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.

1 Inledning

Dokumentet beskriver högnivådesignen för NewPussSystem, ett tidrapporteringssystem för projekt som diverse användare kan logga in på.

2 Referensdokument

1. SRS - Software Requirements Specification (Dokumentnummer PUSS144401, version 0.20)
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 dokumentet STLDD - Software Top Level Design Document: BaseBlockSystem.

class Administration Se sidan 2 i dokumentet STLDD - Software Top Level Design Document: BaseBlockSystem. En ändring har gjorts där metoden deleteUser ska returnera en boolean och inparametern ändras till (int id).

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, tidrapportmall). 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 (ny sida), 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 submeny med specifika menyval relaterade till tidrapportering. Förutom submenyn kommer första sidan att vara tom. När ett val i submenyn 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 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 automatiskt en lista över alla användare. Härifrån kan projektledaren antingen välja att hantera användaren i listan (lägga till, ta bort ur grupp) eller välja andra funktioner.

class ReportHandling Den här servleten ärver från ServletBase och visar 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.

3.2 Klassmetoder

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

3.2.1 class ProjectGroupAdmin

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

private boolean addProject(String name, String leader, String timeReport)
Adds a new project group.
@param name: The name of the new project group.
@param leader: The name of the user that will be added as project leader to the project group.
@param timeReport: The time report template which will be used for the project group.
@return boolean: True if the group is added succesfully, else false.

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

private boolean addProjectLeader(String leaderName, String groupName)
Adds a project leader to a selected project group.
@param leaderName: The name of the user which will be added as project leader.
@param groupName: The project group to which the project leader will be added.
@return boolean: True if the project leader is added succesfully, else false.

private boolean removeProjectLeader(String leaderId, String groupId)

Removes a project leader from a selected project group.
@param leaderId: The id of the user which will be removed as project leader.
@param groupId: The id of the project group from which the project leader will be removed.
@return boolean: True if the project leader is removed succesfully, else false.

3.2.2 class TimeReporting

protected void printSubMenu()

Prints out a sub menu including the following option: view time reports and new time report.

protected void viewReportList()

Prints out a list of the user's own reports.

protected void viewReport(String reportID)

Fetches the data for the time report specified by the String parameter and passes the data on to the static method viewReport in the static ReportGenerator class.
@param reportID: The id of the time report.

protected void printUpdateReport(String reportID)

Fetches the data for the time report specified by the String parameter and passes the data on to the static method updateReport in the static ReportGenerator class.
@param string: The id of the time report.

protected void printNewReport(int weekNumber)

Fetches the correct template from the database and passes it on to the static method newReport in the ReportGenerator class.
@param weekNumber: The weeknumber for the time report.

protected void updateReport(String reportID)

Updates the data stored in the database for the report specified by the String parameter.
@param string: The id of the time report to update.

protected void deleteReport(String reportID)

Deletes the report specified by the String parameter after confirmation and if and only if it is unsigned.
@param reportID: The id of the time report to delete.

protected void addNewReport()

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

protected void filterReportList(String filter)

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

@param filter: The filter that should be applied.

3.2.3 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.

3.2.4 class ProjectLeader

public void showAllUsers()
Shows a list of all the users in the system.

public boolean assignRoleToUser(String user, String role)
Assigns a role to a user in a project group.
@param user: The user to be assigned a role.
@param role: Which role to assign.
@return boolean: True if the user was succesfully added, else false.

public boolean removeUserFromProject(String user)
Removes a user from a project group.
@param user: The user to be removed.
@return boolean: True if the user was successfully removed, else false.

3.2.5 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, else false.

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, else false.

private void showAllReports()
Shows a list of all the time reports.

3.2.6 class Statistics

private boolean generateStatisticsReport(String user, String role, String weeks, String activity)
Shows a time report based on the parameters requested by the user.
@param user: 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, else false.

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, else false.

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

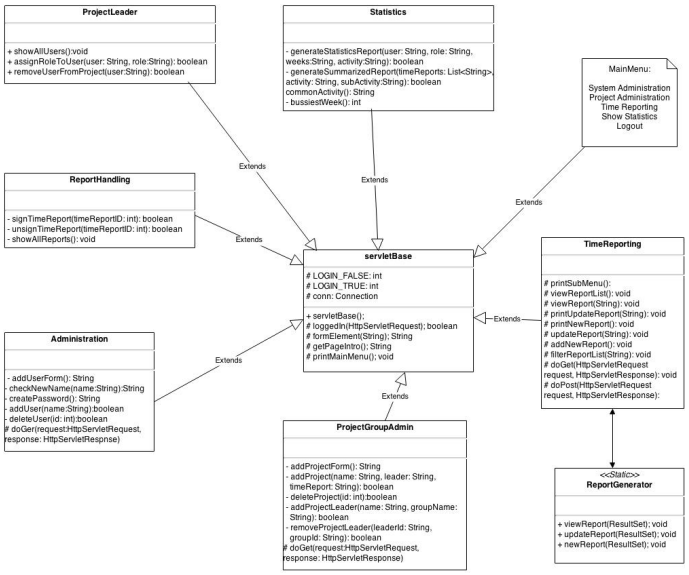
private int busiestWeek()
Finds the week with the most combined minutes reported by the users in the project group.
@return int: Returns the busiest week.

4 Klassdiagram

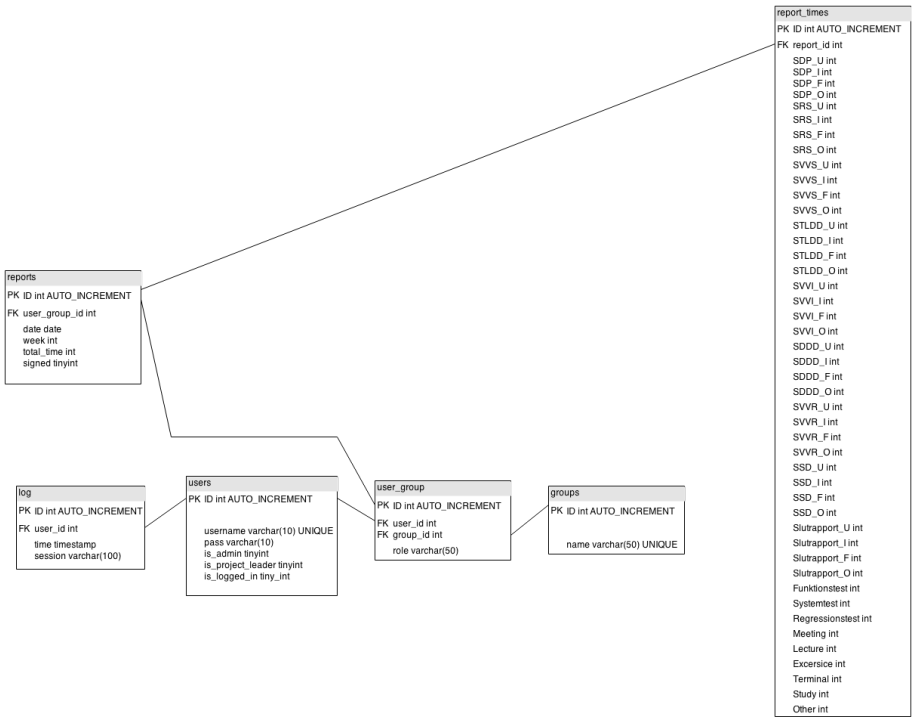
Ett klassdiagram med alla klasser visas i Figur 1.

5 Databas

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



Figur 1: Klassdiagram över alla klasser i systemet.



Figur 2: ER-diagram över databasen för systemet

```
mysql> create database base;
mysql> use base;
mysql> create table users (id int AUTO_INCREMENT, username varchar(10) UNIQUE,
```

```
    -> password varchar(10), is_admin tinyint, is_project_leader tinyint,
    -> is_logged_in tiny_int);
mysql> create table groups (id int AUTO_INCREMENT, name varchar(50) UNIQUE);
mysql> create table user_group (id int AUTO_INCREMENT, user_id int, group_id int,
    ->  role varchar(50), PRIMARY KEY (id), INDEX user_ind (user_id),
    -> INDEX group_ind (group_id), FOREIGN KEY (user_id) REFERENCES users(id) ON
    -> DELETE RESTRICT, FOREIGN KEY (group_id) REFERENCES groups(id) ON
    -> DELETE RESTRICT);
mysql> create table log (id int AUTO_INCREMENT, user_id int, time timestamp,
    -> session varchar(100), INDEX userlog_ind (user_id), FOREIGN KEY (user_id)
    -> REFERENCES users(id) ON DELETE RESTRICT);
mysql> create table reports (id int AUTO_INCREMENT, user_group_id int,
    -> date date, week int, total_time int, signed tinyint, PRIMARY KEY (id),
    -> INDEX usergroup__ind (user_group_id), FOREIGN KEY (user_group_id)
    -> REFERENCES user_group(id) ON DELETE RESTRICT);
mysql> create table times (id int AUTO_INCREMENT, report_id int,
    -> SDP_U int, SDP_I int, SDP_F int, SDP_O int, SRS_U int, SRS_I int,
    -> SRS_F int, SRS_O int, SVVS_U int, SVVS_I int, SVVS_F int, SVVS_O int,
    -> STLDD_U int, STLDD_I int, STLDD_F int, STLDD_O int, SVVI_U int,
    -> SVVI_I int, SVVI_F int, SVVI_O int, SDDD_U int, SDDD_I int, SDDD_F int,
    -> SDDD_O int, SVVR_U int, SVVR_I int, SVVR_F int, SVVR_O int, SSD_U int,
    -> SSD_I int, SSD_F int, SSD_O int, Slutrapport_U int, Slutrapport_I int,
    -> Slutrapport_F int, Slutrapport_O int, Funktionstest int, Systemtest int,
    -> Regressionstest int, Meeting int, Lecture int, Excersice int, Terminal int,
    -> Study int, Other int, PRIMARY KEY (id), INDEX timereport_ind (report_id),
    -> FOREIGN KEY (report_id) REFERENCES reports(id) ON DELETE RESTRICT);
mysql> insert into users (username, password, is_admin, is_project_leader,
    -> is_logged_in) values('admin', 'adminpw', 1, 0, 0);
```

6 Information lagrad i sessioner

I en pågående session sparas följande attribut i sessionen:

Boolean tillstånd: används för att beskriva om användaren är inloggad eller inte. Följande två tillstånd har definierats:
true inloggad
false utloggad

Timestamp log En tidsstämpel som indikerar när användaren senast gjorde en förfrågan i systemet.

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

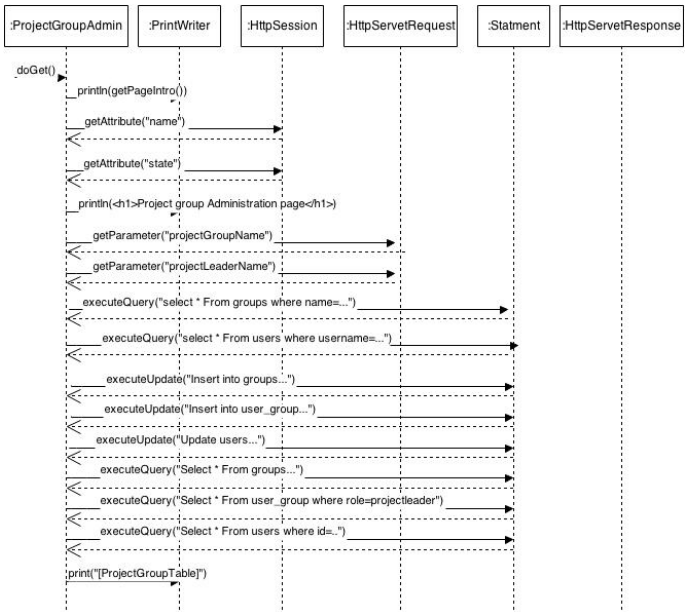
7 Sekvensdiagram

7.1 class Administration

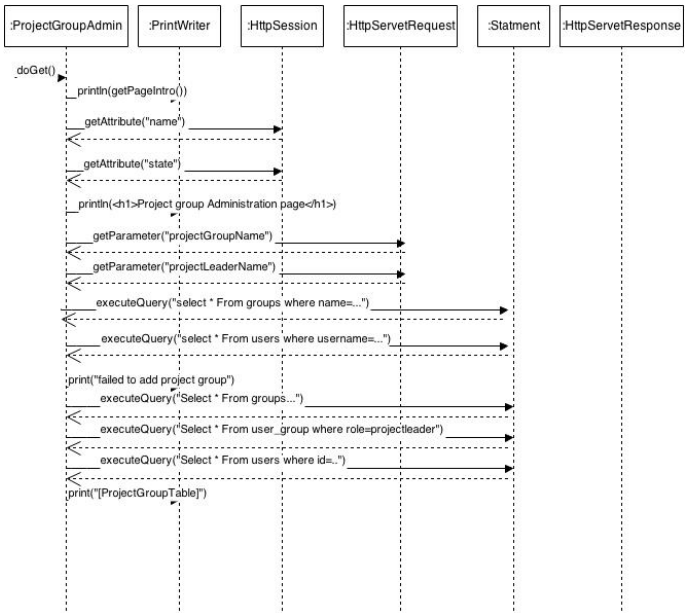
Figur 2 i dokumentet STLDD - Software Top Level Design Document: BaseBlockSystem visar sekvensen för hur servleten Administration hanterar att administratören lägger till en ny användare.

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 att lägga till en projektledare i ett projekt och Figur 6 visar hur det hanteras när den misslyckas, medans Figur 7 samt Figur 8 visar hur servleten ProjectGroupAdmin hanterar en förfrågan att ta bort en projektledare då den lyckas samt då den misslyckas.
Figur 9 visar hur servleten ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp.



Figur 3: Sekvensdiagram som visar hur en lyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



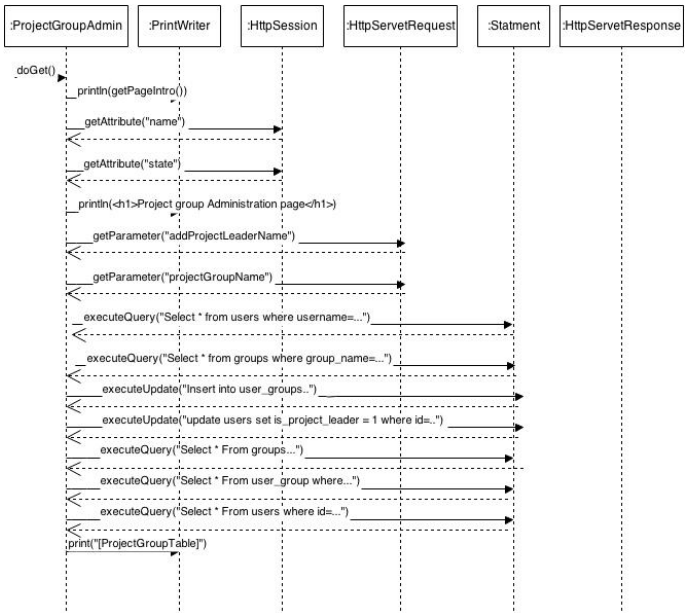
Figur 4: Sekvensdiagram som visar hur en misslyckad förfrågan om att lägga till en ny projektgrupp hanteras i klassen ProjectGroupAdmin. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)

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 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.

7.4 ProjectLeader

Figur 17 visar hur servleten ProjectLeader hanterar en förfrågan om att tilldela en projektroll till en användare. Figur 18 visar hur servleten ProjectLeader hanterar en förfrågan om att ta bort en användare från en projektgrupp.



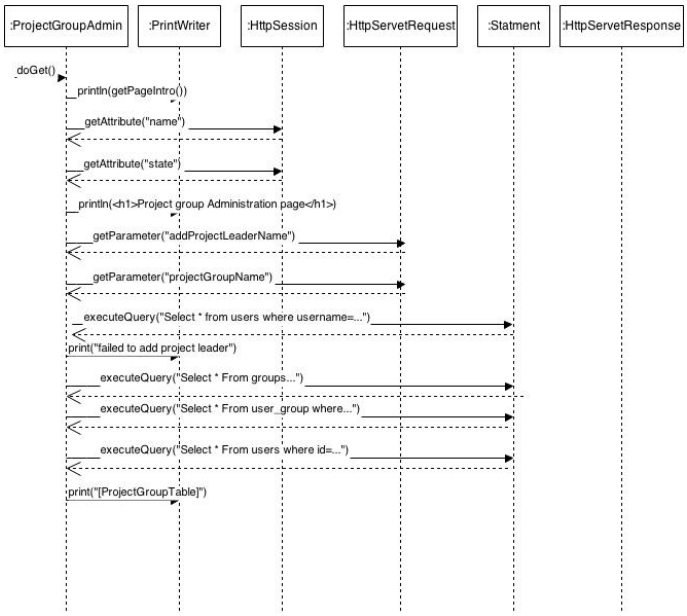
Figur 5: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en lyckad föfrågan om att lägga till en projektledare i en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)

7.5 ReportHandling

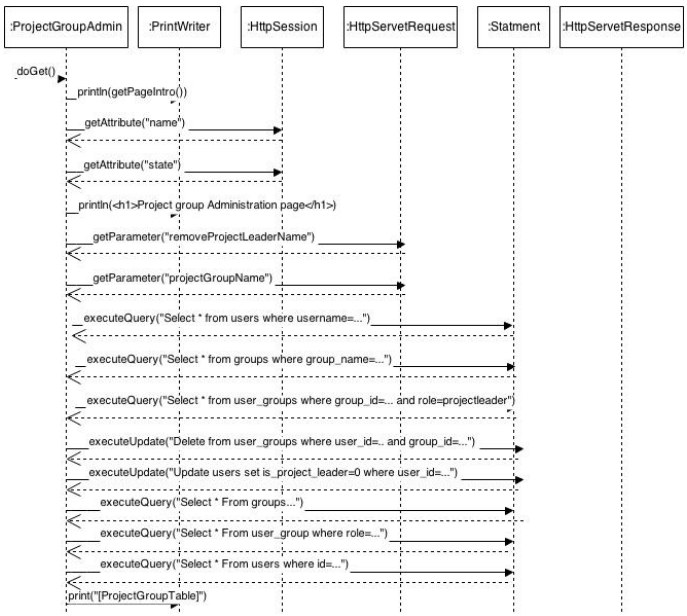
Figur 19 visar hur servleten ReportHandling hanterar en förfrågan om att visa alla tidrapporter i projektgruppen, signerade samt osignerade. Figur 20 visar hur servleten ProjectLeader hanterar en förfrågan om att signera en rapport.

7.6 Statistics

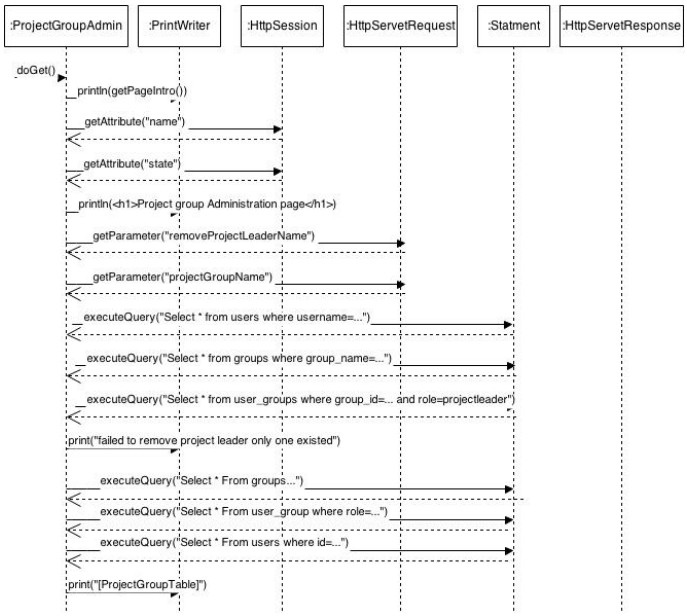
Figur 21 visar hur servleten Statistics hanterar en förfrågan om att visa vilken vecka som har flest totala inlagda minuter.
Figur 22 visar hur servleten Statistics hanterar en förfrågan om att visa vilken aktivitet som har flest totala inlagda minuter.
Figur 23 visar hur servleten Statistics hanterar en förfrågan om att visa en tidrapport som sammanfattar arbetet utfört i en viss aktivitet av en viss grupp/undergrupp under en specifik tid.



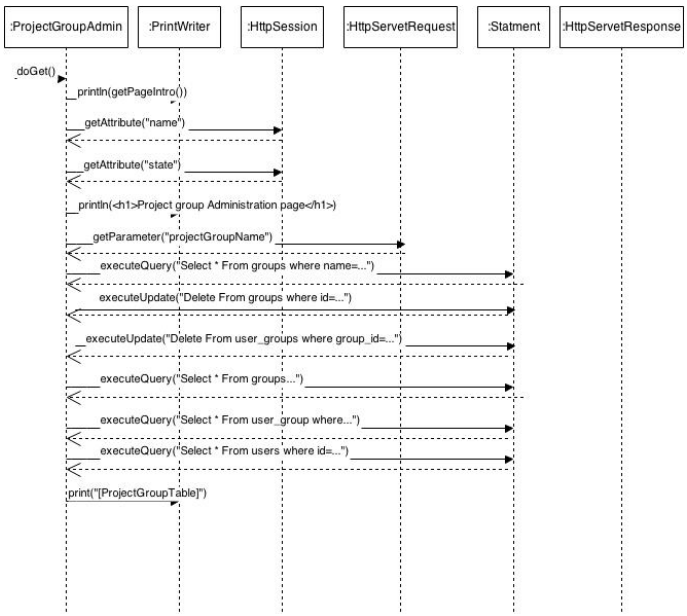
Figur 6: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en misslyckad föfrågan om att lägga till en projektledare i en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



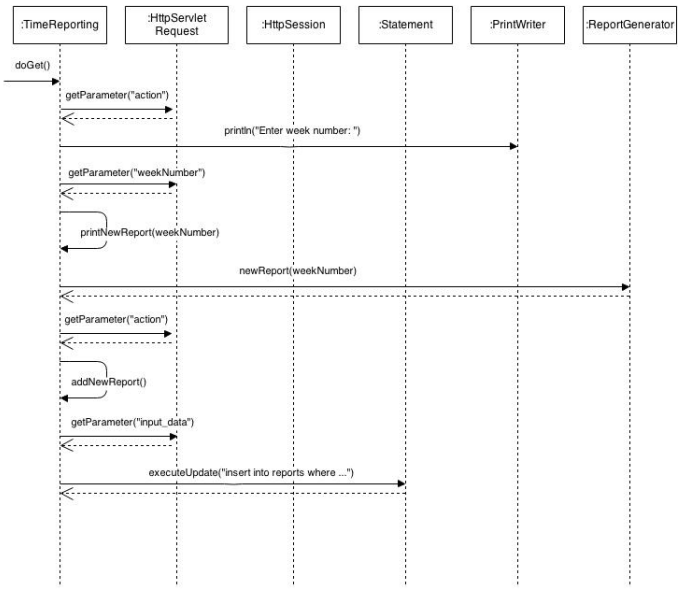
Figur 7: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en lyckad föfrågan om att ta bort en projektledare i en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



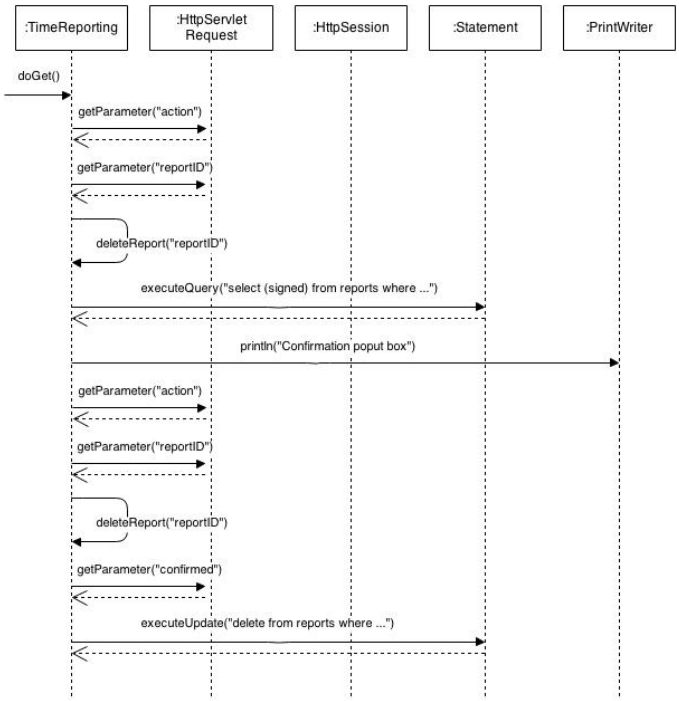
Figur 8: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en misslyckad föfrågan om att ta bort en projektledare i en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



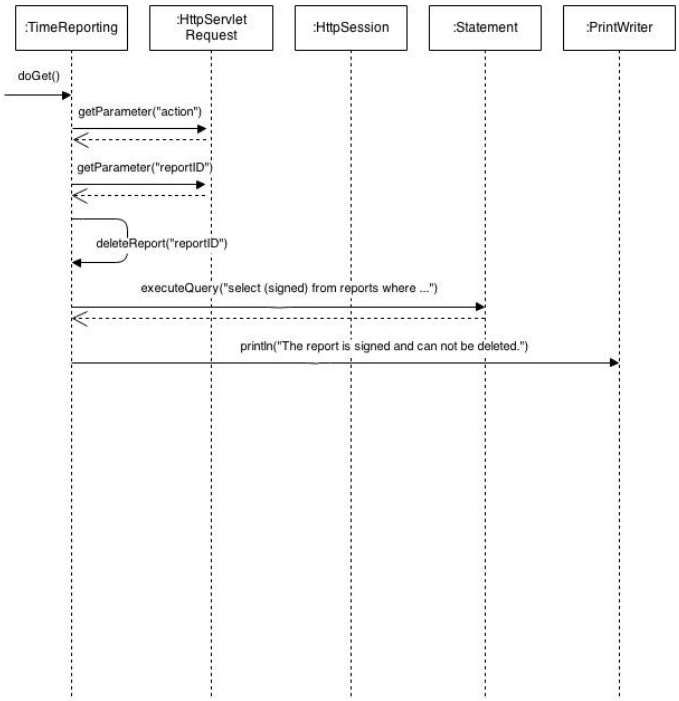
Figur 9: Sekvensdiagram som visar hur klassen ProjectGroupAdmin hanterar en förfrågan om att ta bort en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



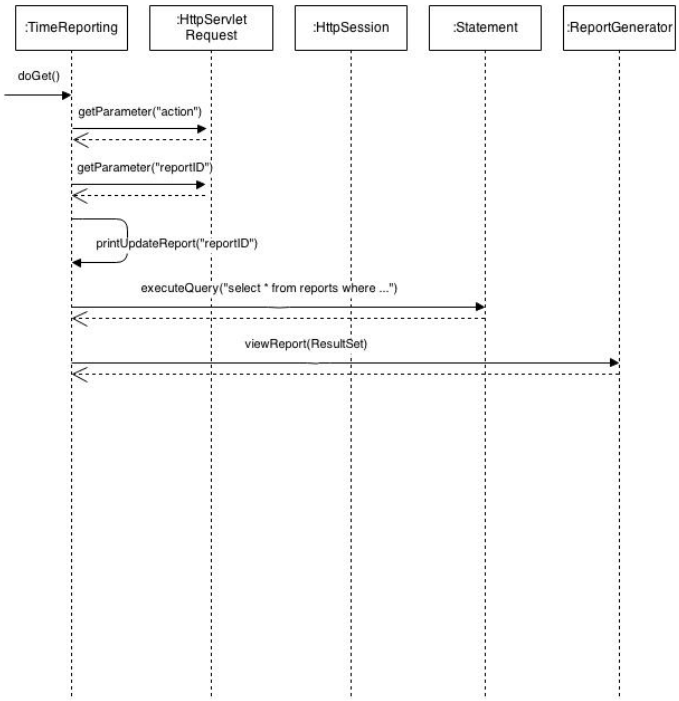
Figur 10: Visar sekvensen av de grundläggande metodanrop som sker då användaren framgångsrikt skapar en ny tidrapport i klassen TimeReporting. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



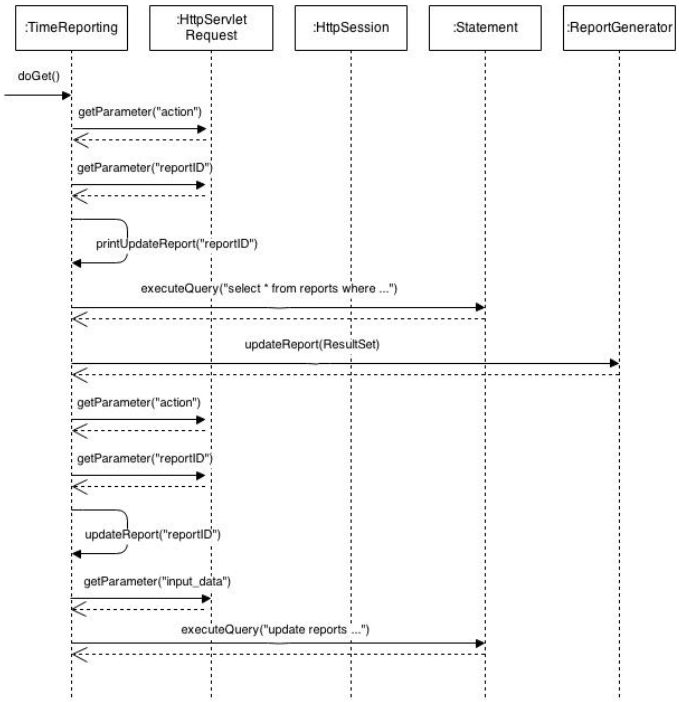
Figur 11: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt tar bort en tidrapport som inte är signerad. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



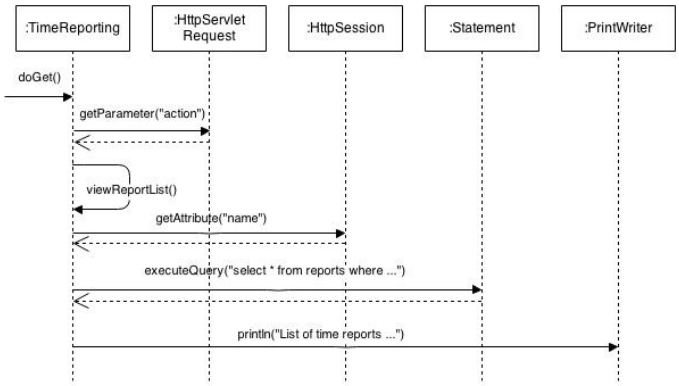
Figur 12: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker ta bort en tidrapport som är signerad. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



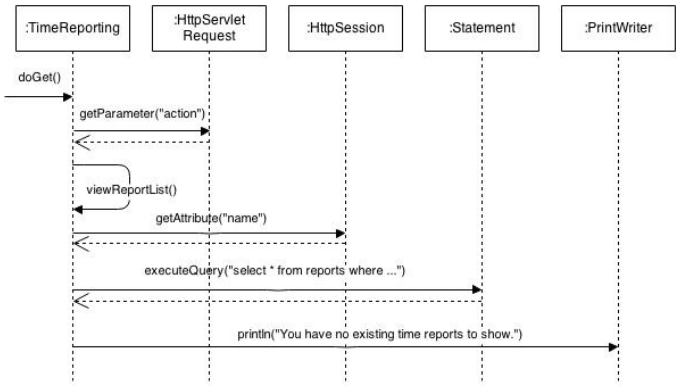
Figur 13: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker uppdatera en tidrapport som är signerad. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



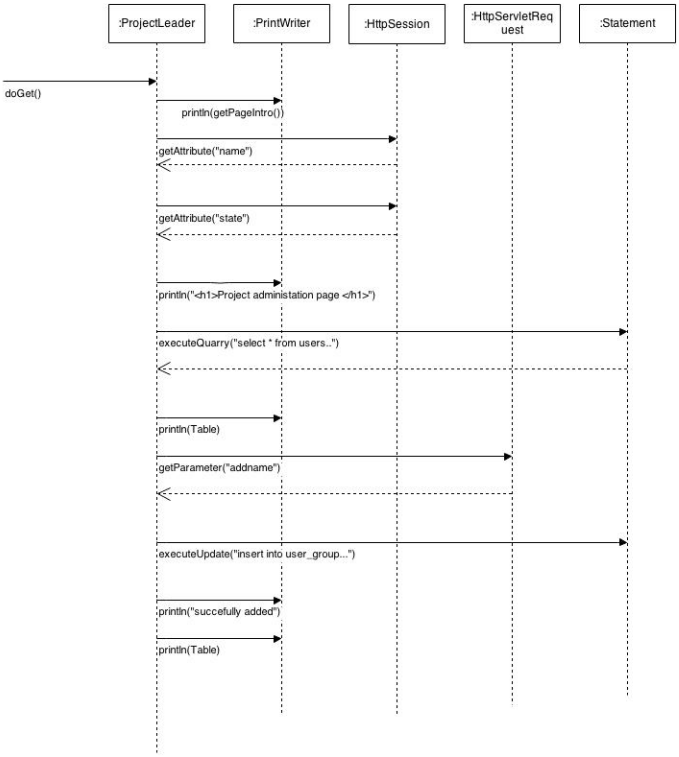
Figur 14: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt uppdaterar en tidrapport som inte är signerad. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



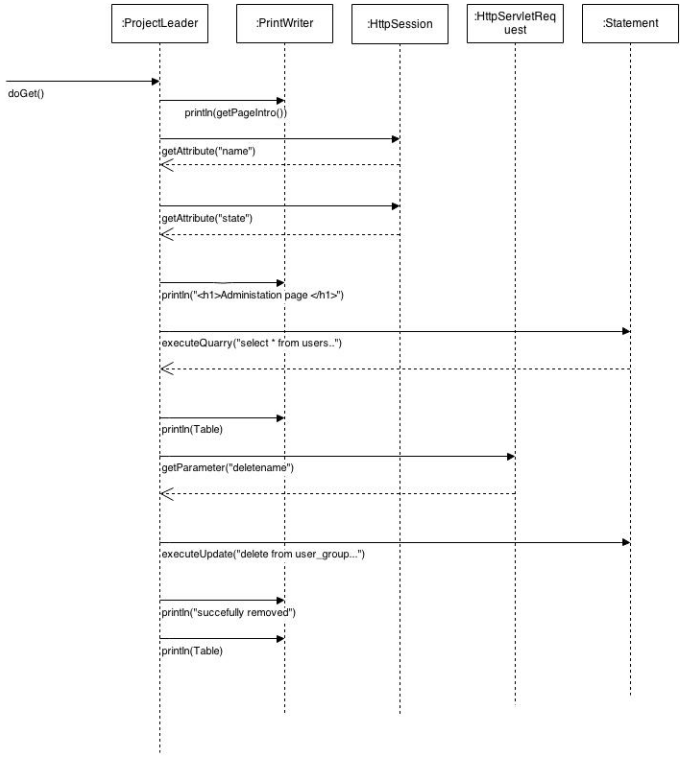
Figur 15: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren framgångsrikt listar alla sina befintliga tidrapporter. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



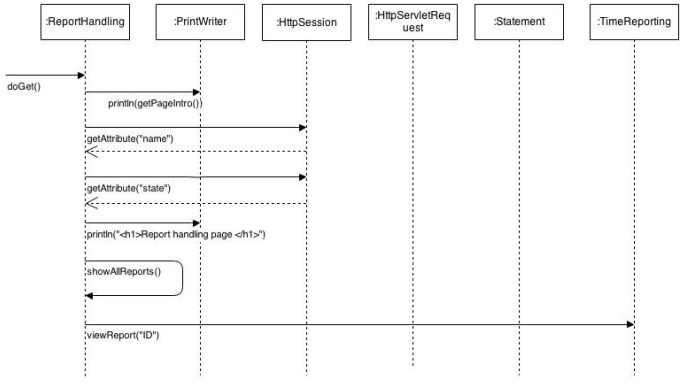
Figur 16: Visar sekvensen av de grundläggande metodanrop som sker i klassen TimeReporting då användaren misslyckat försöker lista alla sina befintliga tidrapporter, då denne inte har några befintliga tidrapporter. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



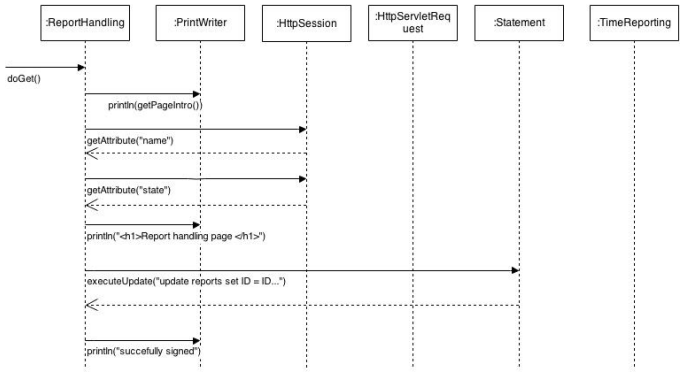
Figur 17: Sekvensdiagram som beskriver hur klassen ProjectLeader hanterar en förfrågan om att tilldela en användare en projektroll. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



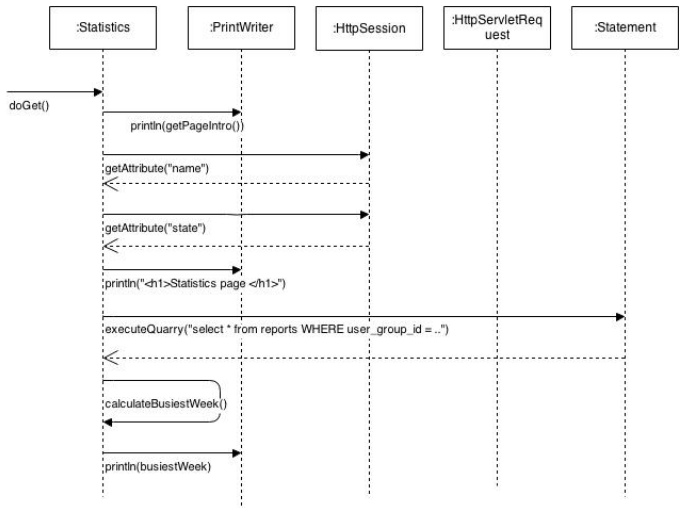
Figur 18: Sekvensdiagram som visar hur en förfrågan om att ta bort en användare från en projektgrupp hanteras i klassen ProjectLeader.(Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



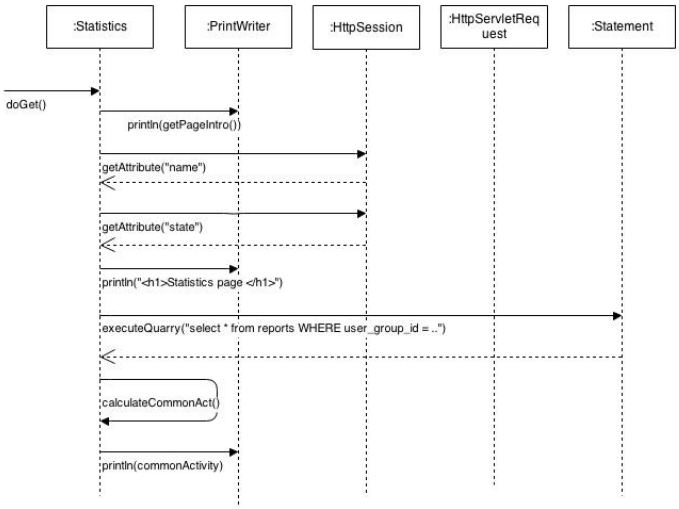
Figur 19: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att visa alla tidrapporter i en projektgrupp. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



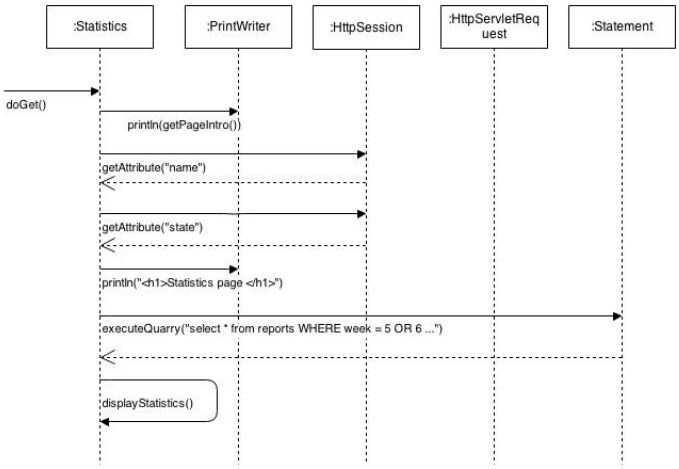
Figur 20: Sekvensdiagram som visar hur klassen ReportHandling hanterar en förfrågan om att signera en rapport. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



Figur 21: Sekvensdiagram som visar hur en förfrågan om att visa vilken vecka som har flest totala inlagda minuter hanteras i klassen Statistics. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



Figur 22: Sekvensdiagram som beskriver hur en förfrågan om att visa vilken aktivitet som har flest totala inlagda minuter hanteras i klassen Statistics. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)



Figur 23: 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. (Observera att inte alla meddelanden visas i sekvensen, utan endast dom som krävs för att förstå diagrammet.)