3: Sekvensdiagram som visar hur en lyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin.

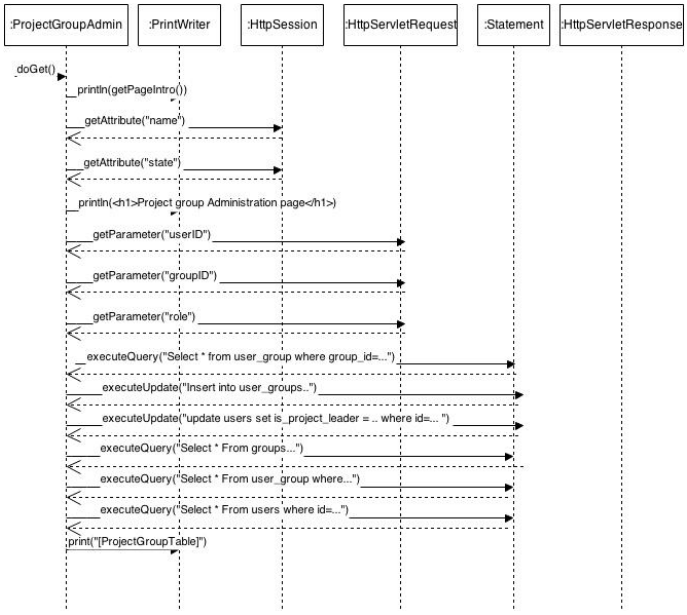
Figur 5 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att lägga till en användare i en projektgrupp, medan Figur 6 visar hur det hanteras när den misslyckas.
Figur 7 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att ta bort en användare från en projektgrupp, medan Figur 8 visar hur det hanteras när den misslyckas.
Figur 9 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp.

7.3 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 och Figur 12 visar hur det hanteras när den misslyckas.
Figur 13 visar hur servleten TimeReporting hanterar en förfrågan om att uppdatera en tidrapport som redan blivit signerad och Figur 14 visar hur en förfrågan om att uppdatera en tidrapport som ännu inte blivit signerad hanteras.

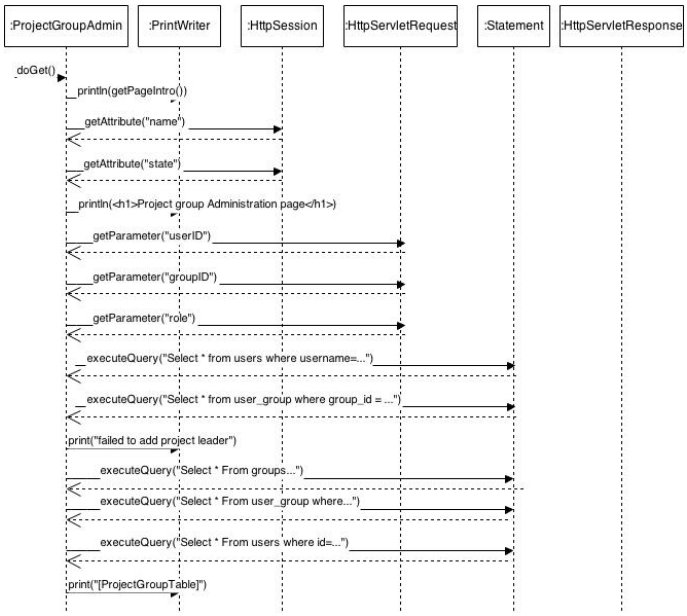


Figur 4: Sekvensdiagram som visar hur en misslyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin.

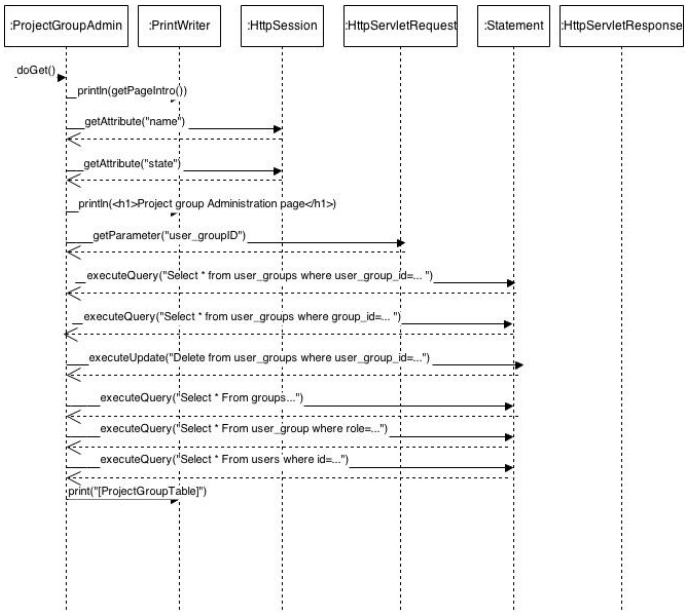


Figur 5: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en förfrågan om att lägga till en användare i en projektgrupp.

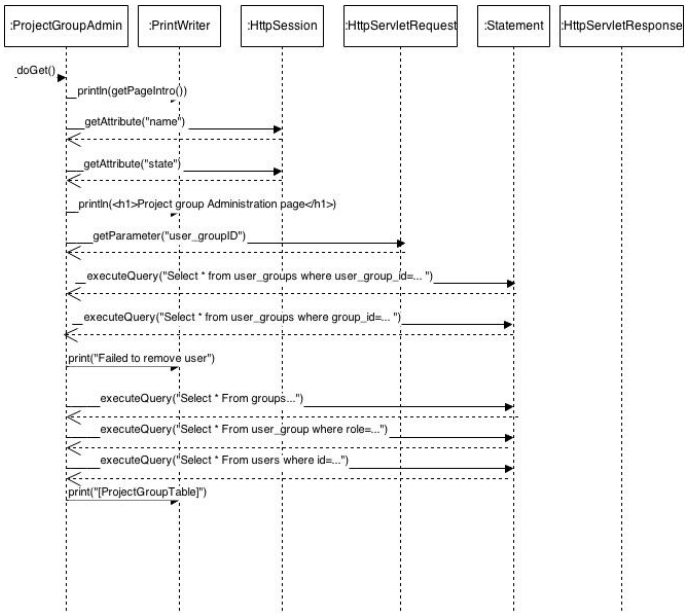
Figur 15 visar hur servleten TimeReporting hanterar en förfrågan om att skriva ut en användares alla tidrapporter, Figur 16 visar hur den hanteras när den misslyckas.



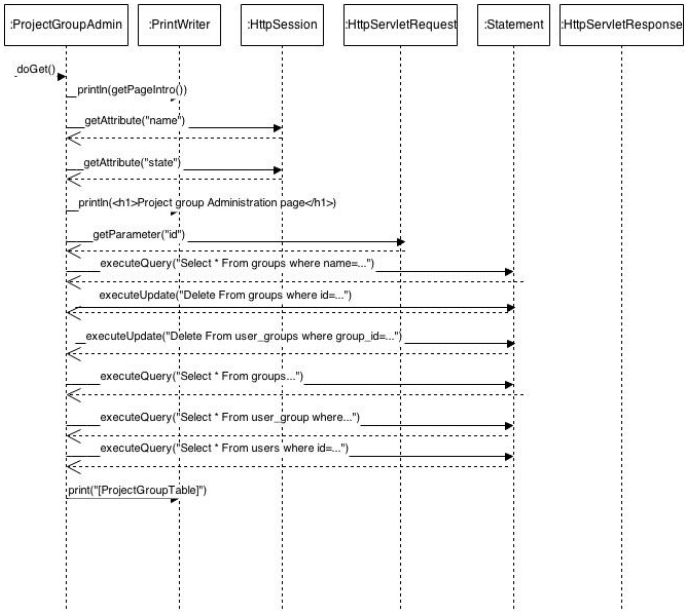
Figur 6: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en misslyckad förfrågan om att lägga till en användare i en projektgrupp.



Figur 7: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en förfrågan om att ta bort en användare från en projektgrupp.



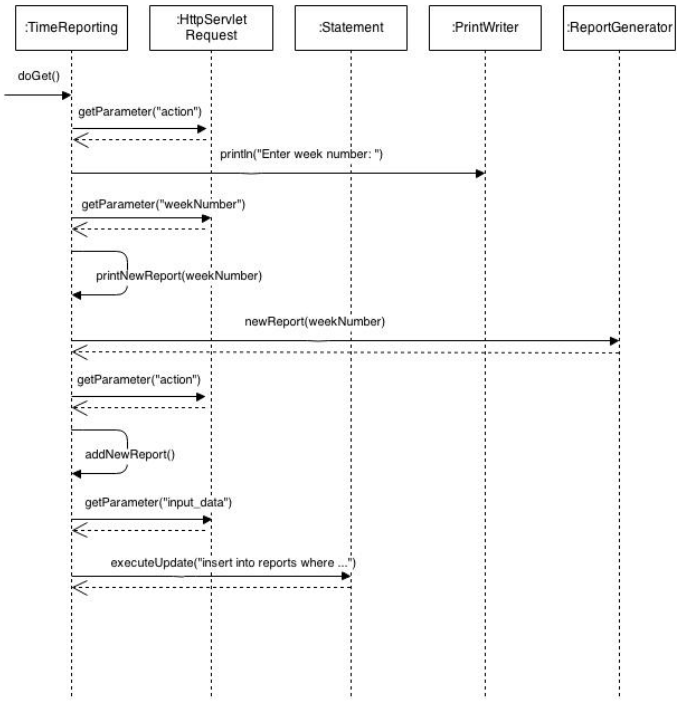
Figur 8: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en misslyckad förfrågan om att ta bort en användare från en projektgrupp.



Figur 9: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp.

7.4 ProjectLeader

Figur 17 visar hur servleten ProjectLeader hanterar en förfrågan om att ändra en projektroll för en användare.



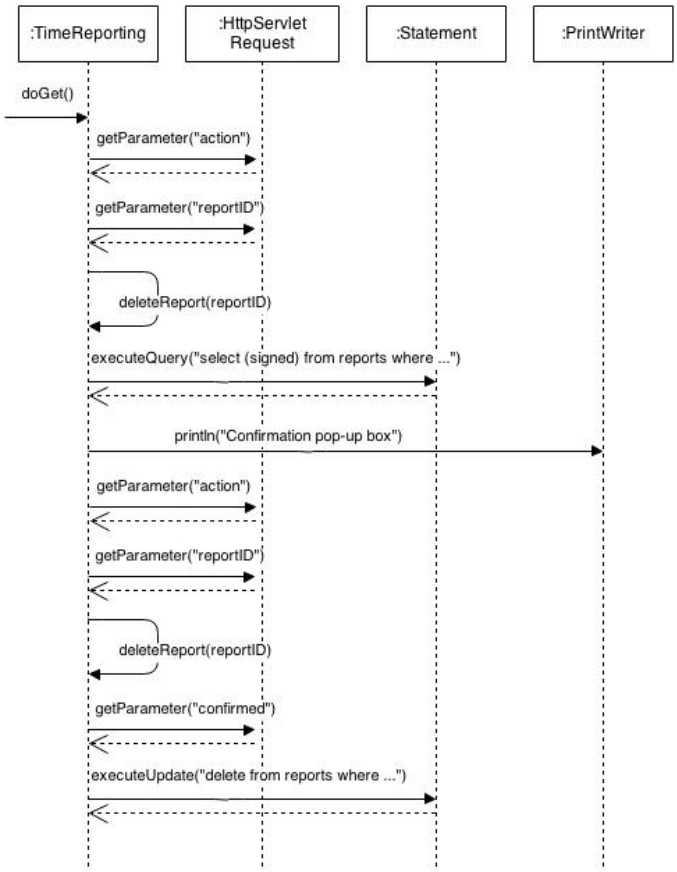
Figur 10: Visar sekvensen av de grundläggande metodanrop som sker då användaren framgångsrikt skapar en ny tidrapport i klassen TimeReporting.

7.5 ReportHandling

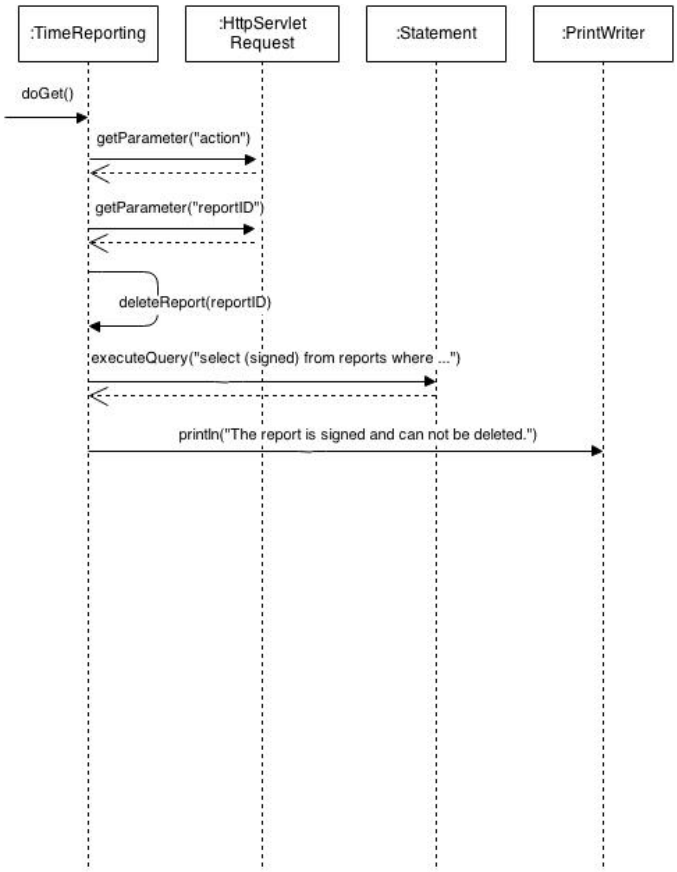
Figur 18 visar hur servleten ReportHandling hanterar en förfrågan om att visa alla tidrapporter i projektgruppen, signerade samt osignerade. Figur 19 visar hur servleten ProjectLeader hanterar en förfrågan om att signera en rapport.

7.6 Statistics

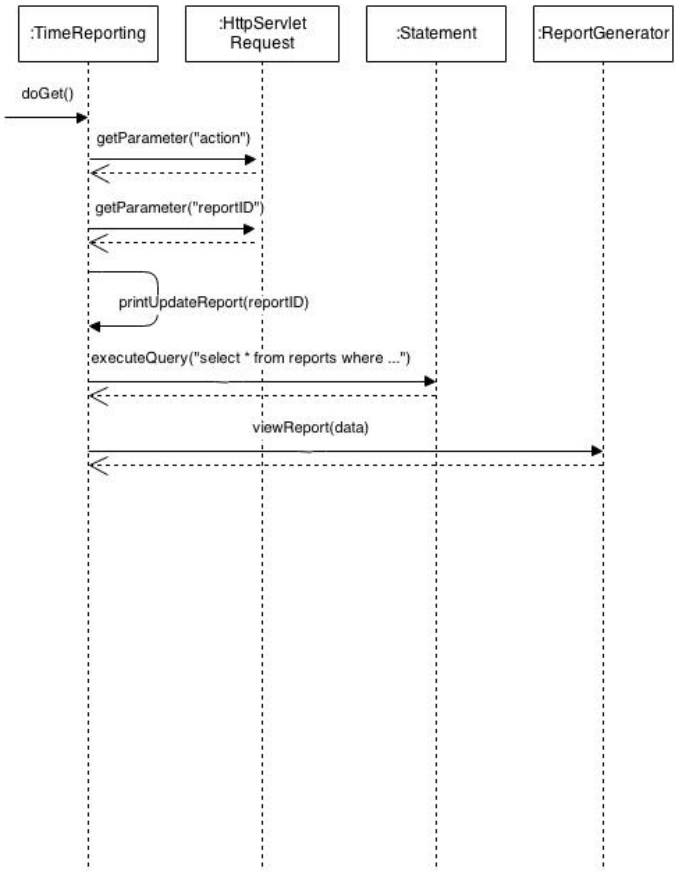
Figur 20 visar hur servleten Statistics hanterar en förfrågan om att visa vilken vecka som har flest totala inlagda minuter.
Figur 21 visar hur servleten Statistics hanterar en förfrågan om att visa vilken aktivitet som har flest totala inlagda minuter.
Figur 22 visar hur servleten Statistics hanterar en förfrågan om att visa en tidrapport som sammanfattar arbetet utfört i en viss aktivitet av en viss grupp/undergrupp under en specifik tid.



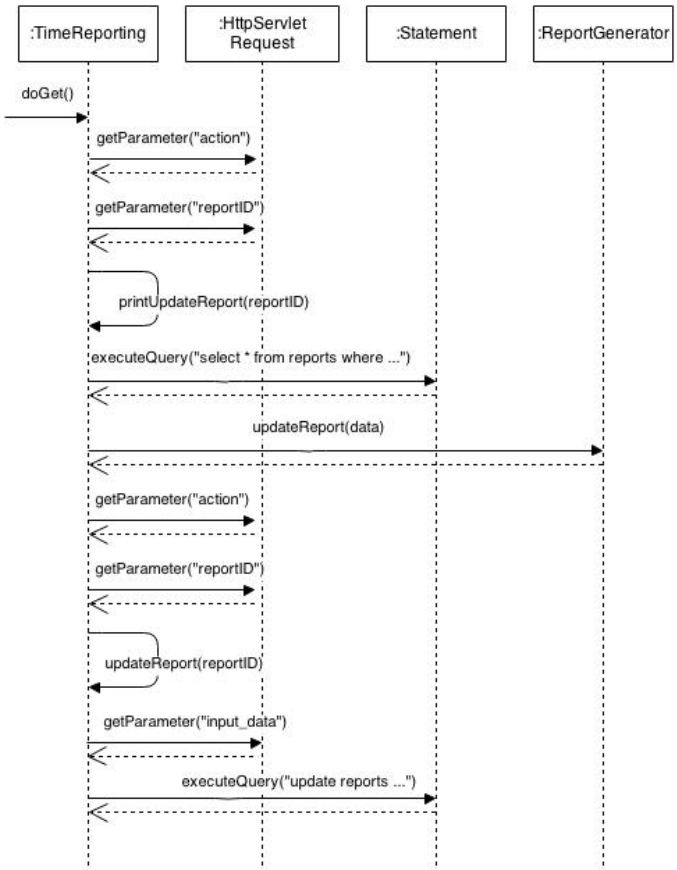
Figur 11: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt tar bort en tidrapport som inte är signerad.



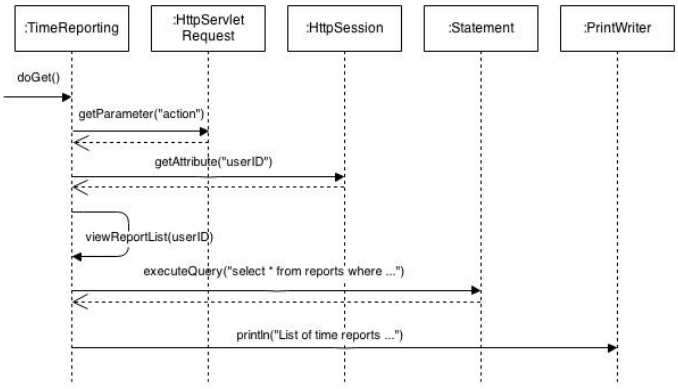
Figur 12: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker ta bort en tidrapport som är signerad.



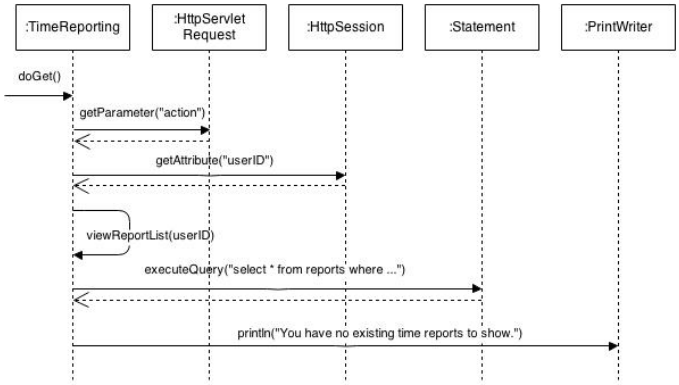
Figur 13: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker uppdatera en tidrapport som är signerad.



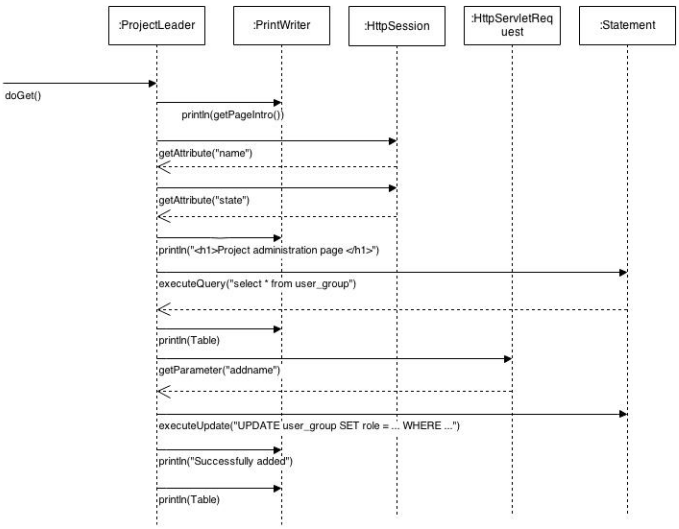
Figur 14: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt uppdaterar en tidrapport som inte är signerad.



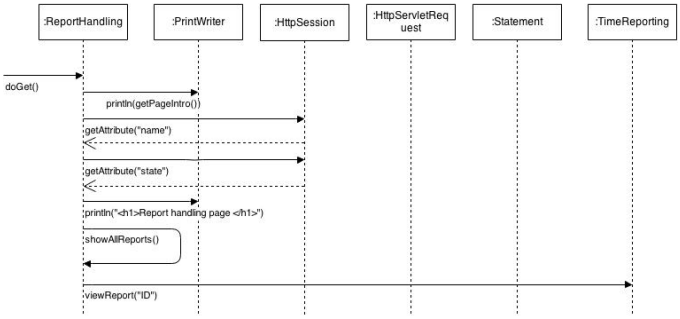
Figur 15: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt listar alla sina befintliga tidrapporter.



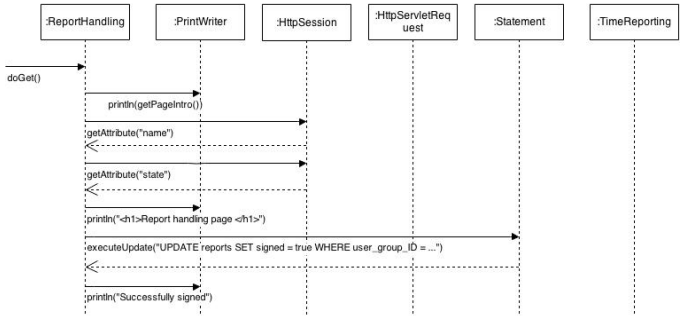
Figur 16: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker lista alla sina befintliga tidrapporter, då denne inte har några befintliga tidrapporter.



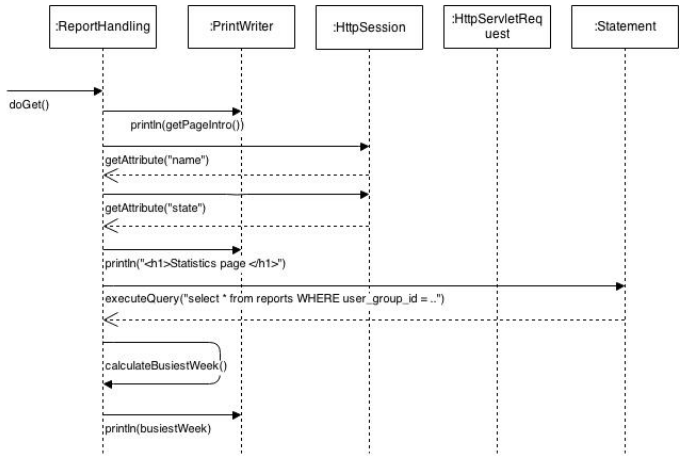
Figur 17: Sekvensdiagram som beskriver hur klassen ProjectLeader hanterar en förfrågan om att ändra en projektroll för en användare.



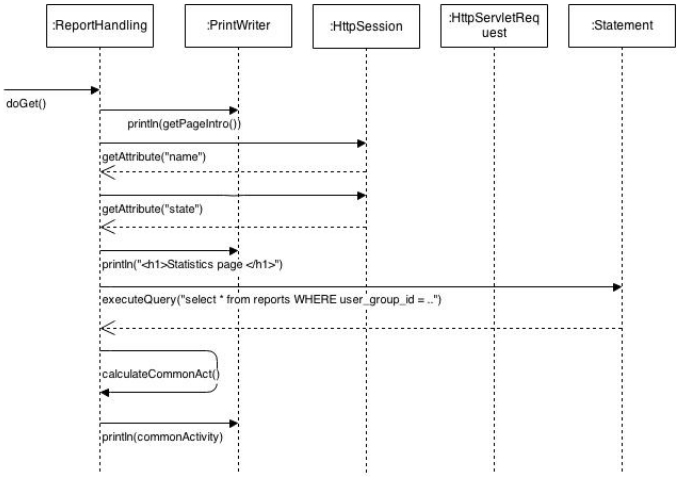
Figur 18: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att visa alla tidrapporter i en projektgrupp.



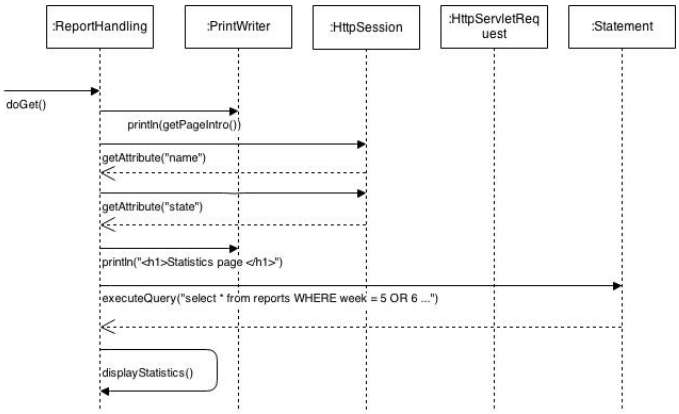
Figur 19: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att signera en rapport.



Figur 20: Sekvensdiagram som visar hur en förfrågan om att visa vilken vecka som har flest totala inlagda minuter hanteras i klassen Statistics.



Figur 21: Sekvensdiagram som beskriver hur en förfrågan om att visa vilken aktivitet som har flest totala inlagda minuter hanteras i klassen Statistics.



Figur 22: Sekvensdiagram som visar hur klassen 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.