Dell Semester project

15COS2v14, Group 2, May 10th, 2015

Jan Bačík, Andreas Poulsen, Lasse Jacobsen

Table of Contents

Product Report

Business Case

By Andreas

Claus Borvang of Dell’s MDF department introduced us to the project with a presentation. Claus talked about the current system of communication between Dell and a partner, which sounded overly complicated, and like a system that had been improvised since the beginning. He understandably wants a system, one that makes the work of him and his colleagues easier and faster. It was clear to tell that Claus had a very clear idea and outline of an envisioned product, and this vision is what we want to base our application on.

Dell’s current process of dealing with projects is through a lot of emails and the state of these projects manually written down in a spreadsheet shared within his department. This method of working is clearly not very effective which is also stressed by Claus, and can easily be improved with an application as requested.

Vision

By Jan

The vision is to create an application that both simplifies and improves the workflow of everyone involved in the Dell’s MDF program. This will happen through an easy-to-use and intuitive application with a web interface.

Traceability model

By Jan

Visions	Goals	Features
Fast workflow	Reduce time spent on project management by 81%	Smart sort and search
Scalable solution	Ability to manage up to 2147483647 projects	Web application
Structured, reliable system	Consistent project structure	Notifications
Effective budget utilization	Spend >95% of quarter budgets	Automated statistics

		Budget overview

Activity diagram

By Andreas

As is

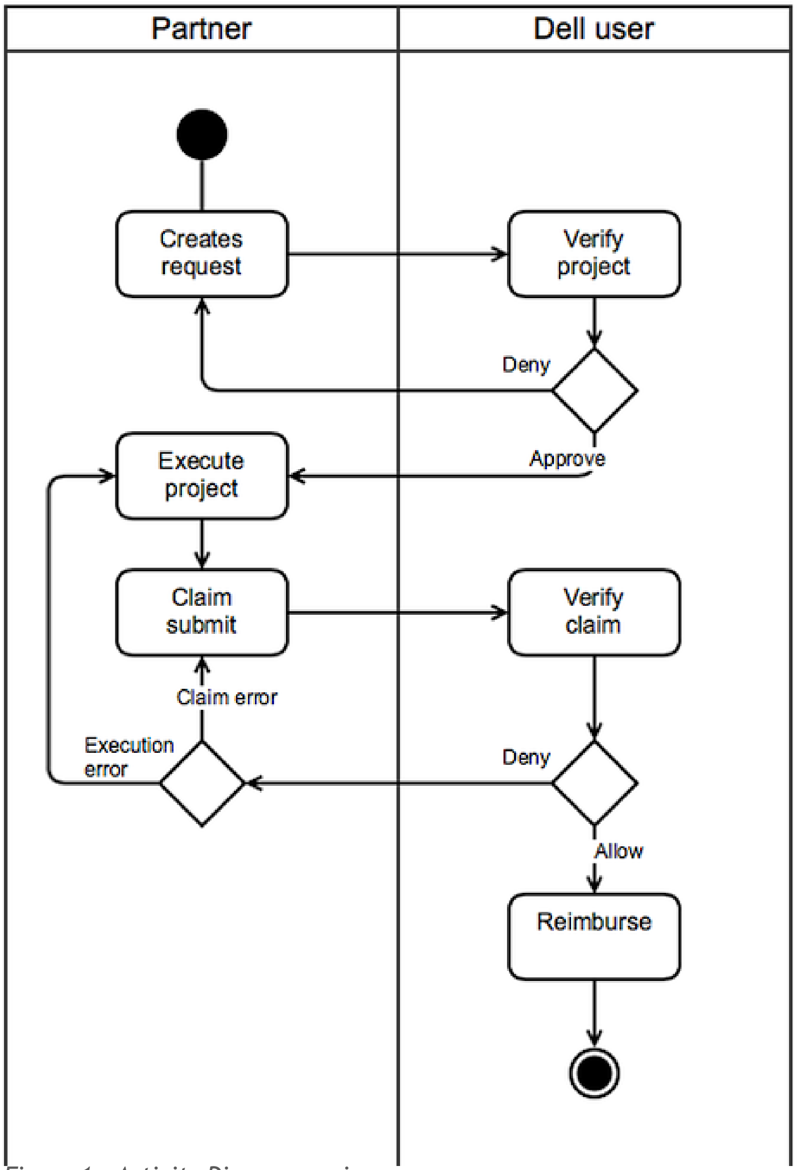


Figure 1 - Activity Diagram, as is

Activity Diagram shows the process of a project in the current system. It is to be noted that every arrow between the two swim lanes is communication between the partner and dell users through email. As well that either party can cancel the process at nearly every step.

To be

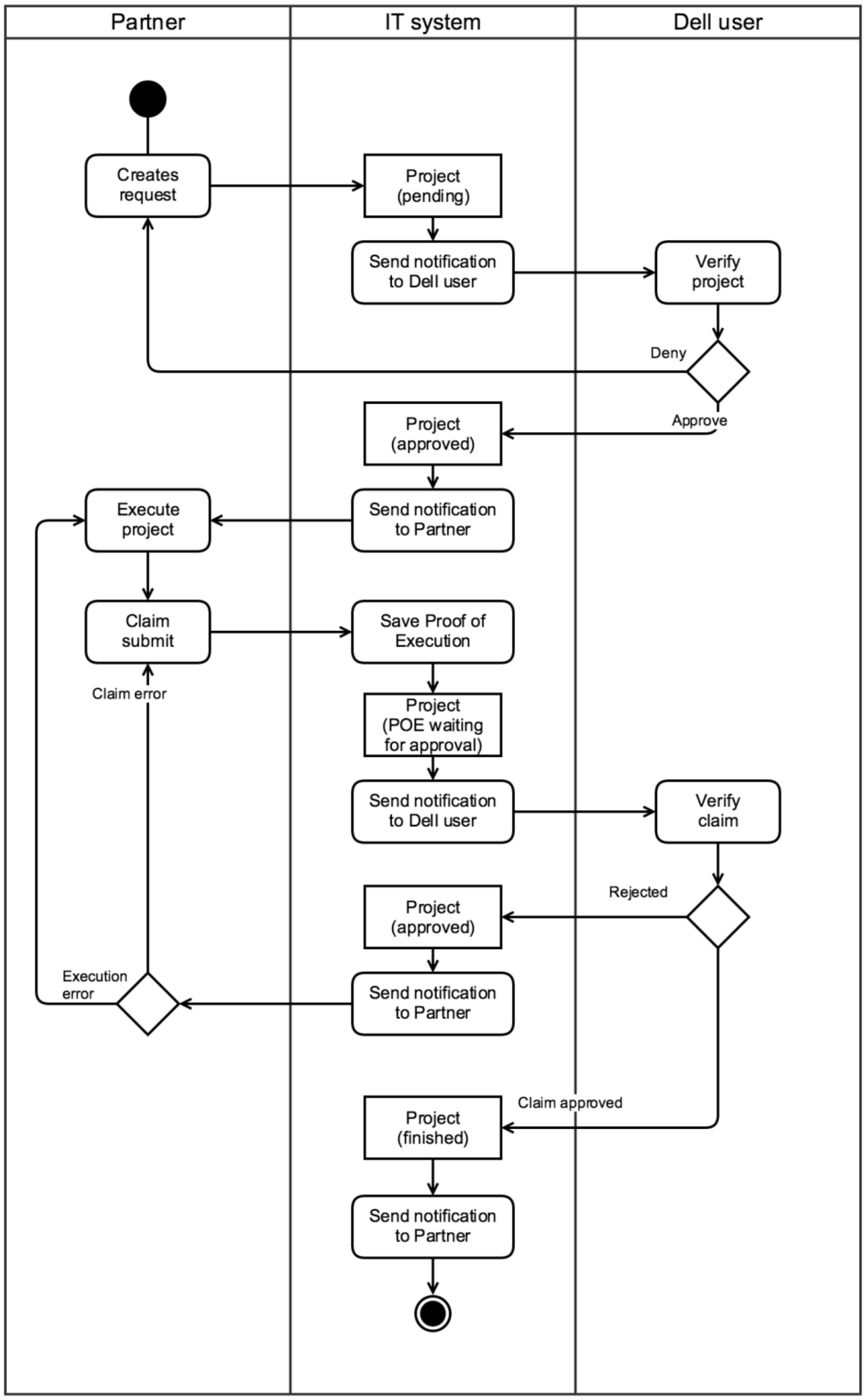


Figure 2 – Activity Diagram, to be

To be diagram shows the process of a project if it went through our IT system. The big difference between the two diagrams is that our applications acts as an intermedium between the two parties, simplifying and unifying all data and communication. Also not shown in this diagram is the ability for both parties to cancel the project at any stage but the last.

Domain model

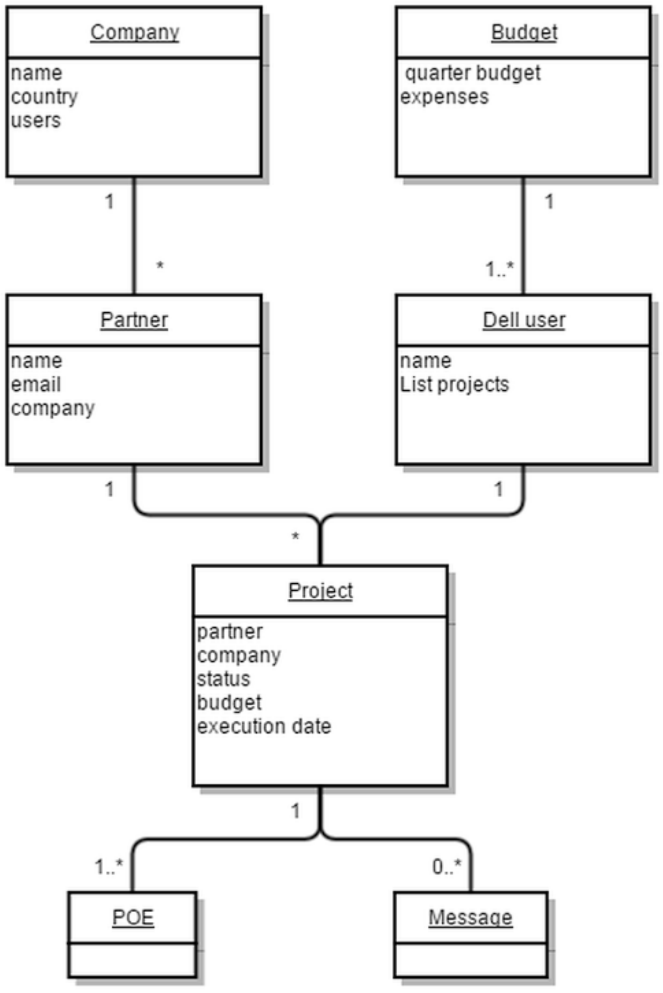


Figure 3 - Domain Model

Our domain model is simple and easily readable. Partner and Dell user are connected through projects which has both POE’s and messages, this is how the two parties communicate and do business. Partners have are individual users belonging to tem, Dell users are individual as well and have access to the quarterly budget.

Architecture and Design Patterns

By Lasse

When building applications of moderate to large scale involving multiple developers it's essential to maintain a well-defined software architecture. Application architecture is primarily meant to provide an efficient development environment to make it easier for developers to build software.

We will spent some time here describing and justifying the patterns we've used within our implementation.

Layering

We have used a 3-layer model (more commonly known as 'three-tier architecture') for our project structure. The three layers are as follows:

- Presentation layer
- Domain layer
- Data source layer

The presentation layer

…is the connection point for the user to the system as a whole; this layer contains the application interface and enables the user to interact with the system via inputs and outputs. In our case, the presentation layer is presented through a web browser using a combination of JSPs and one servlet. The presentation layer uses the services of the domain layer, and is connected to the domain layer exclusively via controller to ensure low coupling.

The domain layer

…is the 'core' of the program; within this layer, we have the logic and more complex calculations that make up the most meaningful aspects of an application. Within the domain layer we have our controller class; this is the connecting point between the presentation layer, the data source layer, and the standard java classes which exist in the domain layer. Whereas the presentation layer uses the services of the domain layer, the domain layer uses the services of the data source layer.

The data source layer

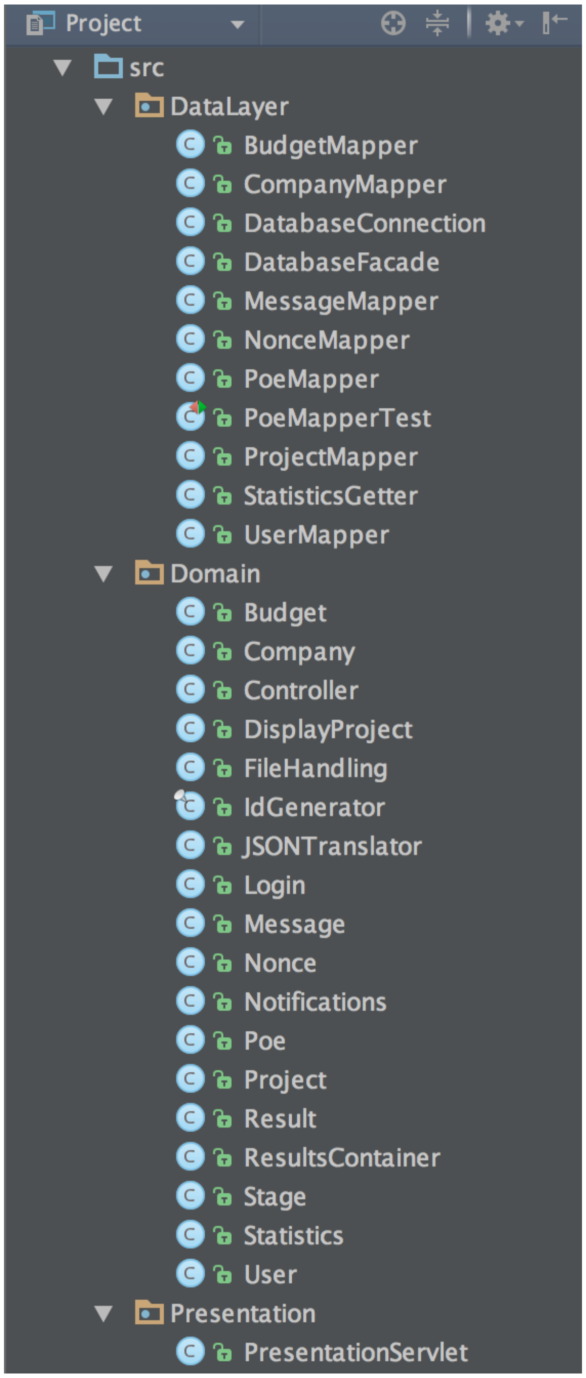
…is the sole connection to the database within the application; any and all SQL scripts are processed within this layer. The connection to the database is retrieved and used exclusively in this layer. The data source layer can be utilized by making calls to the database facade;

much like the controller knows everything in the domain layer, the database facade knows everything within the data source layer; this is done to ensure low coupling and encourage high cohesion within the application.

The advantages of 3-layer model

- Low coupling. There are only single access points between the three layers, thereby ensuring low coupling between the layers.
- High cohesion. Here is a short description of high cohesion:
  "In object-oriented programming, if the methods that serve a class tend to be similar in many aspects, then the class is said to have high cohesion."
  The 3 layers have very well defined purposes, and it's therefore very easy to generalize the way methods are constructed (most of the method calls will be similar in nature; take a look at our data source layer for a great example of high cohesion).
- Parallel working. When you combine high cohesion with low coupling efficiency is almost a guarantee; it's very easy for several people to work in the same or different layers when there are standardized method calls and well defined connections between the layers.

There are more advantages to using the layering model (several services could use a specific layer independently, for example), but the aforementioned advantages are the most relevant in the context of our project and use of the 3-layer model.



```
Project
▼ src
  ▼ DataLayer
      BudgetMapper
      CompanyMapper
      DatabaseConnection
      DatabaseFacade
      MessageMapper
      NonceMapper
      PoeMapper
      PoeMapperTest
      ProjectMapper
      StatisticsGetter
      UserMapper
  ▼ Domain
      Budget
      Company
      Controller
      DisplayProject
      FileHandling
      IdGenerator
      JSONTranslator
      Login
      Message
      Nonce
      Notifications
      Poe
      Project
      Result
      ResultsContainer
      Stage
      Statistics
      User
  ▼ Presentation
      PresentationServlet
```

Figure 4 - Project structure with the 3 layers. Note that the JSP files are stored elsewhere, and are therefore not visible here.

Here is an overview of the project structure for our application (to document our use of the 3-layer model)

Design Patterns (Design qualities)

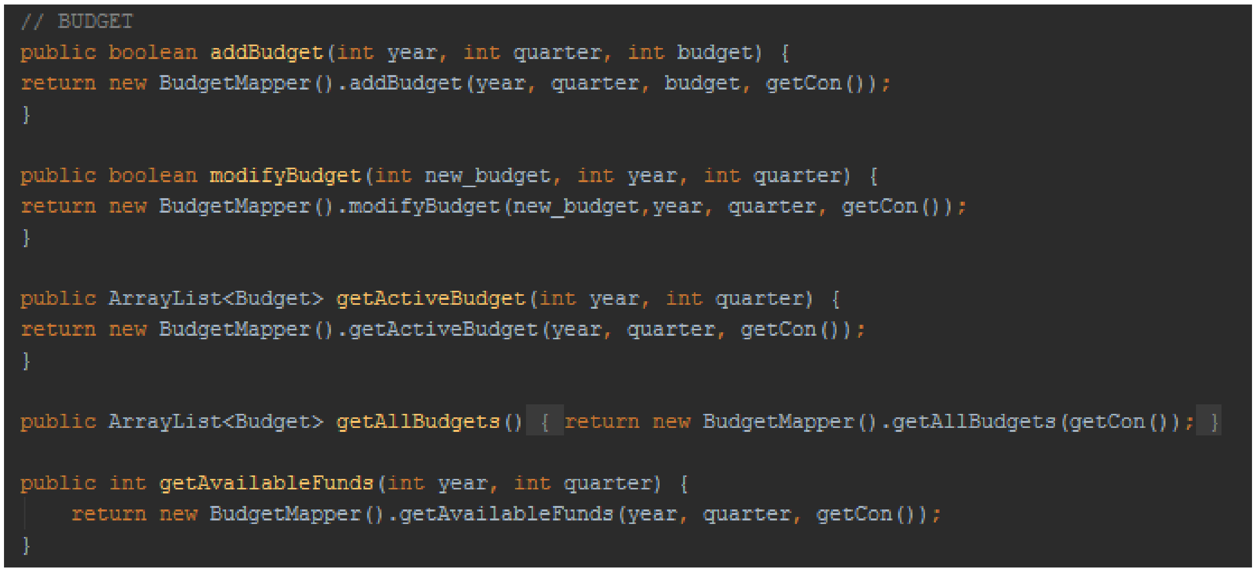
"Design pattern is a common way of solving recurring problem. Classes in all design patterns are just standard classes. What is important is how they are structured and how they work together to solve a given problem in the best possible way."

http://stackoverflow.com/questions/5242429/what-is-facade-design-pattern

When structuring your code, there is variety of design patterns you can use to ease code implementation and understanding. We're going to describe which ones we've used and where:

The facade pattern could be said to be an 'interface to a layer'; a facade is a connection point to a layer and works by 'collecting' all the functionality within a layer in a single class. The advantages of the facade pattern include simplifying access to functionality within a layer as well as potentially providing an easy way to use and read code of any complexity.

We used the facade pattern twice in our project: Once in the domain (Controller) and once in the data source (DatabaseFacade). Here is a code snippet from our DatabaseFacade class demonstrating our use of the facade pattern. The snippet illustrates how the facade handles calls to the budget object mapper class.

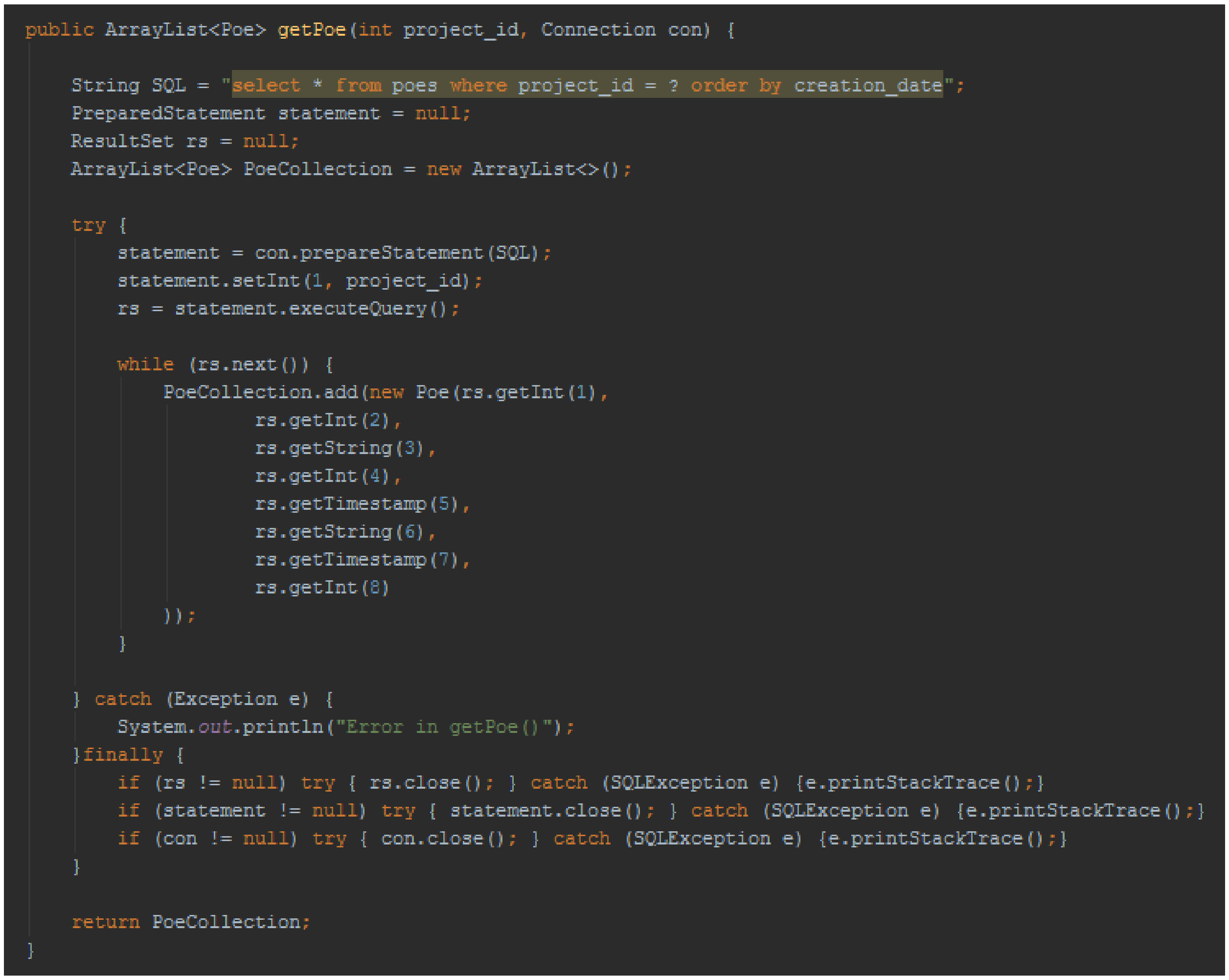


```
// BUDGET
public boolean addBudget(int year, int quarter, int budget) {
return new BudgetMapper().addBudget(year, quarter, budget, getCon());
}

public boolean modifyBudget(int new_budget, int year, int quarter) {
return new BudgetMapper().modifyBudget(new_budget,year, quarter, getCon());
}

public ArrayList<Budget> getActiveBudget(int year, int quarter) {
return new BudgetMapper().getActiveBudget(year, quarter, getCon());
}

public ArrayList<Budget> getAllBudgets() { return new BudgetMapper().getAllBudgets(getCon()); }

public int getAvailableFunds(int year, int quarter) {
    return new BudgetMapper().getAvailableFunds(year, quarter, getCon());
}
```

Figure 5 - Facade structure

This is the structure the facade pattern follows; it has no inherent functionality outside organizing method calls from the classes in the data source layer.

Object-relational mapping (ORM) is a tool you can use to translate information from a database to objects; when you make select statements to your database you store the results within objects so they can be used in an object-oriented programming language like java. We've used ORM with great success within our application - here's an example from our code:



```java
public ArrayList<Poe> getPoe(int project_id, Connection con) {

    String SQL = "select * from poes where project_id = ? order by creation_date";
    PreparedStatement statement = null;
    ResultSet rs = null;
    ArrayList<Poe> PoeCollection = new ArrayList<>();

    try {
        statement = con.prepareStatement(SQL);
        statement.setInt(1, project_id);
        rs = statement.executeQuery();

        while (rs.next()) {
            PoeCollection.add(new Poe(rs.getInt(1),
                    rs.getInt(2),
                    rs.getString(3),
                    rs.getInt(4),
                    rs.getTimestamp(5),
                    rs.getString(6),
                    rs.getTimestamp(7),
                    rs.getInt(8)
            ));
        }

    } catch (Exception e) {
        System.out.println("Error in getPoe()");
    }finally {
        if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
        if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
        if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
    }

    return PoeCollection;
}
```

Figure 6 - Proof Of Execution mapper

Above is an example of our POE (Proof Of Execution) object mapper; the most important thing to note is that we are creating a POE object for each entry in our ResultSet; this is ORM.

Design class diagram

By Andreas

Due to the extent of our application, our class diagram is not simple but complex and with a lot of classes and methods, we have therefore chosen not to include the full diagram directly in the report but it is available here in full eversion and quality - i.imgur.com/Q8gSmlq.png.

In the below version, the data classes have been omitted as well as some redundant methods. You can clearly see the diagram divided into the three layers as described previously. Starting with our one servlet that is only dependent on our controller class.

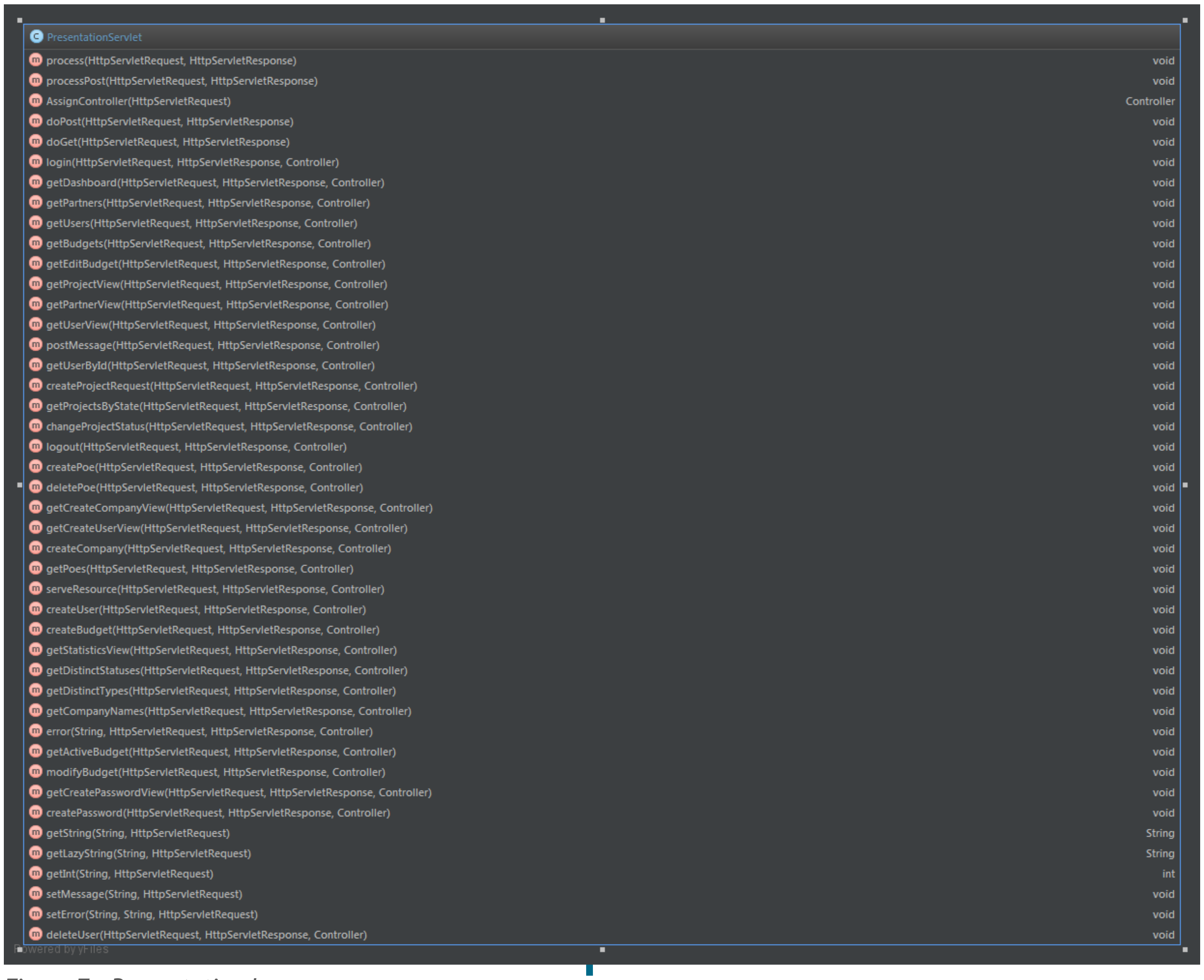

PresentationServlet
process(HttpServletRequest, HttpServletResponse) void
processPost(HttpServletRequest, HttpServletResponse) void
AssignController(HttpServletRequest) Controller
doPost(HttpServletRequest, HttpServletResponse) void
doGet(HttpServletRequest, HttpServletResponse) void
login(HttpServletRequest, HttpServletResponse, Controller) void
getDashboard(HttpServletRequest, HttpServletResponse, Controller) void
getPartners(HttpServletRequest, HttpServletResponse, Controller) void
getUsers(HttpServletRequest, HttpServletResponse, Controller) void
getBudgets(HttpServletRequest, HttpServletResponse, Controller) void
getEditBudget(HttpServletRequest, HttpServletResponse, Controller) void
getProjectView(HttpServletRequest, HttpServletResponse, Controller) void
getPartnerView(HttpServletRequest, HttpServletResponse, Controller) void
getUserView(HttpServletRequest, HttpServletResponse, Controller) void
postMessage(HttpServletRequest, HttpServletResponse, Controller) void
getUserById(HttpServletRequest, HttpServletResponse, Controller) void
createProjectRequest(HttpServletRequest, HttpServletResponse, Controller) void
getProjectsByState(HttpServletRequest, HttpServletResponse, Controller) void
changeProjectStatus(HttpServletRequest, HttpServletResponse, Controller) void
logout(HttpServletRequest, HttpServletResponse, Controller) void
createPoe(HttpServletRequest, HttpServletResponse, Controller) void
deletePoe(HttpServletRequest, HttpServletResponse, Controller) void
getCreateCompanyView(HttpServletRequest, HttpServletResponse, Controller) void
getCreateUserView(HttpServletRequest, HttpServletResponse, Controller) void
createCompany(HttpServletRequest, HttpServletResponse, Controller) void
getPoes(HttpServletRequest, HttpServletResponse, Controller) void
serveResource(HttpServletRequest, HttpServletResponse, Controller) void
createUser(HttpServletRequest, HttpServletResponse, Controller) void
createBudget(HttpServletRequest, HttpServletResponse, Controller) void
getStatisticsView(HttpServletRequest, HttpServletResponse, Controller) void
getDistinctStatuses(HttpServletRequest, HttpServletResponse, Controller) void
getDistinctTypes(HttpServletRequest, HttpServletResponse, Controller) void
getCompanyNames(HttpServletRequest, HttpServletResponse, Controller) void
error(String, HttpServletRequest, HttpServletResponse, Controller) void
getActiveBudget(HttpServletRequest, HttpServletResponse, Controller) void
modifyBudget(HttpServletRequest, HttpServletResponse, Controller) void
getCreatePasswordView(HttpServletRequest, HttpServletResponse, Controller) void
createPassword(HttpServletRequest, HttpServletResponse, Controller) void
getString(String, HttpServletRequest) String
getLazyString(String, HttpServletRequest) String
getInt(String, HttpServletRequest) int
setMessage(String, HttpServletRequest) void
setError(String, String, HttpServletRequest) void
deleteUser(HttpServletRequest, HttpServletResponse, Controller) void
Powered by yFiles

Figure 7 - Presentation layer

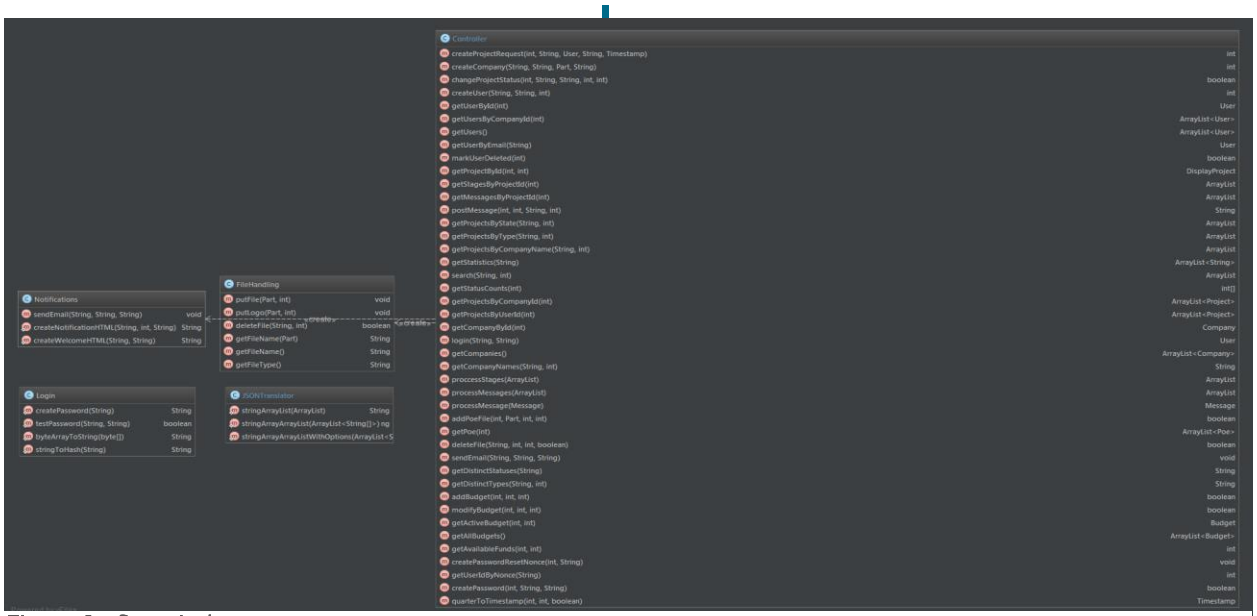

Controller
createProjectRequest(int, String, User, String, Timestamp) int
createCompany(String, String, Part, String) int
changeProjectStatus(int, String, String, int, int) boolean
createUser(String, String, int) int
getUserById(int) User
getUsersByCompanyId(int) ArrayList<User>
getUsers() ArrayList<User>
getUserByEmail(String) User
markUserDeleted(int) boolean
getProjectById(int, int) DisplayProject
getStagesByProjectId(int) ArrayList
getMessagesByProjectId(int) ArrayList
postMessage(int, int, String, int) String
getProjectsByState(String, int) ArrayList
getProjectsByType(String, int) ArrayList
getProjectsByCompanyName(String, int) ArrayList
getStatistics(String) ArrayList<String>
search(String, int) ArrayList
getStatusCounts(int) int[]
getProjectsByCompanyId(int) ArrayList<Project>
getProjectsByUserId(int) ArrayList<Project>
getCompanyById(int) Company
login(String, String) User
getCompanies() ArrayList<Company>
getCompanyNames(String, int) String
proccessStages(ArrayList) ArrayList
processMessages(ArrayList) ArrayList
processMessage(Message) Message
addPoeFile(int, Part, int, int) boolean
getPoe(int) ArrayList<Poe>
deleteFile(String, int, int, boolean) boolean
sendEmail(String, String, String) void
getDistinctStatuses(String) String
getDistinctTypes(String, int) String
addBudget(int, int, int) boolean
modifyBudget(int, int, int) boolean
getActiveBudget(int, int) Budget
getAllBudgets() ArrayList<Budget>
getAvailableFunds(int, int) int
createPasswordResetNonce(int, String) void
getUserIdByNonce(String) int
createPassword(int, String, String) boolean
quarterToTimestamp(int, int, boolean) Timestamp
Notifications
sendEmail(String, String, String) void
createNotificationHTML(String, int, String) String
createWelcomeHTML(String, String) String
FileHandling
putFile(Part, int) void
putLogo(Part, int) void
deleteFile(String, int) boolean
getFileName(Part) String
getFileName() String
getFileType() String
«create»
«create»
Login
createPassword(String) String
testPassword(String, String) boolean
byteArrayToString(byte[]) String
stringToHash(String) String
JSONTranslator
stringArrayList(ArrayList) String
stringArrayArrayList(ArrayList<String[]>) ng
stringArrayArrayListWithOptions(ArrayList<S

Figure 8 - Domain layer

The controller has a few helper classes for file handling, email sending, login verification and translating java data objects into JSON. These classes could have been in the controller instead, but to maintain organization we have split them up, to reduce controller`s complexity. The controller is retrieving all data through the database facade, which is the only relation between controller and the database.

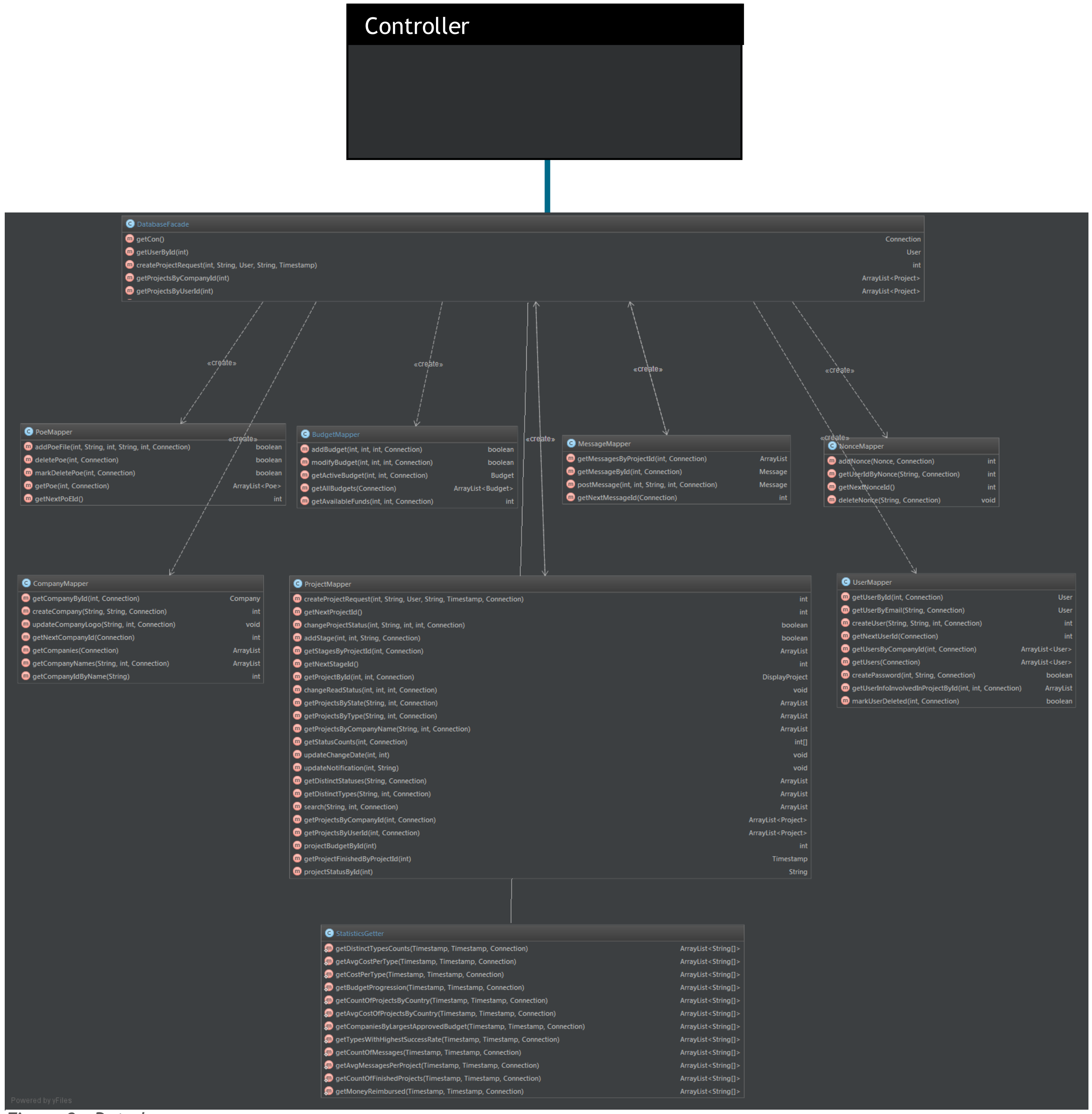

Controller
DatabaseFacade
getCon() Connection
getUserById(int) User
createProjectRequest(int, String, User, String, Timestamp) int
getProjectsByCompanyId(int) ArrayList<Project>
getProjectsByUserId(int) ArrayList<Project>
«create»
«create»
«create»
«create»
«create»
«create»
«create»
PoeMapper
addPoeFile(int, String, int, String, int, Connection) boolean
deletePoe(int, Connection) boolean
markDeletePoe(int, Connection) boolean
getPoe(int, Connection) ArrayList<Poe>
getNextPoEId() int
BudgetMapper
addBudget(int, int, int, Connection) boolean
modifyBudget(int, int, int, Connection) boolean
getActiveBudget(int, int, Connection) Budget
getAllBudgets(Connection) ArrayList<Budget>
getAvailableFunds(int, int, Connection) int
MessageMapper
getMessagesByProjectId(int, Connection) ArrayList
getMessageById(int, Connection) Message
postMessage(int, int, String, int, Connection) Message
getNextMessageId(Connection) int
NonceMapper
addNonce(Nonce, Connection) int
getUserIdByNonce(String, Connection) int
getNextNonceId() int
deleteNonce(String, Connection) void
CompanyMapper
getCompanyById(int, Connection) Company
createCompany(String, String, Connection) int
updateCompanyLogo(String, int, Connection) void
getNextCompanyId(Connection) int
getCompanies(Connection) ArrayList
getCompanyNames(String, int, Connection) ArrayList
getCompanyIdByName(String) int
ProjectMapper
createProjectRequest(int, String, User, String, Timestamp, Connection) int
getNextProjectId() int
changeProjectStatus(int, String, int, int, Connection) boolean
addStage(int, int, String, Connection) boolean
getStagesByProjectId(int, Connection) ArrayList
getNextStageId() int
getProjectById(int, int, Connection) DisplayProject
changeReadStatus(int, int, int, Connection) void
getProjectsByState(String, int, Connection) ArrayList
getProjectsByType(String, int, Connection) ArrayList
getProjectsByCompanyName(String, int, Connection) ArrayList
getStatusCounts(int, Connection) int[]
updateChangeDate(int, int) void
updateNotification(int, String) void
getDistinctStatuses(String, Connection) ArrayList
getDistinctTypes(String, int, Connection) ArrayList
search(String, int, Connection) ArrayList
getProjectsByCompanyId(int, Connection) ArrayList<Project>
getProjectsByUserId(int, Connection) ArrayList<Project>
projectBudgetById(int) int
getProjectFinishedByProjectId(int) Timestamp
projectStatusById(int) String
UserMapper
getUserById(int, Connection) User
getUserByEmail(String, Connection) User
createUser(String, String, int, Connection) int
getNextUserId(Connection) int
getUsersByCompanyId(int, Connection) ArrayList<User>
getUsers(Connection) ArrayList<User>
createPassword(int, String, Connection) boolean
getUserInfoInvolvedInProjectById(int, int, Connection) ArrayList
markUserDeleted(int, Connection) boolean
StatisticsGetter
getDistinctTypesCounts(Timestamp, Timestamp, Connection) ArrayList<String[]>
getAvgCostPerType(Timestamp, Timestamp, Connection) ArrayList<String[]>
getCostPerType(Timestamp, Timestamp, Connection) ArrayList<String[]>
getBudgetProgression(Timestamp, Timestamp, Connection) ArrayList<String[]>
getCountOfProjectsByCountry(Timestamp, Timestamp, Connection) ArrayList<String[]>
getAvgCostOfProjectsByCountry(Timestamp, Timestamp, Connection) ArrayList<String[]>
getCompaniesByLargestApprovedBudget(Timestamp, Timestamp, Connection) ArrayList<String[]>
getTypesWithHighestSuccessRate(Timestamp, Timestamp, Connection) ArrayList<String[]>
getCountOfMessages(Timestamp, Timestamp, Connection) ArrayList<String[]>
getAvgMessagesPerProject(Timestamp, Timestamp, Connection) ArrayList<String[]>
getCountOfFinishedProjects(Timestamp, Timestamp, Connection) ArrayList<String[]>
getMoneyReimbursed(Timestamp, Timestamp, Connection) ArrayList<String[]>
Powered by yFiles

Figure 9 - Data layer

Our data layer contains one facade and a mapper class almost all tables in the database as well as a one extra, the StatisticsGetter that receives data from multiple tables.

If you are looking at the full diagram (i.imgur.com/Q8gSmIq.png), you can see, in addition to the three layers, all our data classes are both connected to the controller but also a mapper most likely.

Sequence diagram

By Lasse

We've made a sequence diagram of a project request workflow. Project request is the first step in the workflow and is therefore considered core functionality.

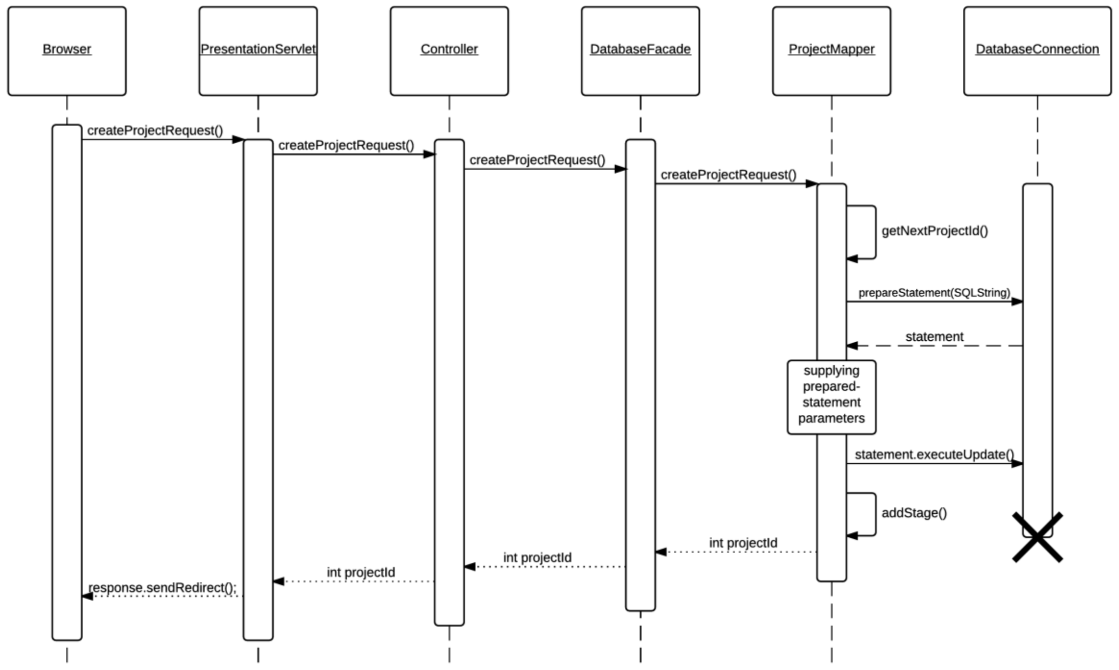


Figure 10 - Sequence diagram of the 'createProjectRequest()' method

Sequence diagram explained:

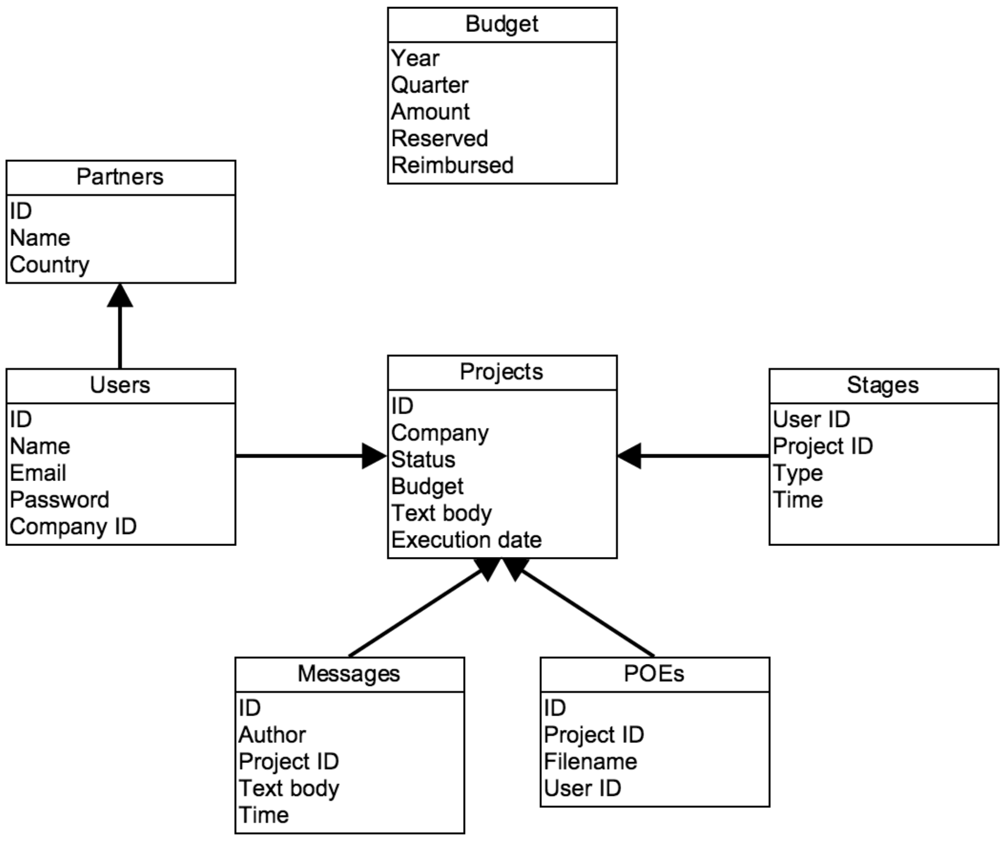
The diagram assumes that user is on the project creation page ('createproject.jsp') and submits a valid form input. The input is passed through the layers to the 'ProjectMapper' in data source layer where the project request is inserted into the database. The project creation method returns id of the new project so the servlet can redirect the user. Here is how the redirect is done:

```
response.sendRedirect("/project?id=" + projectId);
```

It's worthy to note that the methods 'getNextProjectId()' and 'addStage()' each use a separate database connection. Ideally they should all be done on the same connection using transactions to prevent errors in our data; this is definitely something we regret not doing, and were we to do it again, we'd increase our use of transactions to ensure data consistency. Read more on our use of transactions under the report section "SQL Queries".

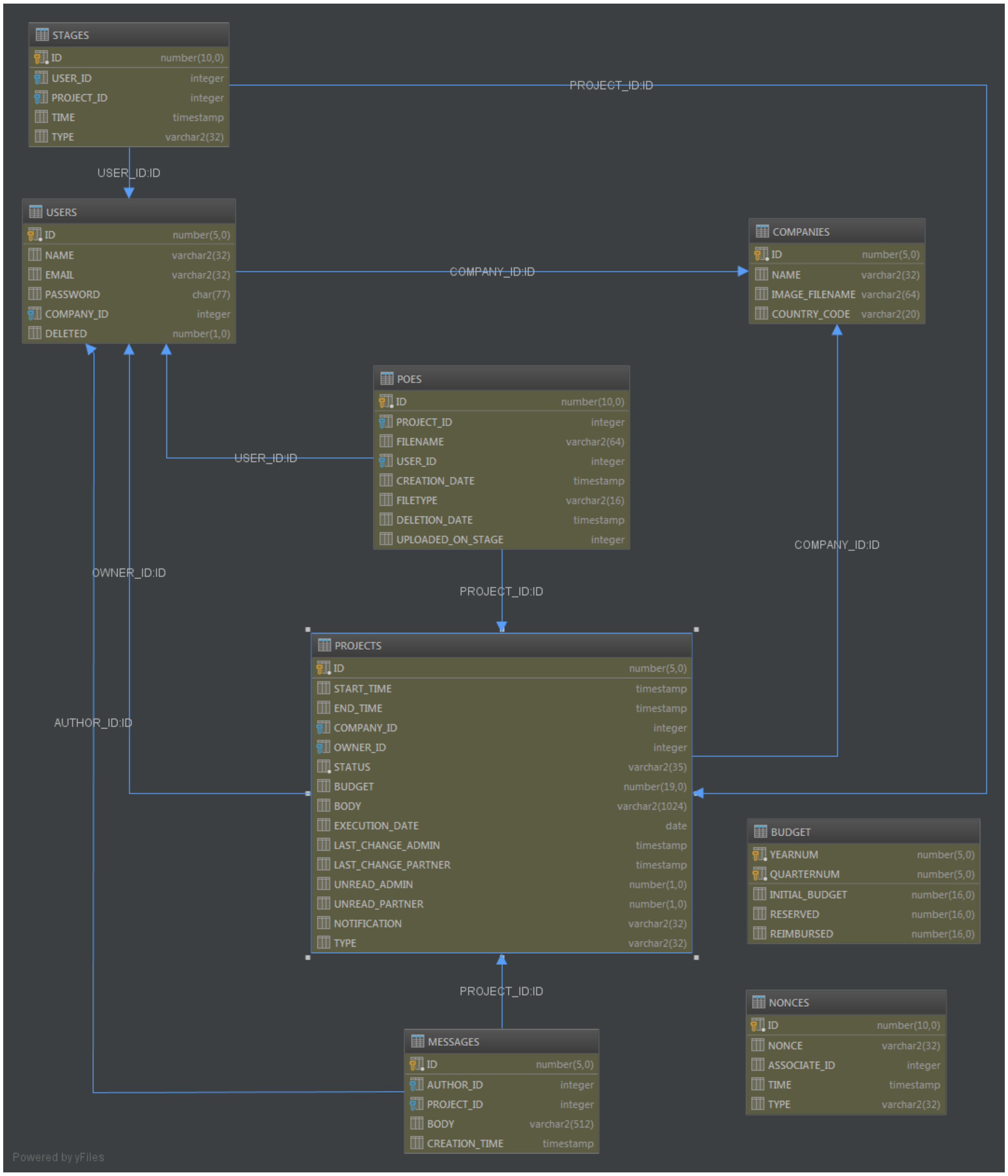
E/R Diagram

By Jan



Relational Schema

By Andreas



Our application requires eight tables to operate at full functionality. Six out of the eight are connected through foreign keys. The two tables not having any relations are ‘Budget’ and ‘Nonces’, ‘Budget’ doesn’t need any relation and is independent as is. The ‘Nonces’ table does not have a foreign key defined but is still related to other tables, it has a column named ‘Associate_id’ which will refer to some id in another table. The idea behind this structure is that the column can refer to multiple tables and which table is defined by the 'Type' column; in retrospect, we realize that this is not the best practice, but it does have some advantages.

Normal forms

1st

We have nowhere used multiple values in the same column, satisfying the atomicity of NF1, as well we have no duplicate rows as all rows are unique.

2nd

We only have one table with composite primary keys, 'Budget'. This table meets the requirements for NF2 because none of the non primary keys are dependent on only one of the primary keys.

3rd

An example of our database fulfilling NF3 is our 'Messages' table where instead of having 'Author_name' and 'Author_id' we only use the later which refers to the users table. 'Author_id' is not determining any functional dependencies in 'Messages' table.

BCNF

Due to our setup all our tables fulfill the Boyce-Codd normal form, this is partly because of the fact that all tables, except 'Budget', are using ID's as the primary key and the nature of that style will often fulfill this normal form. In addition, whenever we have foreign keys, we have only the ID of the foreign table in our table and not multiple columns from the foreign table.

Generally, our database architecture has been really strong since day one, and have not cause us any problems. Whenever we created a table we were very careful about the structure of the table and we would always consult the whole group to really think about it thoroughly. This has most likely caused us to follow the normal forms as they are defined with logic in mind.

Figure 11 - Database architecture

User interface

By Andreas & Jan

One of our goals was to create an application that is easy to use and provides pleasant experience by clever functionality and intuitive user interface. To make this possible, application undergone many UI iterations during the process. The result is simple, consistent design language that is used throughout the system.

Application walkthrough and description

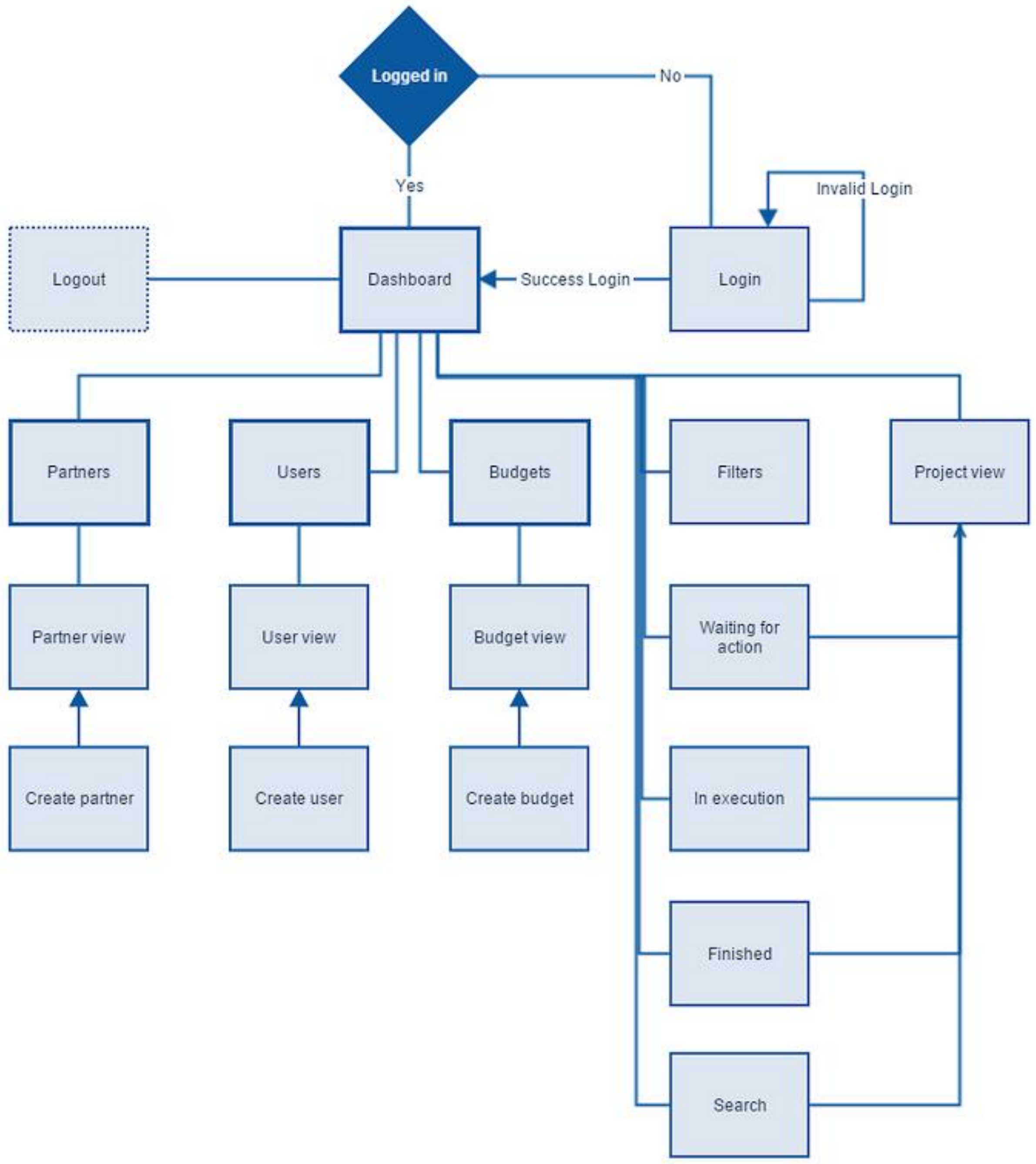


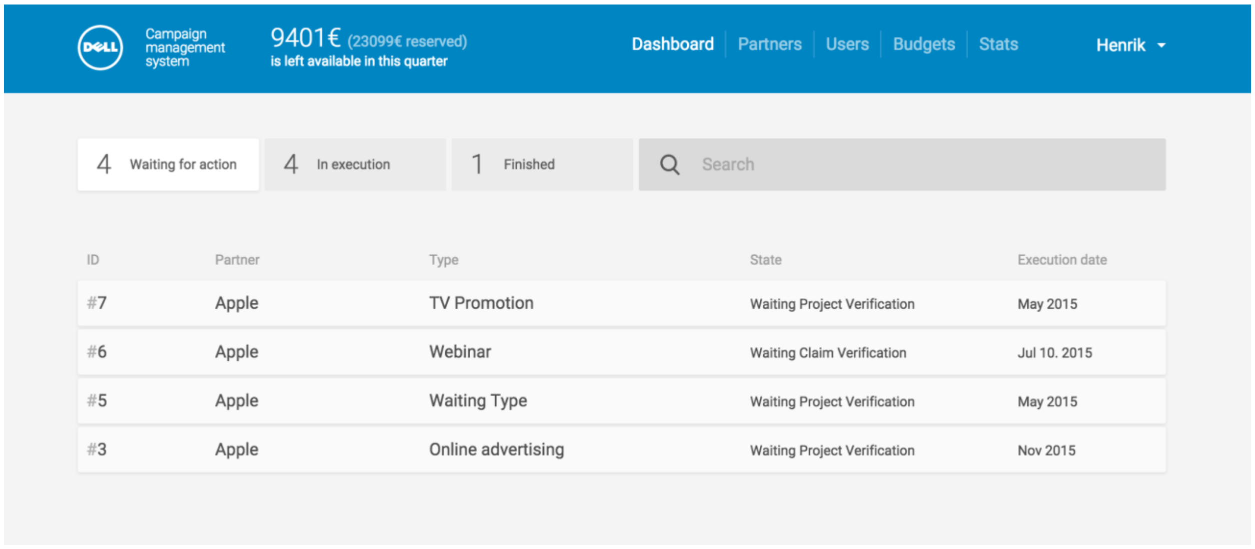
Figure 12 - Website tree

This is an overview of the structure of our web application. On initial visit you will be redirected to a login page, once logged in you will be presented with the dashboard, that is the homepage.

Dashboard

If you are logged in as a Dell user the page features a list of projects, by default they will be filtered by what we call 'Waiting for action'(1), these are projects where for them to continue to the next stage they will require an action from your side. This filter will work like a to-do list, if the list is empty, you've done your job, else you can start working on projects in this section. There are two other filters; 'In Execution' is for projects that are currently being executed by the partner, 'Finished' is a list for projects that have been finished and don't require any action from neither party.

The list of projects features 5 columns, the first 3 are ID, Partner and Type which are all meant to be used for quick identification so Dell users are sure of what project they are clicking on and don't need to open each project to find the one they're looking for. The column 'State' is in what stage the project is in, or if the project is unread it will be a short message of what happened since you last saw the project. Projects will be marked unread by a clear difference by a style change and once clicked they are marked unread. The read/unread is shared between all users on Dell's side.



ID	Partner	Type	State	Execution date
#7	Apple	TV Promotion	Waiting Project Verification	May 2015
#6	Apple	Webinar	Waiting Claim Verification	Jul 10. 2015
#5	Apple	Waiting Type	Waiting Project Verification	May 2015
#3	Apple	Online advertising	Waiting Project Verification	Nov 2015

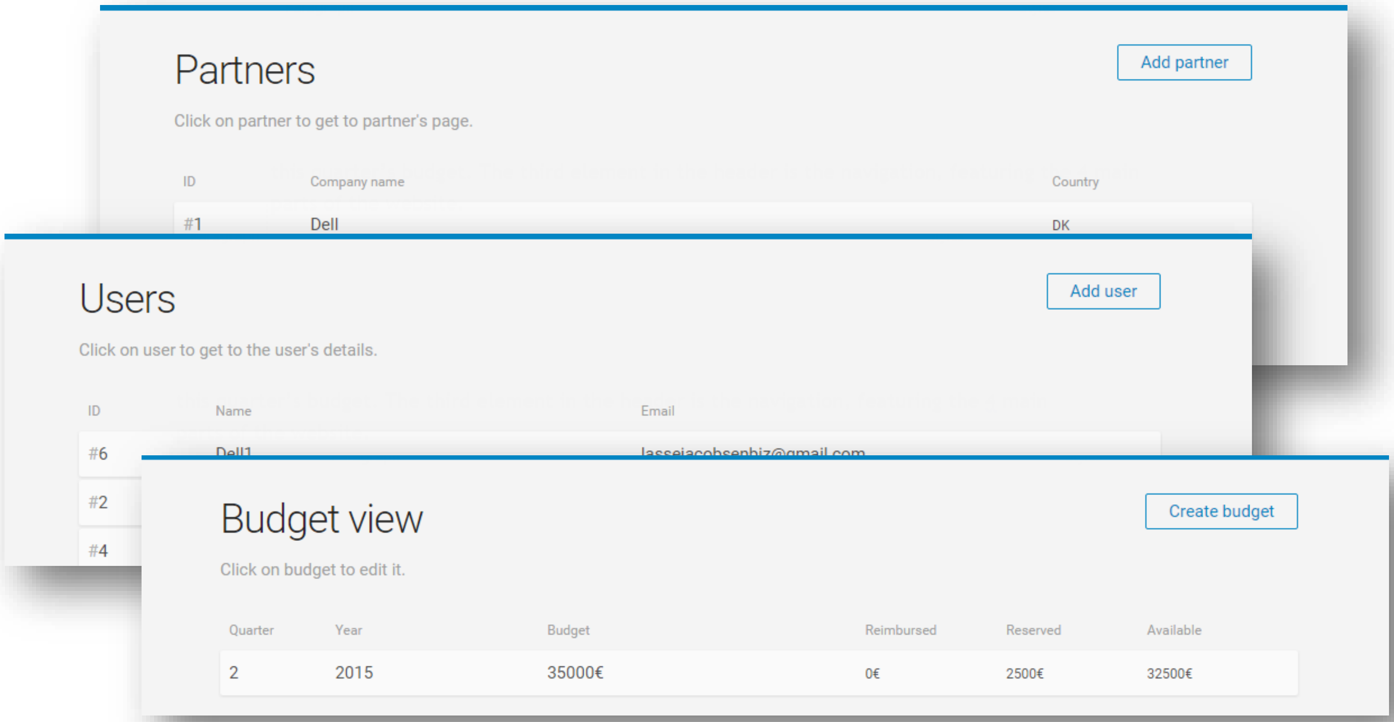
Figure 13 - Dashboard for Dell users

Header

The header is sitting at the top with Dell's signature blue color and features it's logo in to the left, this doubles as a link to the dashboard. Next is shown the available amount left in this quarter's budget. The third element in the header is the navigation, featuring the 5 main parts of the website.

Partners, Users, Budget – Dell users only

These three features follow the same layout and architecture to simplify the users experience. On the first page, when you navigate from the header, you will see a list with all items of the corresponding header you clicked on. From here you can either click on an item in the list you can click the 'Add' button that is placed on the top of the page.


Partners
Add partner
Click on partner to get to partner's page.
Users
Add user
Click on user to get to the user's details.
Budget view
Create budget
Click on budget to edit it.

ID	Company name	Country
#1	Dell	DK

ID	Name	Email
#6	Dell1	lassejacobsenbiz@gmail.com
#2		
#4		

Quarter	Year	Budget	Reimbursed	Reserved	Available
2	2015	35000€	0€	2500€	32500€

Figure 14 - Partner-, Users- and Budget view

Clicking on the 'Add' Button you will be greeted by a page with a form and input fields required for creating a new item of this type. Once created you will be redirected to the page of this newly created item, this screen will be the same as you clicked on an item on the list from the above site.

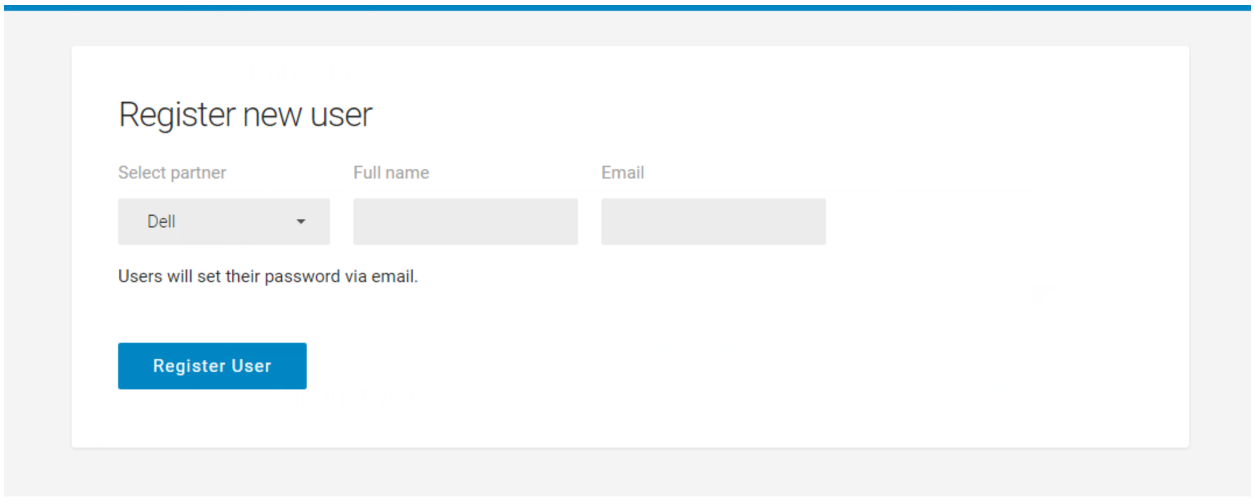


Figure 15 - User creation page

Project view

Timeline approach to project view is the most distinctive and opinionated part of the application. Whole page resembles time-ordered sequence of project related actions or messages in intuitive way. All your actions (examples of actions are project request, project approval, etc.) are aligned to the right, other user's actions are aligned to the left for easier orientation. Important actions representing positive or negative change are color coded as seen on screenshot below.

You can see all the most important information about the project in the upper section. Project's ID, type, state and project request information as partner's name, requested budget, and body text of project request. There is also an option to cancel the project, which by clicking the application asks you again in priority lightbox view, to ensure no one will cancel project unintentionally.

Actions

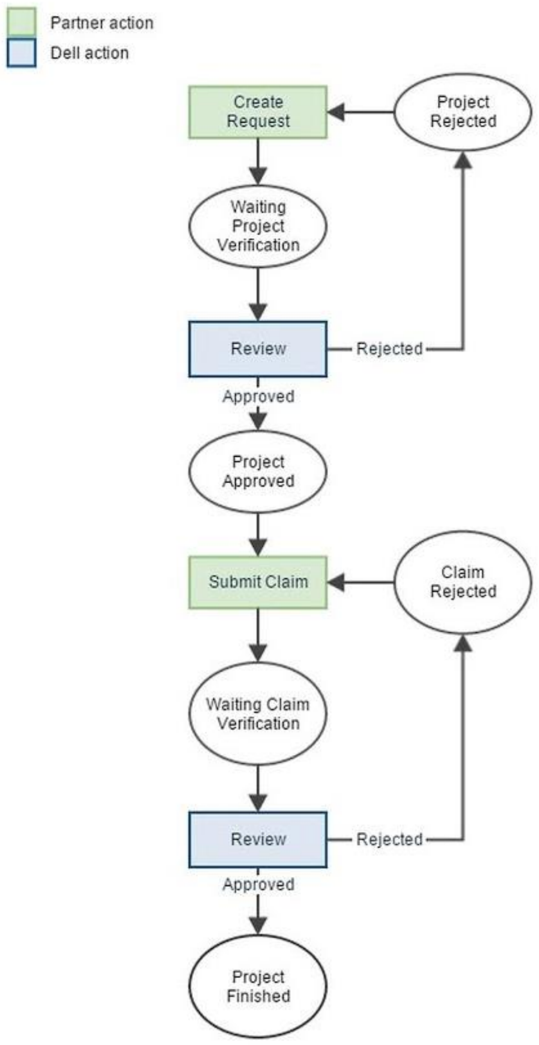
Every action and message has the information about user who created it, time and date. All actions are also specifically diversified to serve their purpose best. You can see this in "Waiting claim verification" box, where Dell user has the ability to download uploaded proofs, or preview images online. Our application can distinguish between different types of files, and serves you either live view of images with ability to preview them in fullscreen lightbox, or displays filetype-specific icon, as table sheet icon displayed below.

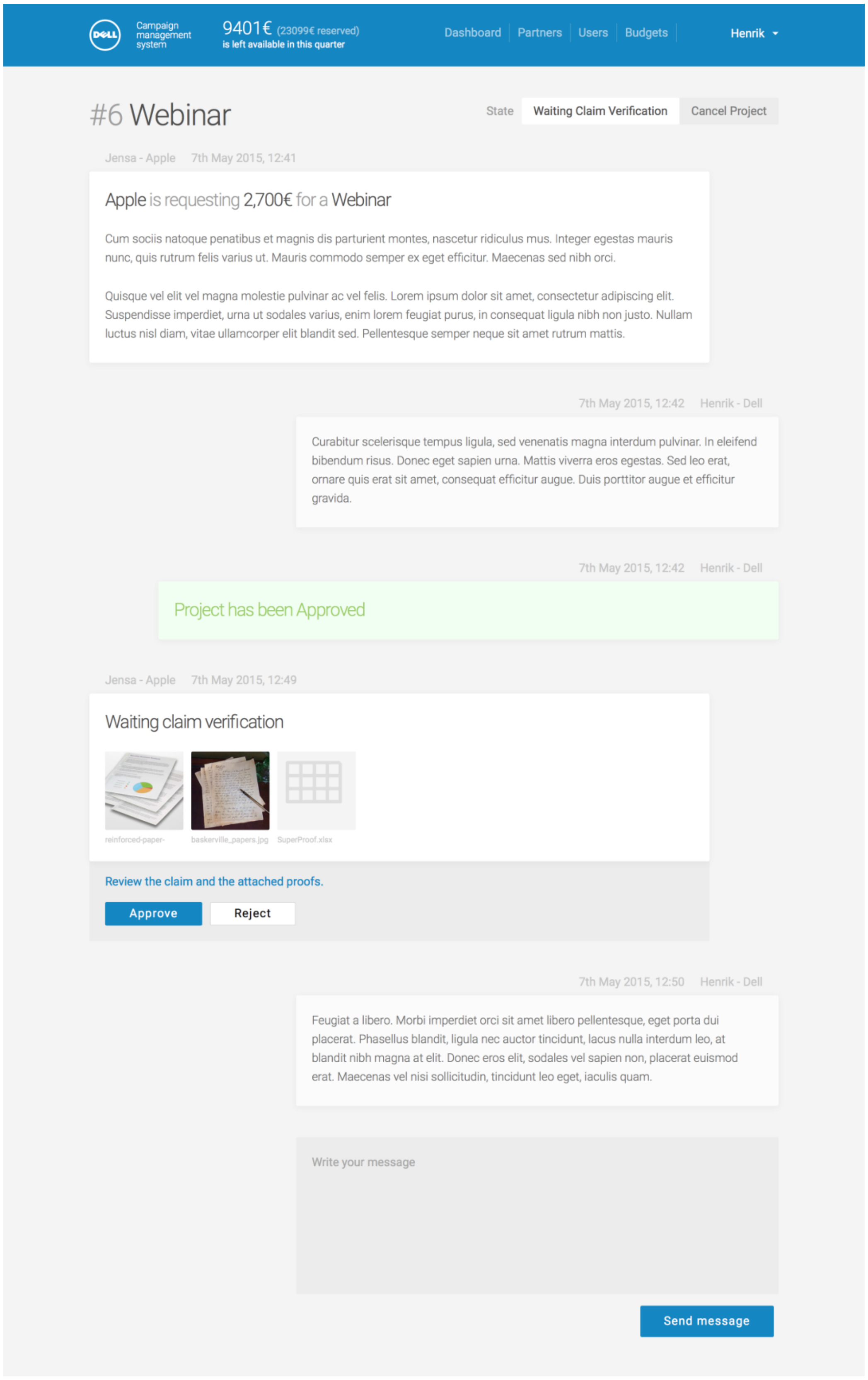
States

A big part of our application is that we have chosen to subdivide projects into seven states, which we have extracted from the initial project presentation, so it can easily convert from Dell's current system. For a project to be finished it doesn't have to advance through all seven, since some of them are meant for a negative state. The states are as follows:

- Waiting Project Verification
- Project Rejected
- Project Approved
- Waiting Claim Verification
- Claim Rejected
- Project Finished
- Cancelled

These states are the core of the application, they will be used to organize and clearly translate the current status of the project to the user.





#6 Webinar

Jensa - Apple 7th May 2015, 12:41

Apple is requesting 2,700€ for a Webinar

Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer egestas mauris nunc, quis rutrum felis varius ut. Mauris commodo semper ex eget efficitur. Maecenas sed nibh orci.

Quisque vel elit vel magna molestie pulvinar ac vel felis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse imperdiet, urna ut sodales varius, enim lorem feugiat purus, in consequat ligula nibh non justo. Nullam luctus nisl diam, vitae ullamcorper elit blandit sed. Pellentesque semper neque sit amet rutrum mattis.

7th May 2015, 12:42 Henrik - Dell

Curabitur scelerisque tempus ligula, sed venenatis magna interdum pulvinar. In eleifend bibendum risus. Donec eget sapien urna. Mattis viverra eros egestas. Sed leo erat, ornare quis erat sit amet, consequat efficitur augue. Duis porttitor augue et efficitur gravida.

7th May 2015, 12:42 Henrik - Dell

Project has been Approved

Jensa - Apple 7th May 2015, 12:49

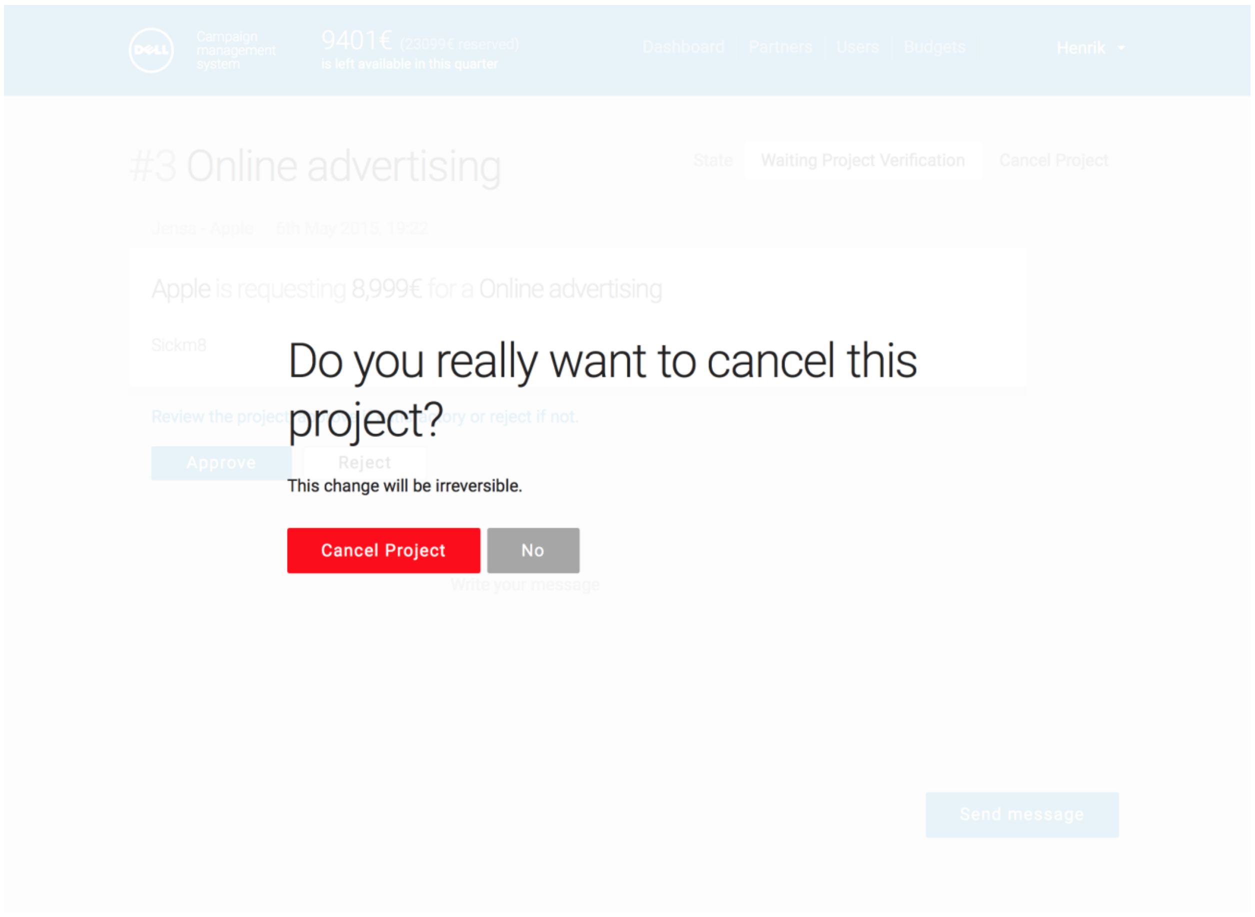
Waiting claim verification

Review the claim and the attached proofs.

7th May 2015, 12:50 Henrik - Dell

Feugiat a libero. Morbi imperdiet orci sit amet libero pellentesque, eget porta dui placerat. Phasellus blandit, ligula nec auctor tincidunt, lacus nulla interdum leo, at blandit nibh magna at elit. Donec eros elit, sodales vel sapien non, placerat euismod erat. Maecenas vel nisi sollicitudin, tincidunt leo eget, iaculis quam.

Figure 16 - Project view


Campaign management system
9401€ (23099€ reserved)
is left available in this quarter
Dashboard
Partners
Users
Budgets
Henrik
#3 Online advertising
State
Waiting Project Verification
Cancel Project
Apple is requesting 8,999€ for a Online advertising
Approve
Reject
Send message

Do you really want to cancel this project?

This change will be irreversible.

Cancel Project | No

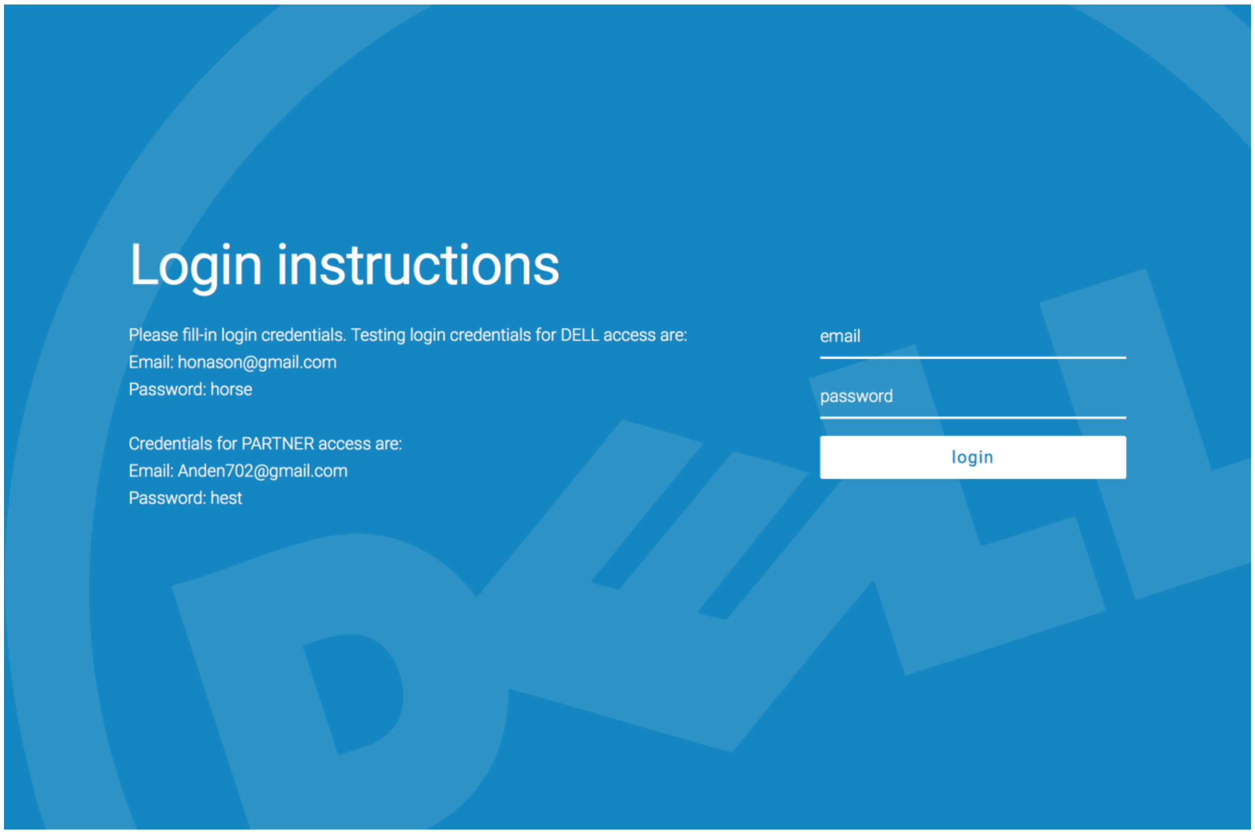
Figure 17 - Project cancellation

Browser compatibility

Application is made to work seamlessly on all major platforms and it's browsers, included IE 11+, Google Chrome, Safari, Mozilla Firefox, Opera, mobile browsers on Apple iOS and Google Android.

Responsive design

Responsive layout of web application is added feature that provides both Dell and partners instant access to the system on their tablets and mobile phones. The layout scales from big desktop screens all the way to displays of mobile phones without losing any of application's functionality.

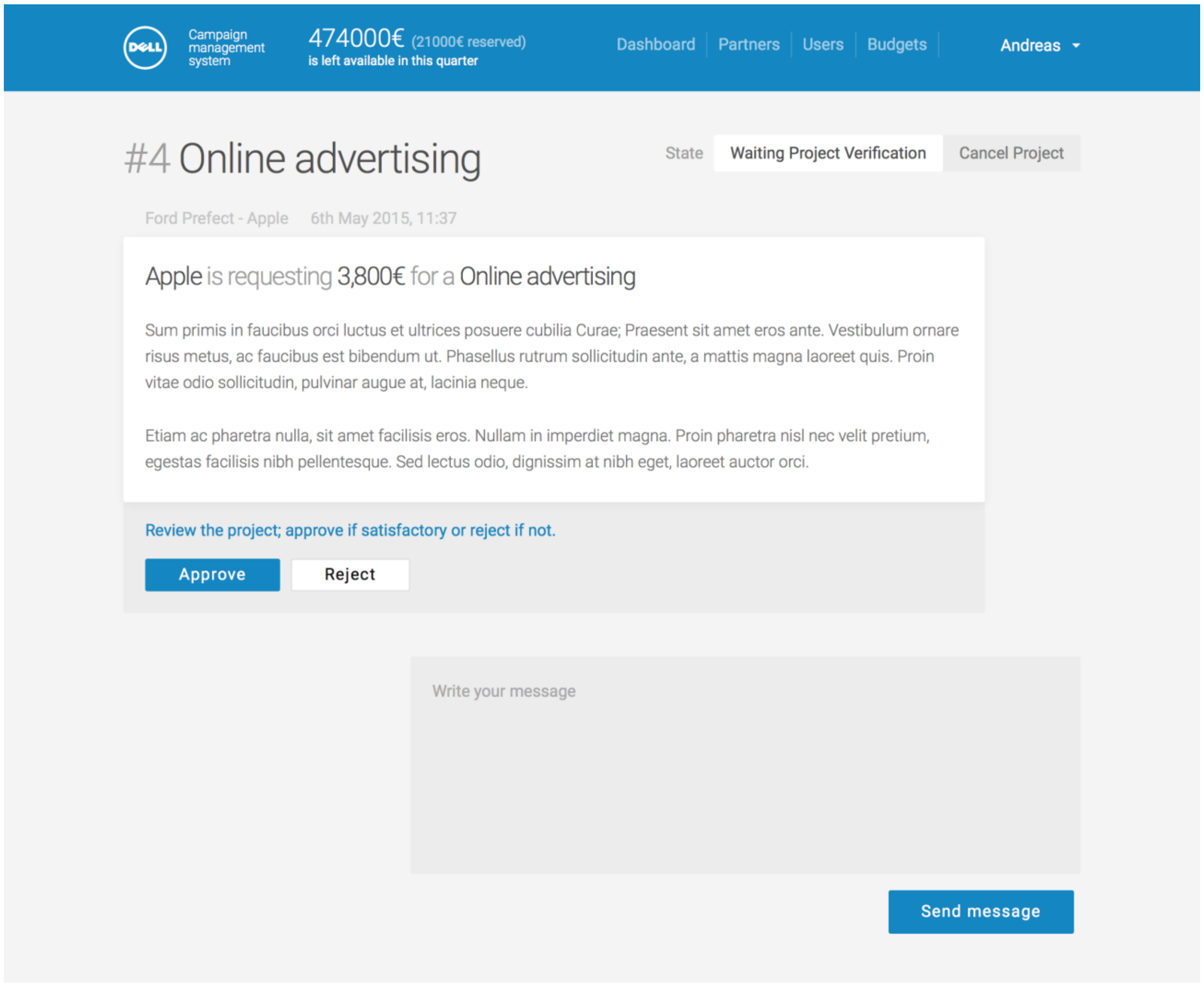


Login instructions

Please fill-in login credentials. Testing login credentials for DELL access are:
Email: honason@gmail.com
Password: horse

Credentials for PARTNER access are:
Email: Anden702@gmail.com
Password: hest

Figure 18 - Login page


Campaign management system
474000€ (21000€ reserved)
is left available in this quarter
Dashboard
Partners
Users
Budgets
Andreas
State
Waiting Project Verification
Cancel Project
Approve
Reject
Write your message
Send message

#4 Online advertising

Ford Prefect - Apple 6th May 2015, 11:37

Apple is requesting 3,800€ for a Online advertising

Sum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent sit amet eros ante. Vestibulum ornare risus metus, ac faucibus est bibendum ut. Phasellus rutrum sollicitudin ante, a mattis magna laoreet quis. Proin vitae odio sollicitudin, pulvinar augue at, lacinia neque.

Etiam ac pharetra nulla, sit amet facilisis eros. Nullam in imperdiet magna. Proin pharetra nisl nec velit pretium, egestas facilisis nibh pellentesque. Sed lectus odio, dignissim at nibh eget, laoreet auctor orci.

Review the project; approve if satisfactory or reject if not.

Figure 19 - Project request view in project page

Complex solutions

By Andreas & Lasse

Email notifications

When projects change state we notify relevant parties of changes that require action on their part; for example, if a partner submits a proof of execution, Dell would receive an email notification that a project needs their attention to move to the next step.

We send emails using Google's SMTP server; this is a temporary solution while we do not have our own dedicated mail server available. We use a Gmail account designated to send emails for us as well as Java's mail package. This combination is enough to be able to send emails. To add an extra flair to our emails, we have chosen to use HTML emails instead of plain text. This allows us to style the email more and have a consistent design across all domains, e.g. we use the same header on the website as in the email. Having the same layout in our emails lets us easily change the content and create new email types.

What happens in our sendEmail method is that we first setup a session with properties matching Gmail server settings, we also supply the session with an Authenticator instance of our user login details. We then create an empty message and add the recipient. We set the content type of the message to html so the html is rendered correctly in the recipients email client. And finally we send it.

```
public void sendEmail(String recipient, String subject, String html){

    try{
        final String fromEmail = "automessagejava@gmail.com";
        final String password = "sappword123";
        final String toEmail = recipient;

        Properties props = new Properties();
        props.put("mail.smtp.host", "smtp.gmail.com"); //SMTP Host
        props.put("mail.smtp.port", "587"); //TLS Port
        props.put("mail.smtp.auth", "true"); //enable authentication
        props.put("mail.smtp.starttls.enable", "true"); //enable STARTTLS
        props.put("mail.smtp.ssl.trust", "smtp.gmail.com");


        Authenticator auth = new Authenticator() {
            //override the getPasswordAuthentication method
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(fromEmail, password);
            }
        };
        Session session = Session.getInstance(props, auth);

        MimeMessage message = new MimeMessage(session);
        message.setFrom(new InternetAddress(fromEmail));
        message.addRecipient(Message.RecipientType.TO, new
InternetAddress(toEmail));

        message.setSubject(subject);


        message.setContent(html, "text/html; charset=utf-8");
        Transport.send(message);
    }catch(Exception ex){
        System.out.println(ex);
    }
}
```

To be compatible with most email clients, the HTML structure is terrible because of the unfortunate and outdated nature of HTML emails.

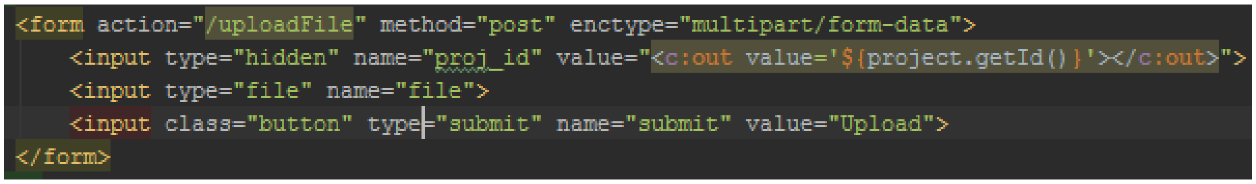
Upon having our own email sender it would most likely be a simple change of options for it to work.

File Handling

Being able to upload proofs of execution is a vital for Dell to be able to judge whether a project has been executed in a satisfactory manner. To facilitate this, we have put in place a file handling system which allows partners to upload files and for Dell to view or download them.

Uploading files

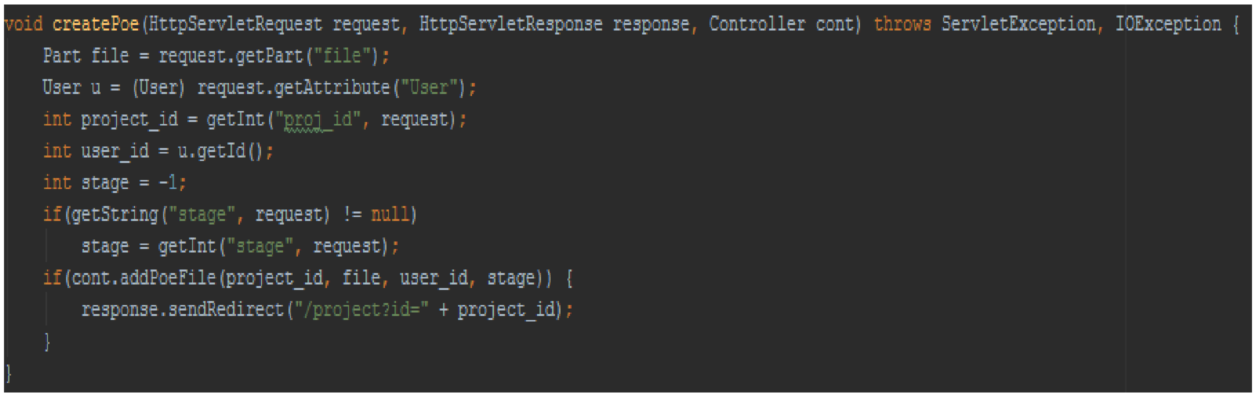
Users can upload files in browser by using a special purpose form for file processing; we can then receive this file in servlet and pass it to our controller. This is what the input form looks like:



```
<form action="/uploadFile" method="post" enctype="multipart/form-data">
    <input type="hidden" name="proj_id" value="<c:out value='${project.getId()}'></c:out>">
    <input type="file" name="file">
    <input class="button" type="submit" name="submit" value="Upload">
</form>
```

Figure 20 - input form in browser

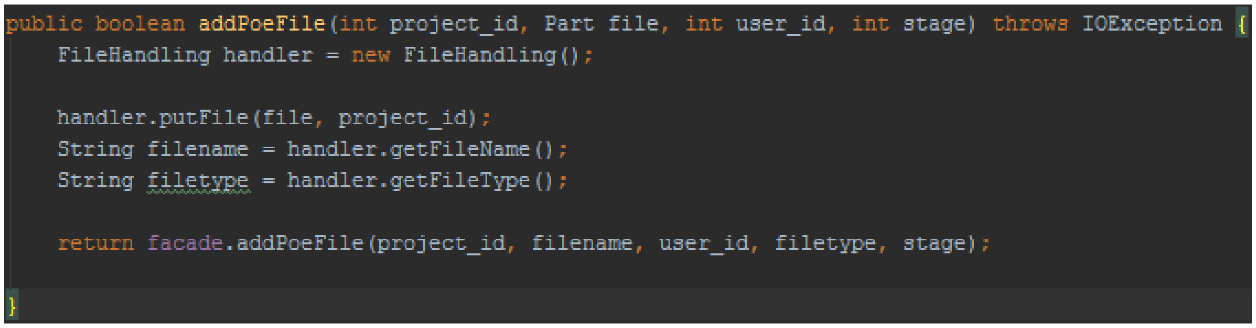
The input form sends the file to our servlet:



```
void createPoe(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException, IOException {
    Part file = request.getPart("file");
    User u = (User) request.getAttribute("User");
    int project_id = getInt("proj_id", request);
    int user_id = u.getId();
    int stage = -1;
    if(getString("stage", request) != null)
        stage = getInt("stage", request);
    if(cont.addPoeFile(project_id, file, user_id, stage)) {
        response.sendRedirect("/project?id=" + project_id);
    }
}
```

Figure 21 - 'createPoe' method in servlet - receives the file from the input form

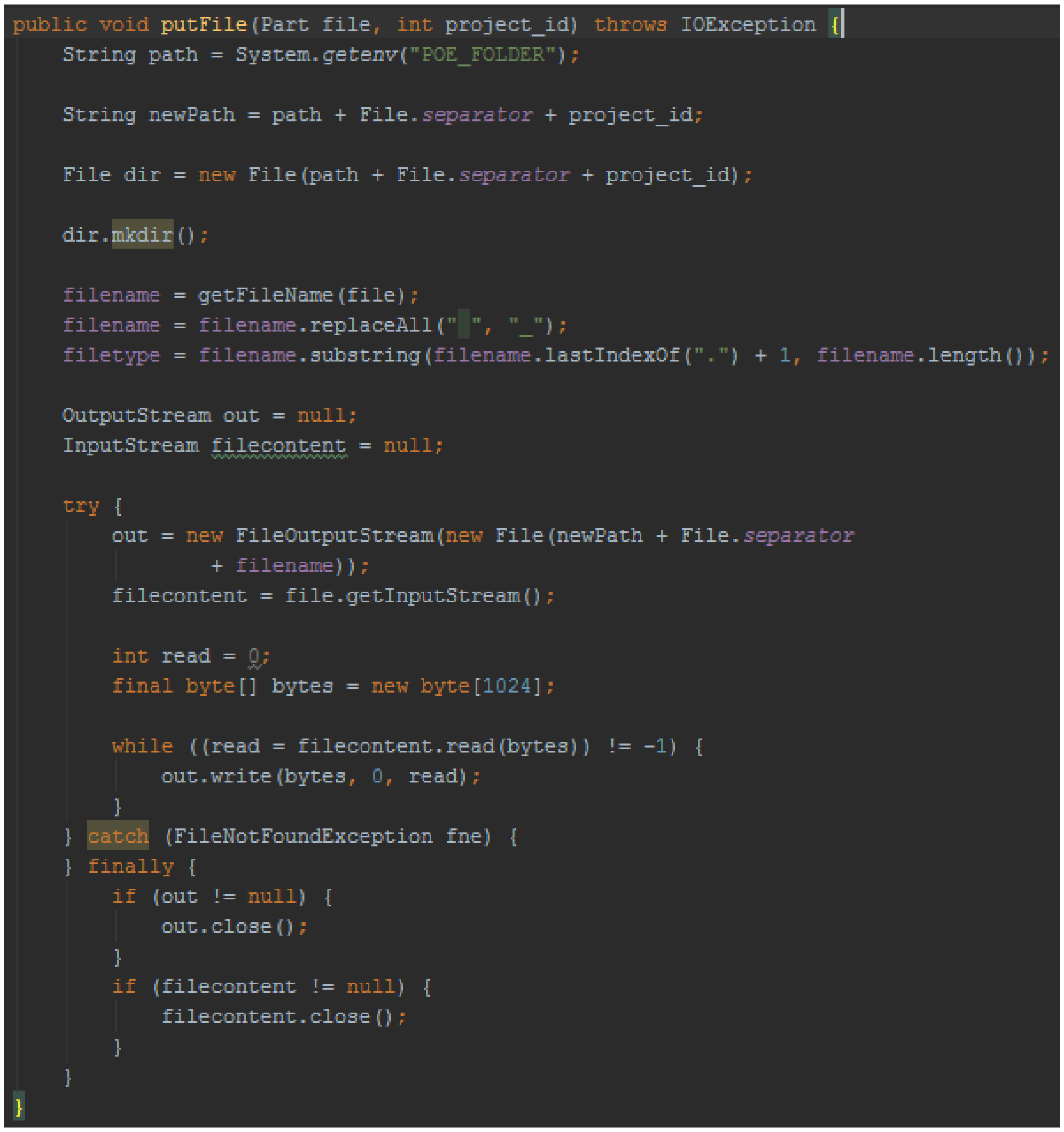
We collect the file (of type 'Part') and pass it to ‘Controller’. The ‘createPoe’ method makes a call to the controller method ‘addPoeFile’ which can be seen below:



```
public boolean addPoeFile(int project_id, Part file, int user_id, int stage) throws IOException {
    FileHandling handler = new FileHandling();

    handler.putFile(file, project_id);
    String filename = handler.getFileName();
    String filetype = handler.getFileType();

    return facade.addPoeFile(project_id, filename, user_id, filetype, stage);

}
```

Figure 22 - 'addPoeFile' method in Controller

The ‘addPoeFile’ method in the controller first instantiates new object of the ‘FileHandling’ class; within this class we save the file to a directory and extract necessary database values (filename + file type). This is done using the ‘putFile’ method:



```java
public void putFile(Part file, int project_id) throws IOException {
    String path = System.getenv("POE_FOLDER");

    String newPath = path + File.separator + project_id;

    File dir = new File(path + File.separator + project_id);

    dir.mkdir();

    filename = getFileName(file);
    filename = filename.replaceAll(" ", "_");
    filetype = filename.substring(filename.lastIndexOf(".") + 1, filename.length());

    OutputStream out = null;
    InputStream filecontent = null;

    try {
        out = new FileOutputStream(new File(newPath + File.separator
                + filename));
        filecontent = file.getInputStream();

        int read = 0;
        final byte[] bytes = new byte[1024];

        while ((read = filecontent.read(bytes)) != -1) {
            out.write(bytes, 0, read);
        }
    } catch (FileNotFoundException fne) {
    } finally {
        if (out != null) {
            out.close();
        }
        if (filecontent != null) {
            filecontent.close();
        }
    }
}
```

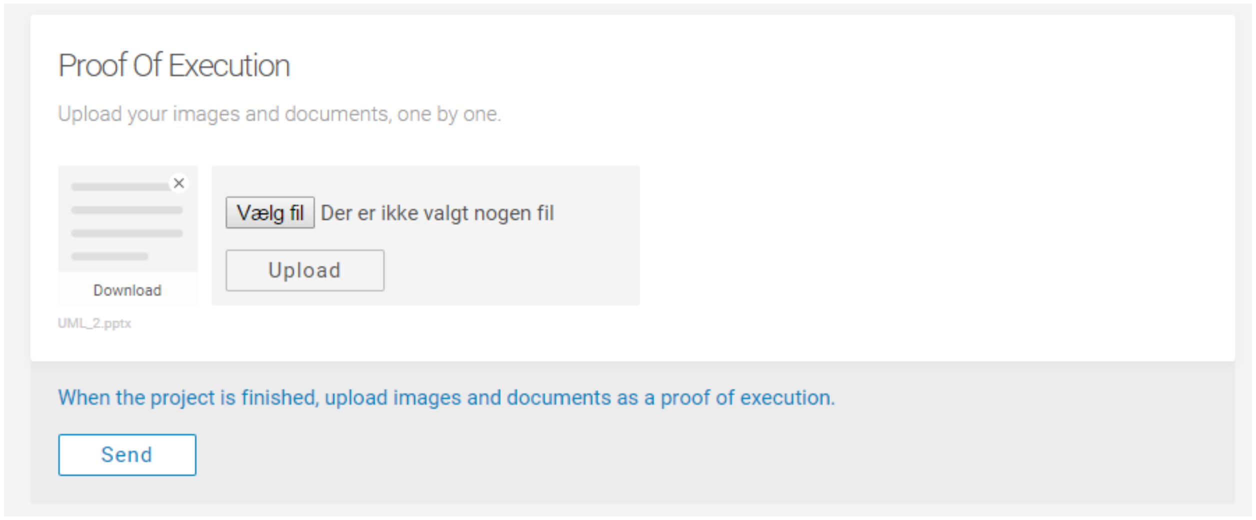
Figure 23 - putFile method

The method does series of things:

1. Retrieves root directory for saving proofs of execution
2. Creates new directory within root directory named by project id the file is associated with unless the directory already exists (in which case nothing happens)
3. Saves the file in the new/existing directory

Once the file has been added to a directory, we save its properties to our database. The most important things to save for later are the filename and project id; we need these values to reconstruct the file path when user wants to view all POE files associated with project.

View of the finished upload file solution in frontend:


Proof Of Execution

Upload your images and documents, one by one.

When the project is finished, upload images and documents as a proof of execution.

Figure 24 - Proof of execution upload view

POE directory:

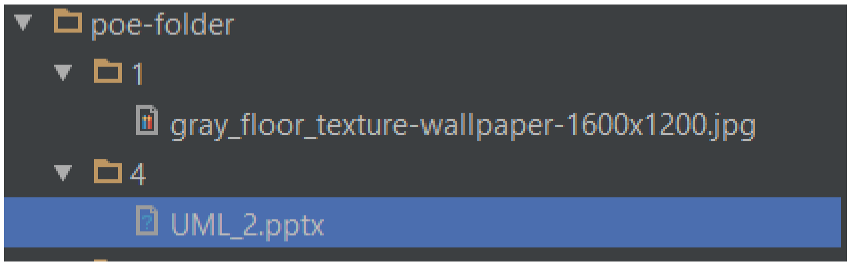
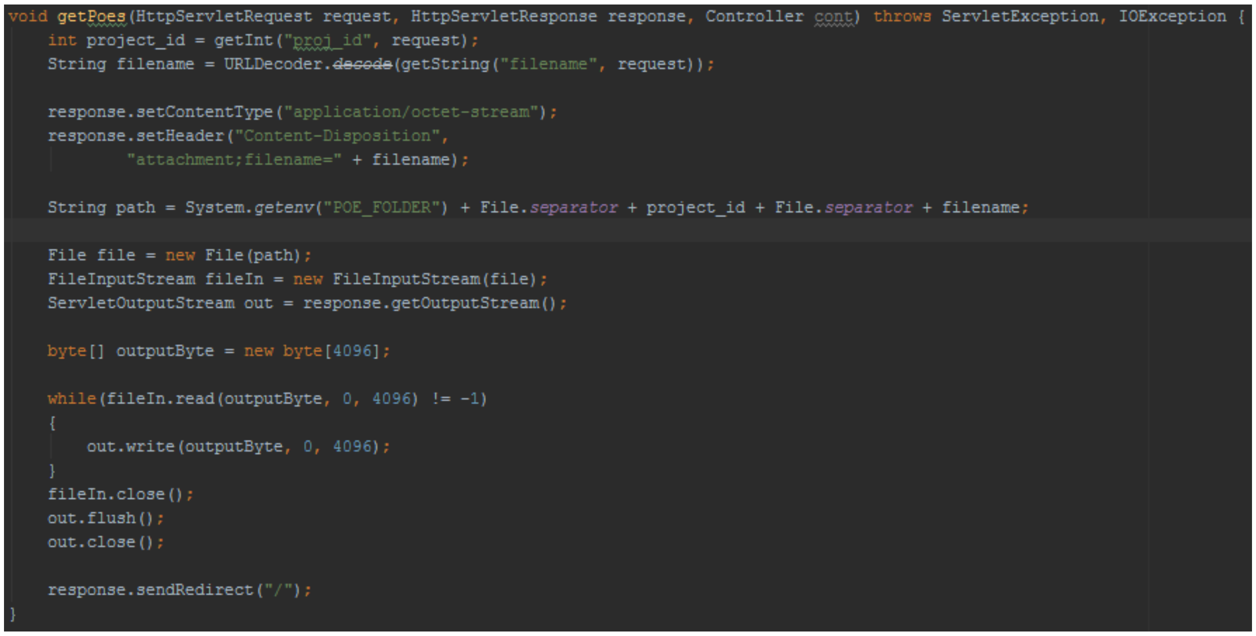

```
poe-folder
  1
    gray_floor_texture-wallpaper-1600x1200.jpg
  4
    UML_2.pptx
```

Figure 25 - POE directory for storing POE files of projects

In this case we can see that projects with id 1 and 4 both have a POE file associated with them. When you see the directory it is easier to see how we reconstruct the file path; you have the root destination, you have the project id, and then you have the filename - when you combine them you get the full path.

Downloading files

Once POE files have been stored within our system, we need to add functionality for Dell users to retrieve them.



```
void getPoes(HttpServletRequest request, HttpServletResponse response, Controller cont) throws ServletException, IOException {
    int project_id = getInt("proj_id", request);
    String filename = URLDecoder.~~decode~~(getString("filename", request));

    response.setContentType("application/octet-stream");
    response.setHeader("Content-Disposition",
            "attachment;filename=" + filename);

    String path = System.getenv("POE_FOLDER") + File.separator + project_id + File.separator + filename;

    File file = new File(path);
    FileInputStream fileIn = new FileInputStream(file);
    ServletOutputStream out = response.getOutputStream();

    byte[] outputByte = new byte[4096];

    while(fileIn.read(outputByte, 0, 4096) != -1)
    {
        out.write(outputByte, 0, 4096);
    }
    fileIn.close();
    out.flush();
    out.close();

    response.sendRedirect("/");
}
```

Figure 26 - getPoes method for retrieving POE files.

The most important thing to note in this method is how we reconstruct the file path. The POE view in the browser is generated by a call to the database that gets every POEs associated with the selected project (in the form of an ArrayList of Poe objects) . You can receive the filename from these objects.

When you click the download button on any POE a form will send the associated filename and project id to the servlet. The complete file path is then reconstructed within the servlet such that the file can be retrieved from the POE directory through an input stream and then downloaded by the user through an output stream.

Login

Despite being advised not to pursue a functional login system, we went above and beyond and did it anyway. We believed that it would be a quick job and that it would add enough value based on the time spent. The principle of secure login and login database is very sound and simple. In short, we are using salted and hashed passwords.

The first thing we do when we store the password is creating a random salt for the password. A salt is just a string of random characters within the range of hex characters. We utilize java's security package (java.security) and its SecureRandom[1] class, which upholds to a number of different standards within cryptography and security, which means the pseudorandom generator practically cannot be predicted. We have chosen to use a salt with the length of 12 characters, we therefore use the SecureRandom class to generate 6 random

[1] https://docs.oracle.com/javase/7/docs/api/java/security/SecureRandom.html

bytes, because each hexadecimal digit represent 4 bits, and 4bits*12=48bits=6bytes. We convert these bytes to hex characters and we have our salt.

```
Random r = new SecureRandom();
byte[] s = new byte[6];
r.nextBytes(s);
String salt = byteArrayToString(s);

public static String byteArrayToString(byte[] a) {
  char[] HEX_CHARS = "0123456789ABCDEF".toCharArray();

  StringBuilder sb = new StringBuilder(a.length * 2);
  for (byte b : a) {
      sb.append(HEX_CHARS[(b & 0xF0) >> 4]);
      sb.append(HEX_CHARS[b & 0x0F]);
  }
  return sb.toString();
}
```

We then prepend this salt to the cleartext, this new string is then hashed using SHA-256, we are again utilizing java's security package and its MessageDigest[2] class which has a method to digest byte arrays. We then convert the concatenated salt and password to a byte array and digest the array. We now have a byte array of 256 bytes, we use the same method (byteArrayToString()) as above to convert it to a hexadecimal representation of the bytes.

```
stringToHash(salt + pw));

public static String stringToHash(String s) {
  byte[] hash = null;
  try {
      MessageDigest digest = MessageDigest.getInstance("SHA-256");
      hash = digest.digest(s.getBytes("UTF-8"));
  } catch (Exception E) {};

  return byteArrayToString(hash);
}
```

We now have a string of 64 characters, created from the salt and password. We now store the original unhashed salt and the hashes salt and password in the database in the format like this: [SALT$HASHEDSALTANDPASSWORD], this string is 77 characters long, the first 12 is the original salt follow by a dollar sign and then the hash. To test a login attempt against this string we will take the salt, which is the first 12 characters, prepend it to the password that

[2] http://docs.oracle.com/javase/7/docs/api/java/security/MessageDigest.html

has been attempted to login with and hash it, if this new hash matches the stored hash it is then the same password and user will be granted access.

```java
public static boolean testPassword(String pw, String storedpw) {
  String salt = storedpw.substring(0, 12);
  return storedpw.equals(salt + "$" + stringToHash(salt + pw));
}
```

Why this method is secure

Just hashing passwords will prevent unauthorized access to the data to use these passwords as they are scrambled, and there is no way to unhash these hashed passwords. Unfortunately for this method there are a things called Rainbow tables, which are tables with the original string, and then the hashed string associated with it. There exists tables for every major hashing function and the list is endless and ever expanding, so if your password happens to have been randomly generated by these computers generating the Rainbow tables, then it is easily looked up and it can be used to gain access to your account.

Salting passwords before hashing makes Rainbow tables useless. Another point is that even if users have the same password, they will have a different hash and salt.

Original password	Salted and hashed password in database
hest	9C86C48E2FA0$436DB060A23E5AA03871B00B79A2C23DF35795D07094E63FF0DEF739B7671370
hest	A70122F9F514$B234134D5F21FEA6012C2D7FFD454E16E1D6A7330DBAA60FD3B2203572DE0B2A
hest	1DDF5BD21D86$4CC225DE3BFCC41D4DE34516DCC793F57072E25F4150389EC2872BB7D1C9B529

If a hacker gains access to password stored in a salted and hashed format, they will have to bruteforce their way specifically for each password. Assuming the average password length is 10 characters, the computer used to bruteforce could manage 3000 million calculations per second[3], which would require a very expensive computer, and that the password only utilizes lower- and uppercase characters as well as numbers (62 possibilities) the time to calculate would be as follows:

[3] http://blog.codinghorror.com/speed-hashing/ (search for 3110.2 M/s)

$$\frac{\frac{62^{10}}{3000M\ c/s}}{60s/60m/24h/365y} = 8{,}87\text{y}$$

Nearly 9 years, which is too long for it to still be relevant.

However, if somebody has access to the database, they would not need a password to gain information, they could just look in a different table for information, but you owe it to your users to keep their password safe. The next step would be to encrypt important data in the database, like bank account details or important numbers like budget.

Once a user have successfully logged in we are using javas session handler to save the logged in state, this will create a cookie with unique id in the users browser which the session handler uses to identify the user again. We are storing a User object with the users information to avoid having to get all this information from the database on each page. We have a logout button, which just deletes the cookie and the session object, the user can then login again.

Search

We have prioritized the search feature because we believe it's very beneficial for the user. We have spent a good amount of time thinking about user's experience, and one of those experiences is finding what he's looking for. That is why we have filters as the first step in finding what you want, and if that approach is fruitless you would resort to the search feature. The search functions in two layers; typeahead and complete search.

Typeahead

The typeahead is a feature that searches as you type. It is well known from Google's search engine, as soon as you begin typing it will recommend common search phrases. We have simplified version of that, we are utilizing Twitter's publicly available JavaScript library which is called Typeahead.js[4]. We have set it up to work like this: after you have typed in 3 letters in the search box it will query our server with that small, most likely unfinished word for predictions. The server will respond with a list matching this search phrase and it will be displayed below the search bar, from here you can select one of the suggestions and it will redirect you as if you had searched for the whole selected phrase. When you continue typing beyond 3 letters, it will continuously filter the list to only match what you are typing, the filtering is done on client side for both speed and sparing the database some work. The typeahead will not search in the whole database but instead some of the most useful data.

[4] https://twitter.github.io/typeahead.js/

The typeahead will search in 4 lists; History, Statuses, Type and Partners.

History

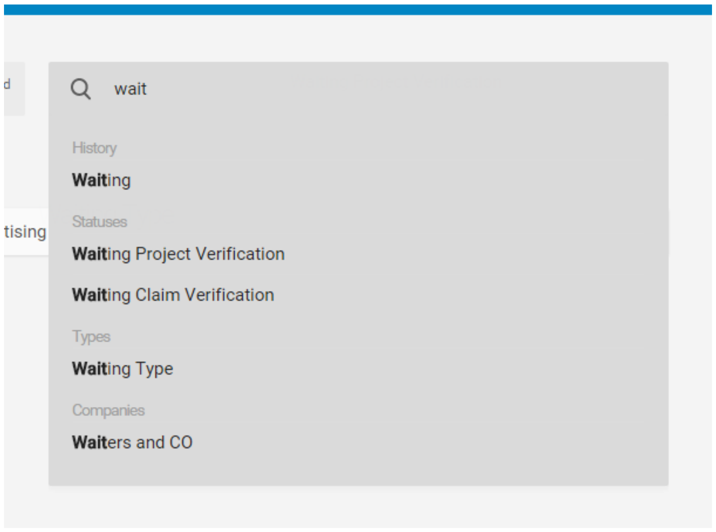


Figure 27 - Typeahead example

The first thing it will search in is your history of searches, every time a user completes a search, that search is saved on client side in his or hers localStorage, from this we can pull all his recent searches and show them.

Statuses

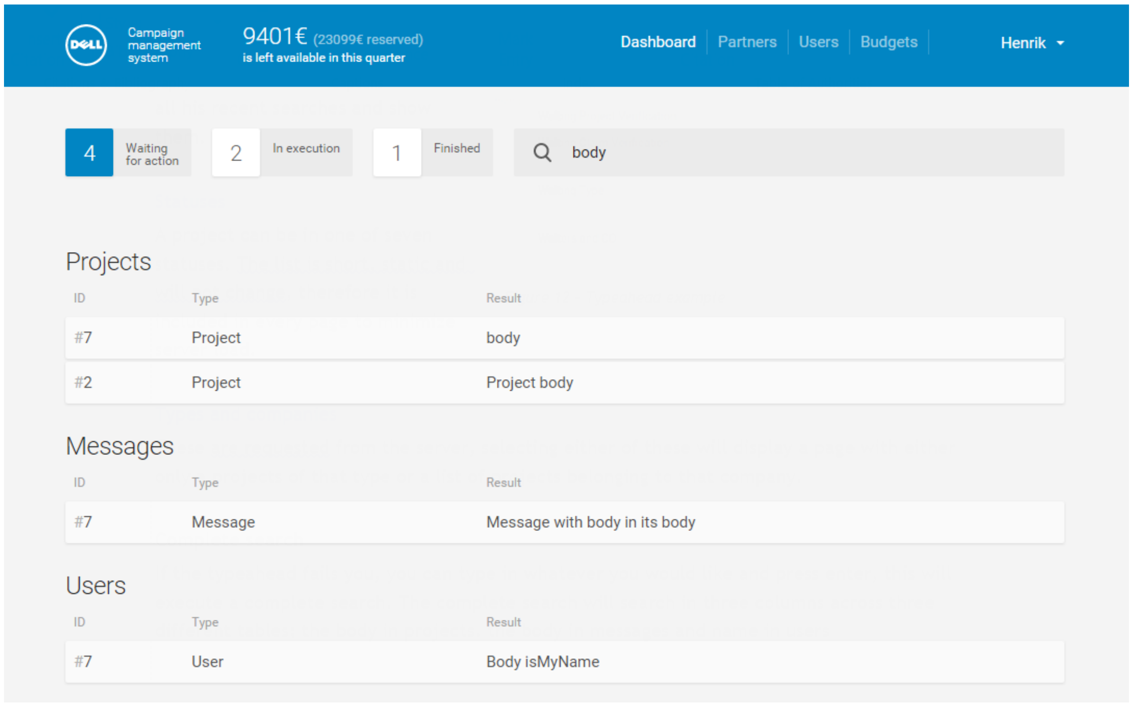
A project can be in one of seven statuses. The list is short, static and will not change, therefore it is included in every page to minimize server load.

Types and partners

These are requested from the server, selecting one of these will display a page with either only a projects of that type or a list of projects belonging to that partner.

Complete search

If the typeahead fails you, you can type in whatever you would like and press enter, this will execute a complete search. The complete search will search in three columns across three different tables; the body in projects, the body in messages and name in users. The results are ordered by relevance to your search, we are using an Oracle-only feature which is called “utl_match.jaro_winkler_similarity”, it is an algorithm to determine the similarity between two strings, we are comparing the original search phrase to the result and ordering it by result of this algorithm. At this early stage it is useful to only order by this but later on when the database is much larger it might be less relevant if the top result is a year old, therefore we would like to implement a second weight to the ranking which is pulled from the date of the item.



Projects

ID	Type	Result
#7	Project	body
#2	Project	Project body

Messages

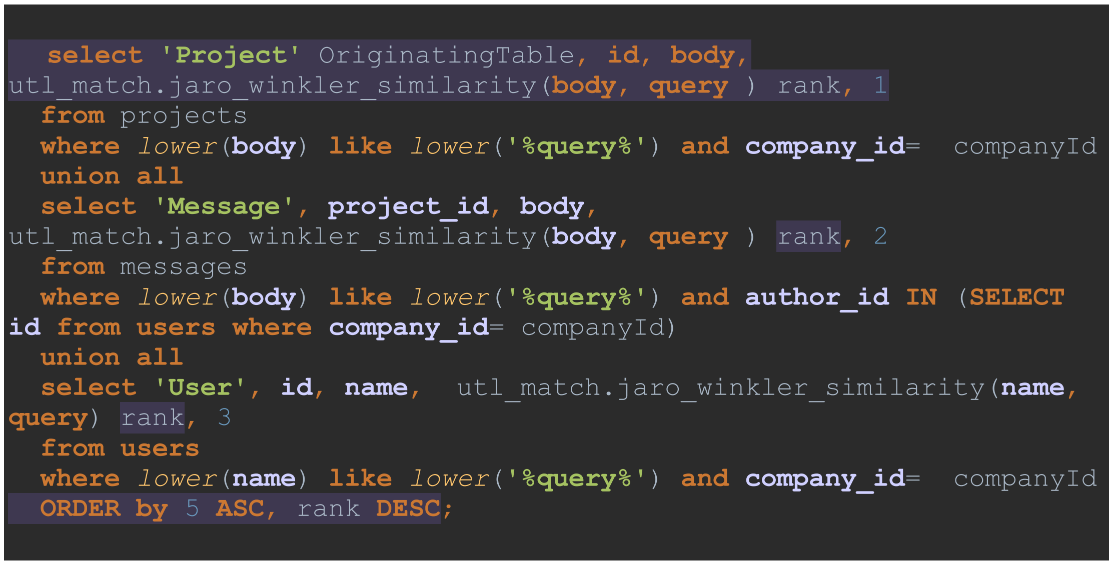
ID	Type	Result
#7	Message	Message with body in its body

Users

ID	Type	Result
#7	User	Body isMyName

Figure 28 - Page view of a complete search

SQL for the complete search is quite simple, it is 3 different selects combined together in one resultset with the columns: Source, id, result and ranking. The source is which table the result is from, it will either be ‘Project’, ‘Message’ or ‘User’. Id is the id of the result, this is used to be able to navigate from the search page to its corresponding individual page. Result is the cell in the database in which it found a match to the search phrase. Ranking is the jaro winkler score that we are ordering by.



```
  select 'Project' OriginatingTable, id, body,
utl_match.jaro_winkler_similarity(body, query ) rank, 1
  from projects
  where lower(body) like lower('%query%') and company_id=  companyId
  union all
  select 'Message', project_id, body,
utl_match.jaro_winkler_similarity(body, query ) rank, 2
  from messages
  where lower(body) like lower('%query%') and author_id IN (SELECT
id from users where company_id= companyId)
  union all
  select 'User', id, name,  utl_match.jaro_winkler_similarity(name,
query) rank, 3
  from users
  where lower(name) like lower('%query%') and company_id=  companyId
  ORDER by 5 ASC, rank DESC;
```

Mapping users and partners

On the project view page we have messages and there could potentially end up being a lot of message from different users on the project. On the page we need to display the name of the author of each message as well as the partner's name of this user. Each of these message objects have an author_id which represent the id of the user who wrote the message. Then we need to go to the database for more information about these messages. Because of the high probability of duplicate authors and partners it would be a waste to go the database for a user if we already have gotten that user previously for another message. We have therefore created a method that solves this problem.

Initially the method will create two HashMaps, one for users and another for partners. We then start looping through the messages, if the author_id is not in the usermap we'll have to go to the database for that, after we have the user we then store it with it's ID in the user HashMap. Next time we need to get a user with the same id, it will already exist in the HashMap and we don't need to go the database for it. The same process is done for partners.

```
public ArrayList processMessages(ArrayList messages) {
    HashMap companyMap = new HashMap();
    HashMap userMap = new HashMap();
    User user = null;
    Company company = null;
    for (Message m : (ArrayList<Message>) messages) {
        if(userMap.containsKey(m.author_id))
            m.user = (User) userMap.get(m.author_id);
        else {
            user = facade.getUserById(m.author_id);
            m.user = user;
            userMap.put(m.author_id, user);
            company = facade.getCompanyById(user.getCompany_id());
            companyMap.put(user.getCompany_id(), company);
        }
        if(m.company == null) {
            if(companyMap.containsKey(m.user.getCompany_id()))
                m.company = (Company)
companyMap.get(m.user.getCompany_id());
            else {
                company =
facade.getCompanyById(m.user.getCompany_id());
                m.company = company;
                companyMap.put(company.id, company);
            }
        }
    }
    Collections.sort(messages, Message.TIME);
    return messages;
}
```

SQL queries

By Jan

All SQL queries, which are used to communicate with Oracle database, are located in data mapper classes, in data source layer. We are using PrepareStatement class to be able to quickly repeat queries execution without hurting performance.

Method modifyBudget shown below modifies initial budget amount for specified quarter. SQL query targets the row by composite primary key - year and quarter. In this case, composite primary key is used to prevent setting multiple budgets to one quarter.

```
public boolean modifyBudget(int new_budget, int year, int quarter, Connection con) {
    String SQL = "update budget set initial_budget=? where yearnum = ? and quarternum = ?";
    PreparedStatement statement = null;

    try {
    statement = con.prepareStatement(SQL);
        statement.setInt(1, new_budget);
        statement.setInt(2, year);
        statement.setInt(3, quarter);
        statement.executeUpdate();

    }
    // catching exceptions
}
```

Transactions

For groups of SQL queries that must be either fully completed or rolled back to initial state in case of error, we are using transactions.

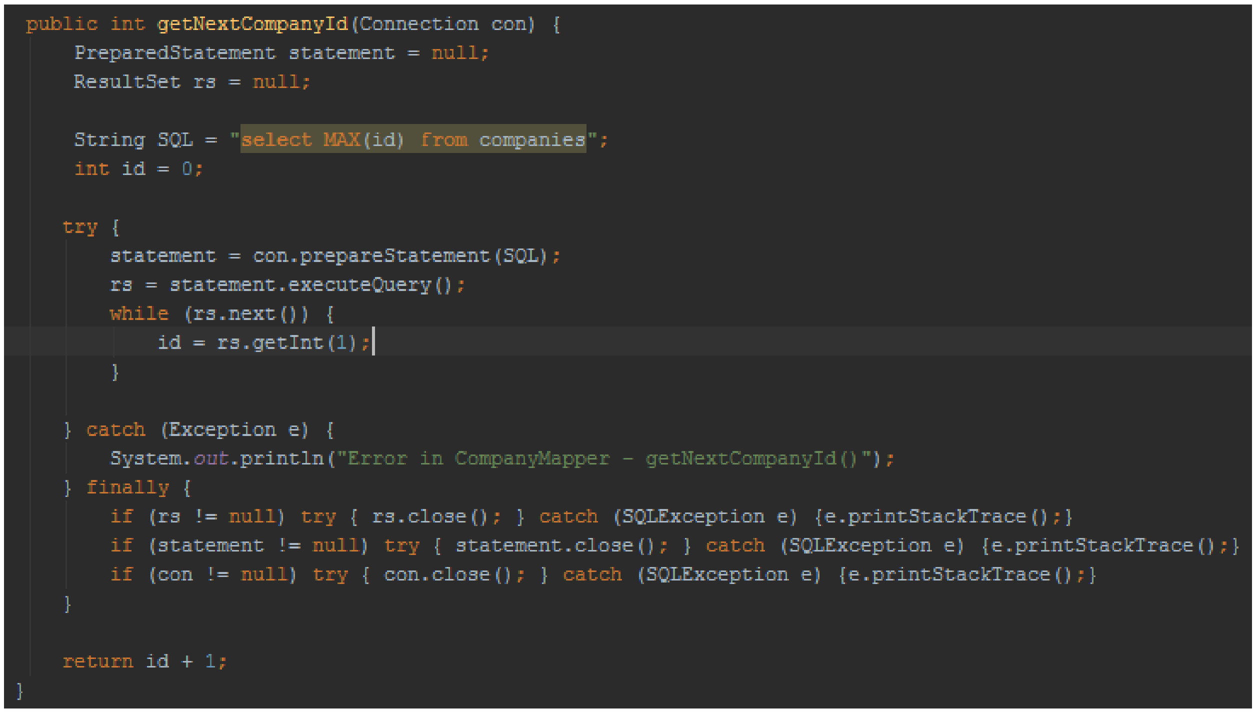
In the following example, when Dell user approves the project, project’s status needs to be changed, and project’s budget needs to be allocated in “reserved” section of Dell’s quarter budget. If any of these queries fails, others must be cancelled or rolled back.

```
try {
    // TRANSACTION BEGIN
    con.setAutoCommit(false);
    // PROJECT CHANGES
    statement = con.prepareStatement(SQL);
    statement.setString(1, new_status);
    statement.setInt(2, project_id);
    statement.executeUpdate();
    // BUDGET CHANGES
    budgetStatement = con.prepareStatement(SQLBudget);
    budgetStatement.setInt(1, project_budget);
    budgetStatement.setInt(2, currentYear);
    budgetStatement.setInt(3, currentQuarter);
    budgetStatement.executeUpdate();
    // COMMIT
    con.commit();
    // TRANSACTION OVER
    updateChangeDate(project_id, companyId);
    addStage(userId, project_id, new_status, DatabaseConnection.getInstance().getConnection());
    DatabaseFacade facade = new DatabaseFacade();
    if (companyId == 1)
        facade.markUnread(project_id, 2); // 2 is just not dell, a la partner
    else
        facade.markUnread(project_id, 1);

    return true;
}
```

Our biggest regret is our underuse of transactions; there is potential for bad data in some areas of our code in case of database error, and this is attributed solely to the fact that we sporadically make database insertions/updates that depend on another on two different connections. We have ensured that anywhere the budget has to change we use transactions; it's simply too important to ensure consistency within the budget to forgo it.

Auto increment in SQL

Newer versions of Oracle Database support auto incrementing values in tables; our version does not include this feature, therefore we have to manually retrieve the highest value of an id and increment it manually when we want new data base inserts. This is not a very complex solution, but we thought it was worth demonstrating our workaround to this problem. This is how it looks in our code:



```java
public int getNextCompanyId(Connection con) {
    PreparedStatement statement = null;
    ResultSet rs = null;

    String SQL = "select MAX(id) from companies";
    int id = 0;

   try {
       statement = con.prepareStatement(SQL);
       rs = statement.executeQuery();
       while (rs.next()) {
           id = rs.getInt(1);
       }

   } catch (Exception e) {
       System.out.println("Error in CompanyMapper - getNextCompanyId()");
   } finally {
       if (rs != null) try { rs.close(); } catch (SQLException e) {e.printStackTrace();}
       if (statement != null) try { statement.close(); } catch (SQLException e) {e.printStackTrace();}
       if (con != null) try { con.close(); } catch (SQLException e) {e.printStackTrace();}
   }

   return id + 1;
}
```

Figure 29 - auto-incrementing method getNextCompanyId

Features

By Jan & Andreas

Added extra functionality can be useful addition or a gimmick that clutters the system. That is why our team have spent significant time on deciding what functions are beneficial and supports project workflow in major way. Features described in this section are made to make interaction with our application faster, easier, more powerful and less demanding.

Typeahead search

Search functionality and its implementation is important part of every system that stores important pieces of data in meaningful relations.

Searchbar in the application is available in the Dashboard view, and can search for all the projects, users and partners by project types, project states, project body text, message's text, user's name and partner's name.

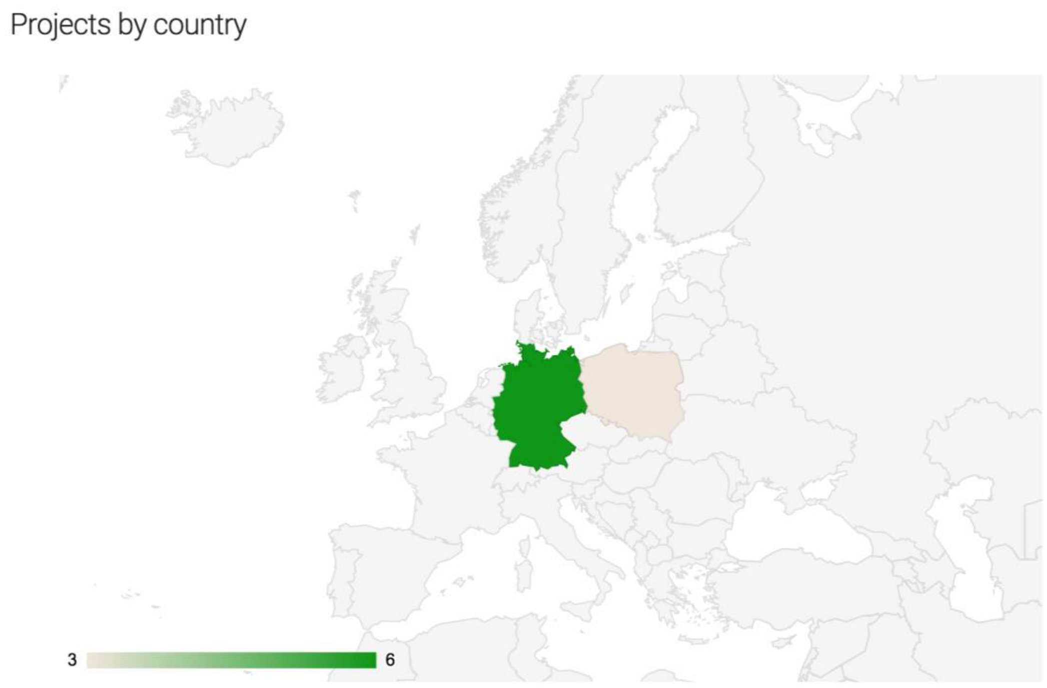
Statistics

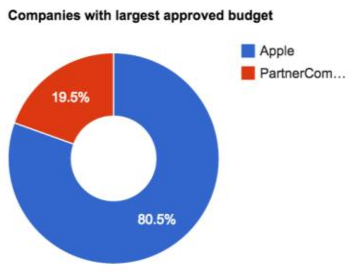
An overview of important data in the application is one of the owner's requirements in this project. Statistics page provides Dell users with comprehensive details about usage of application, budget, projects and partners.

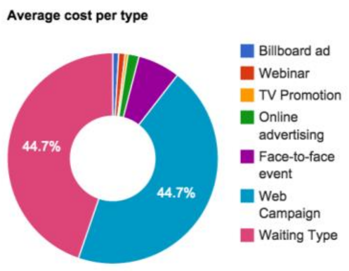
Our statistics have focus mostly on budget properties, and some examples of collected data are:

1. Graph of budget progression – in quarter
2. Companies with largest approved budget – per quarter
3. Projects by country – per quarter
4. Average cost per project type – per quarter
5. Count of projects for project type – per quarter
6. Number of finished projects – all time
7. Money reimbursed – all time

All of this data can be analyzed and be used in Dell's favor, the possibilities are endless. You could figure out which country is lacking in advertising and put more focus on that country, or you could crosscheck it with the amount of Dell products bought in that country and get exciting intel to improve business strategy.







Email messages

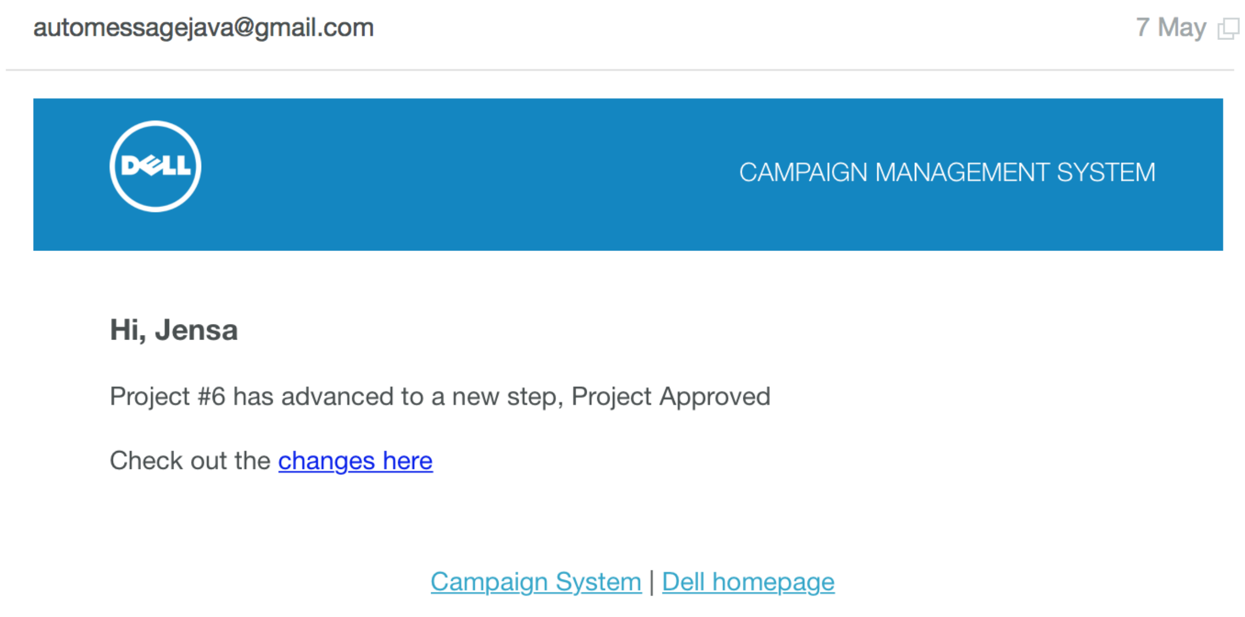
Password reset

Whenever a user is created, we don't set a password for the new user, but we do set his or hers email. Then we send an email to this user welcoming him or her to the application with a link to password creation. This link will only work once and is invalid as soon as the password is reset.

Notifications

Application acts as an active intermediary between Dell and its partners, for quick workflow there is a need to notify involved parties about important events in project's development. Email notifications proved to be the best solution, as application already knows users' emails, and no further settings are necessary.

Change in project is reported to all users that were previously involved with this project. In this context a user is involved if he or she either created the project, changed a state in it or commented in the project.



automessagejava@gmail.com 7 May

CAMPAIGN MANAGEMENT SYSTEM

Hi, Jensa

Project #6 has advanced to a new step, Project Approved

Check out the changes here

Campaign System | Dell homepage

Budget overview

A number representing the currently available budget in the active quarter is always visible to Dell users in the header of application. It is a quick hint for Dell to recognize what project they can approve to fit into current budget. The amount of reserved money (the budget from projects that are approved, but not yet finished and reimbursed) is also displayed.

Combining this number with the graph of budget progression on the statistics page provides great insight into the budget and will allow Dell to spend their money, which is the goal, more intelligently.

Messaging

The function of messaging in the context of project was previously represented in email conversations, and that way proved ineffective. Our application uses in-project messaging between partners and Dell users. It is fast and unified way to communicate about projects in one place. All Dell users can send messages, and all partner’s users can send messages in partner’s projects. Notification about new message is immediately sent to recipient’s email.

Logo fetcher

Logo image is one of the ways to differentiate partners inside application. We have created an automatic logo fetcher that searches for partner’s company logo on the internet, and presents the choice of four logo images to the user creating new partner in application. An option to upload custom image is also available.

Test activities

By Lasse

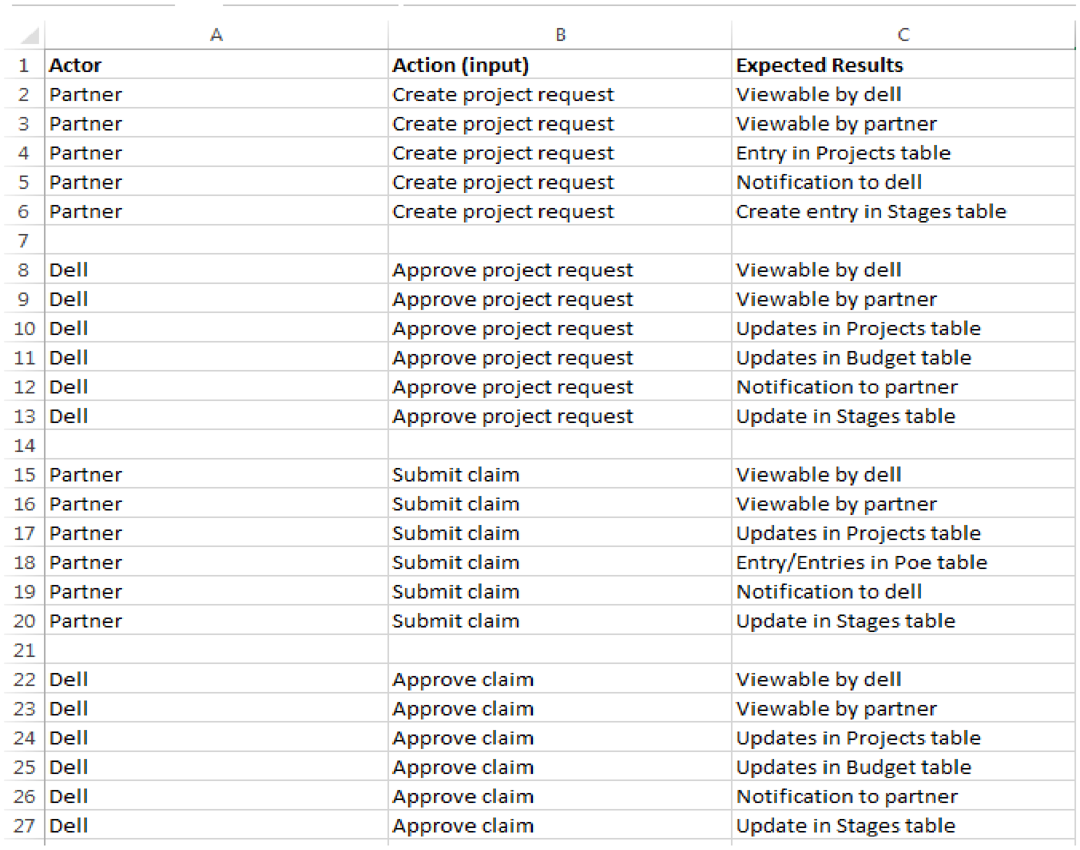
When building any application, it’s important to ensure it works as intended - the significance of things working according to plan is crucial when we are talking about real business cases that involve money or important documents. It would, for the sake of example, be detrimental to the functionality of a business if there are inconsistencies in the data in regards to payments; you could potentially end up in a situation where you owe/are owed money and you’d never be able decipher the details to resolve the problem due to corrupted data.

We are going to (partially) test the most important functionalities of our application in an attempt to guarantee a degree of consistency within our data.

Manual testing

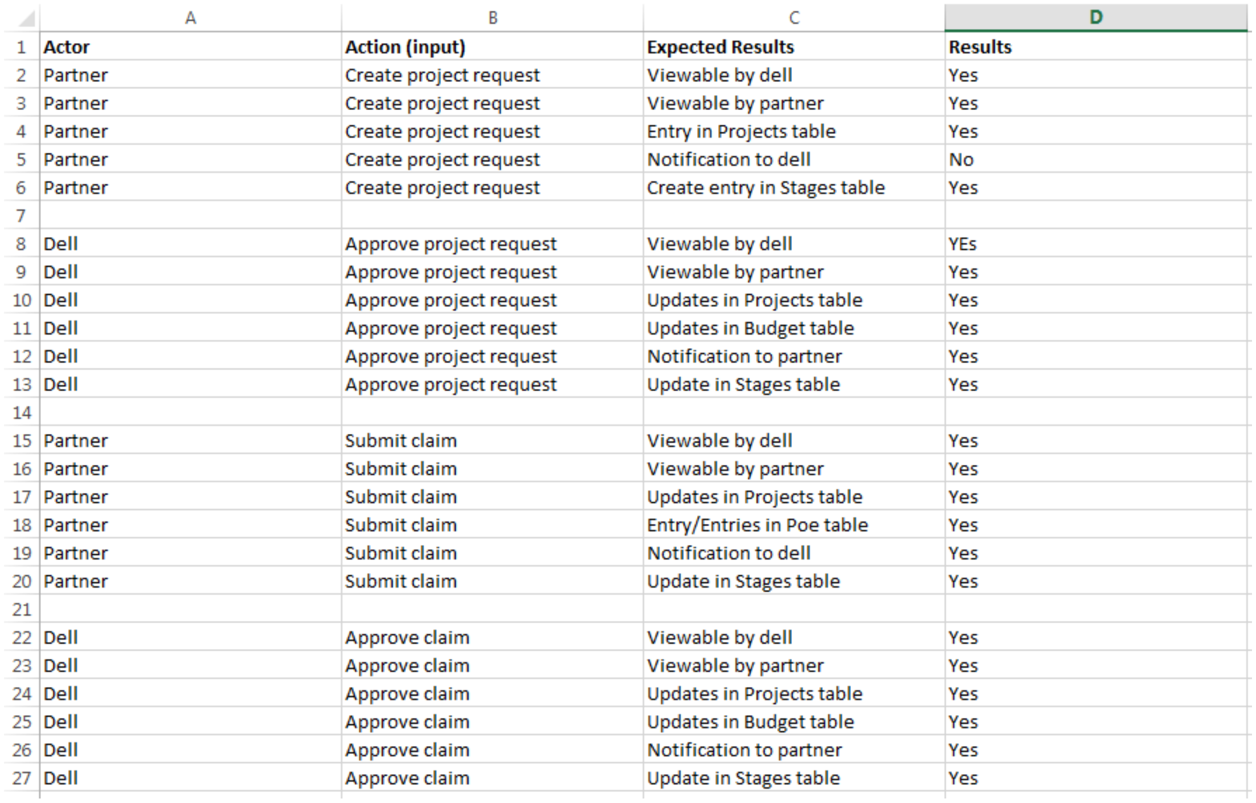
We are going to manually go through and monitor our data as we create a project request and go through the process to completion. We will take extra care to note changes in the budget table to ensure that our data is consistent even when things go wrong. The test assumes that a budget has been created in advance.

Test guidelines:



	A	B	C
1	**Actor**	**Action (input)**	**Expected Results**
2	Partner	Create project request	Viewable by dell
3	Partner	Create project request	Viewable by partner
4	Partner	Create project request	Entry in Projects table
5	Partner	Create project request	Notification to dell
6	Partner	Create project request	Create entry in Stages table
7			
8	Dell	Approve project request	Viewable by dell
9	Dell	Approve project request	Viewable by partner
10	Dell	Approve project request	Updates in Projects table
11	Dell	Approve project request	Updates in Budget table
12	Dell	Approve project request	Notification to partner
13	Dell	Approve project request	Update in Stages table
14			
15	Partner	Submit claim	Viewable by dell
16	Partner	Submit claim	Viewable by partner
17	Partner	Submit claim	Updates in Projects table
18	Partner	Submit claim	Entry/Entries in Poe table
19	Partner	Submit claim	Notification to dell
20	Partner	Submit claim	Update in Stages table
21			
22	Dell	Approve claim	Viewable by dell
23	Dell	Approve claim	Viewable by partner
24	Dell	Approve claim	Updates in Projects table
25	Dell	Approve claim	Updates in Budget table
26	Dell	Approve claim	Notification to partner
27	Dell	Approve claim	Update in Stages table

Results:



	A	B	C	D
1	**Actor**	**Action (input)**	**Expected Results**	**Results**
2	Partner	Create project request	Viewable by dell	Yes
3	Partner	Create project request	Viewable by partner	Yes
4	Partner	Create project request	Entry in Projects table	Yes
5	Partner	Create project request	Notification to dell	No
6	Partner	Create project request	Create entry in Stages table	Yes
7				
8	Dell	Approve project request	Viewable by dell	YEs
9	Dell	Approve project request	Viewable by partner	Yes
10	Dell	Approve project request	Updates in Projects table	Yes
11	Dell	Approve project request	Updates in Budget table	Yes
12	Dell	Approve project request	Notification to partner	Yes
13	Dell	Approve project request	Update in Stages table	Yes
14				
15	Partner	Submit claim	Viewable by dell	Yes
16	Partner	Submit claim	Viewable by partner	Yes
17	Partner	Submit claim	Updates in Projects table	Yes
18	Partner	Submit claim	Entry/Entries in Poe table	Yes
19	Partner	Submit claim	Notification to dell	Yes
20	Partner	Submit claim	Update in Stages table	Yes
21				
22	Dell	Approve claim	Viewable by dell	Yes
23	Dell	Approve claim	Viewable by partner	Yes
24	Dell	Approve claim	Updates in Projects table	Yes
25	Dell	Approve claim	Updates in Budget table	Yes
26	Dell	Approve claim	Notification to partner	Yes
27	Dell	Approve claim	Update in Stages table	Yes

Test conclusion

The test shows that no notification to Dell was sent upon project request creation; this is actually intended and is a mistake within the test itself, not the application.

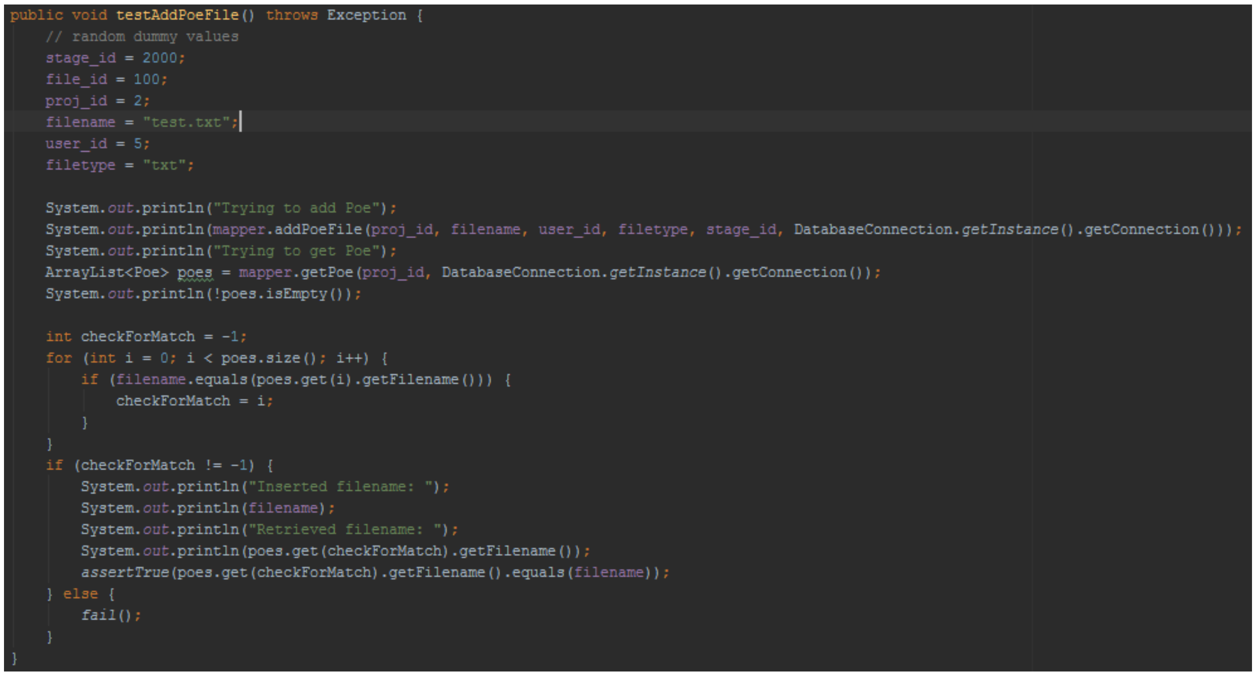
We are especially content with the fact that budget data is intact and working as expected after going through the entire project creation workflow.

Automated Tests (JUnit)

We also have some automated unit tests from the data source layer to test if a chosen set of methods work as intended. We've chosen to test the 'PoeMapper' class, which is a core piece of functionality within the main workflow.

PoeMapper unit test:

Here is the main portion of our unit test in the 'PoeMapper'. A 'PoeMapper' object is assigned before the test is run (@Before annotation in JUnit) and its reference is 'mapper'. We also remove the entry afterwards (@After annotation in JUnit). The full JUnit test can be found in the source code and is called 'PoeMapperTest'.



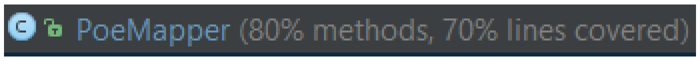
```java
public void testAddPoeFile() throws Exception {
    // random dummy values
    stage_id = 2000;
    file_id = 100;
    proj_id = 2;
    filename = "test.txt";
    user_id = 5;
    filetype = "txt";

    System.out.println("Trying to add Poe");
    System.out.println(mapper.addPoeFile(proj_id, filename, user_id, filetype, stage_id, DatabaseConnection.getInstance().getConnection()));
    System.out.println("Trying to get Poe");
    ArrayList<Poe> poes = mapper.getPoe(proj_id, DatabaseConnection.getInstance().getConnection());
    System.out.println(!poes.isEmpty());

    int checkForMatch = -1;
    for (int i = 0; i < poes.size(); i++) {
        if (filename.equals(poes.get(i).getFilename())) {
            checkForMatch = i;
        }
    }
    if (checkForMatch != -1) {
        System.out.println("Inserted filename: ");
        System.out.println(filename);
        System.out.println("Retrieved filename: ");
        System.out.println(poes.get(checkForMatch).getFilename());
        assertTrue(poes.get(checkForMatch).getFilename().equals(filename));
    } else {
        fail();
    }
}
```

Figure 30 - the test method testAddPoeFile

The test performs following actions:

1. Tries to add a dummy POE file to our database using the 'addPoeFile' method (no actual file is added, we are only testing the 'PoeMapper' class in this test). Returns true if successful, false otherwise.
2. Tries to retrieve the dummy POE data. Makes a check to see if the call returns a non-empty ArrayList of Poe objects; returns true if successful, false otherwise.
3. Checks if the data inserted matches the data received. If it matches the test is deemed successful, otherwise it fails.
4. Deletes the entry (can't be seen in the above method) - this happens after the test has run to ensure our dummy data isn't in the database.

Test coverage:

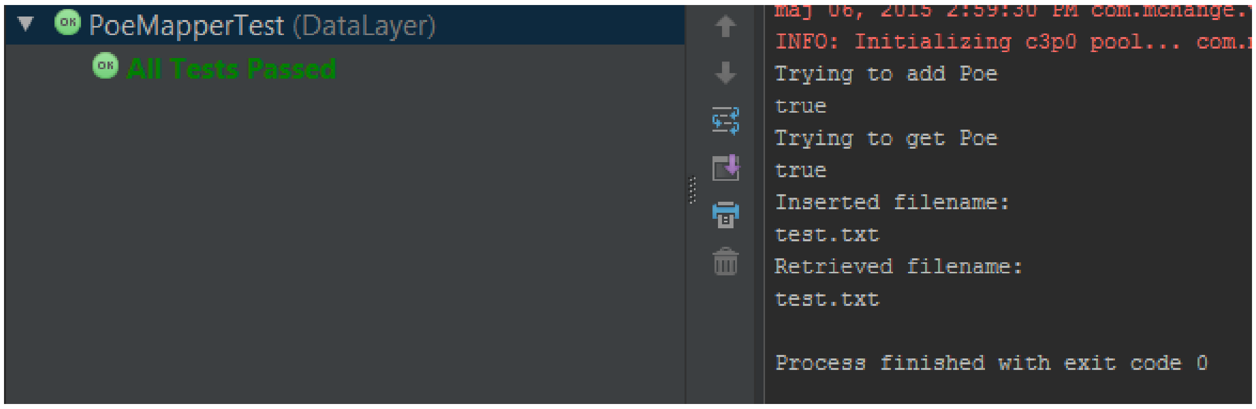


PoeMapper (80% methods, 70% lines covered)

Figure 31 - How much coverage the test has in the "PoeMapper" class

Our test covers 80% of the methods used in 'PoeMapper'; the only method within the 'PoeMapper' that we left untested is the method for marking a POE as deleted, but this is not as essential as the methods we did test.

Test results:



PoeMapperTest (DataLayer)

All Tests Passed

```
maj 06, 2015 2:59:30 PM com.mchange.
INFO: Initializing c3p0 pool... com.
Trying to add Poe
true
Trying to get Poe
true
Inserted filename:
test.txt
Retrieved filename:
test.txt

Process finished with exit code 0
```

Figure 32 - Junit test results

We are pleased with the results; we didn't cover everything within the 'PoeMapper' class but all the important aspects (add POE, get POE and delete POE methods) are covered and working as intended.

Known errors

By Lasse

Our application is not 100% complete; we still have some errors within our application derived from less structured testing that we think are important to note:

You are able to change project statuses without an active budget available

Steps to reproduce:

1. Ensure no budget is available for current quarter
2. Login as Partner and create project request
3. Login as Dell and approve project

Consequences:

- If a budget is later added to the same quarter, you may end up corrupting budget data by making changes in it without accounting for money reserved by approved projects

Suggested solution:

- Check for an active budget before allowing changes in project state

You could potentially corrupt data within the program if internal problems occur

Steps to reproduce:

1. Have a database related error

Consequences:

- Inconsistent data, could lead to significant errors

Suggested solution:

- Increase use of transactions in order to ensure multiple database updates are committed together or not at all

Concerns about transactions

One of our biggest concerns is that we haven't fully encapsulated updates that depend on one another in transactions; things work as intended generally, but we have experienced minor issues when the database server is acting up. It's not outside our realm of capabilities in the sense that we've already used transactions in budget related table updates, but we could definitely have done a better job in some areas of our code. If we were to further develop the application a top priority would definitely be to reconstruct our mapper infrastructure to facilitate more transactions.

Future implementations and considerations

By Andreas

Due to the nature of this development process where we do not have direct access to the product owner, we are unsure about what features they would like implemented and generally how the system should work, but we did have some good ideas. It is our goal that our application will be a full solution and not just solve a part of the problem, these ideas are to help achieve that.

During the development process, we constantly had new ideas for features that could improve the application. Unfortunately, we could not manage to do them all, here is what we would have liked to do and our reason behind the ideas.

Assigning Dell users to partners

A feature that could increase scalability and structure for Dell is assigning Dell users to partners, so that a Dell user will deal with only the same handful of companies. Once a new partner is created they will have one or more Dell users assigned to serve that partner. This will retrieve multiple improvements, one is that partners will always see the same person whenever they communicate with Dell and they have *one* they talk to and trust, this will benefit Dell as well and create some kind of personal connection between the two.

The Dell users projects list will then initially only consist of partners assigned to them, if the list is empty he or she can of course see a list of all projects and assist where needed. Upon a new project request, Dell users who are attached to the projects creator will then receive an email about this.

Product attached to projects

Partners creates a project because they want to promote a certain product or service from Dell; right now, that is nowhere to be seen in the application. We would like to have a page where Dell can manage their current products they want promoted, this list will then be available to the partner whenever they create a project request and they will then pick the appropriate product. This will increase the amount of information Dell is getting and it will be sorted in a better way than if it was to be written in the project body. Additionally, partners could have access to a media library with images of the product so they can use it in their advertisement.

User system for brand partners

Another part of the MDF system is that certain computer part manufacturers, such as Intel, sponsor Dell. These kind of partners would like proof that the projects including their product is living up to their logo and what else requirements they might have. Currently Dell's solution is to manually pick projects, collect the screenshots and then send them to Dell. With the above functionality, it would easily be possible to either create a user type for Intel and partners alike that only has access to projects with their products and the attaches POEs, or a 'Create report' function that would collect all images from projects with each respective brand partner and send an email with the images. In the end, the system should of course be a total solution, and not just a part of it.

Add stage to projects – Reimbursed

Another feature to come close to the final solution is adding a final stage we would like to call 'Reimbursed'. Currently the last stage is 'Project Finished' which happens after the POE has been approved, from this point it is expected that the reimbursement process has begun, but partner will not know when the money arrive.

The idea is that once a project has been marked 'Project Finished', an email is automatically sent to the accounting department of Dell with details of payment, along with this will be a link. The link, which will utilize the nonce feature we already have implemented, will contain a page with a button that will mark the project 'Reimbursed', it is meant to be clicked when the accounting department have confirmed that the payment have been processed. Now partners will receive an email that the reimbursement is officially on its way and can expect it within the next couple of days.

Considerations

By Jan

We have also came through difficulty of decision-making regarding unclearly described sections of the system. Two of which were Budget information sharing, and Currency system.

Current email-driven system counts with Dell's initial activity of providing partners with their available budget. With the information that there are roughly 400 partners in Nordic area, it would be very demanding for Dell to inform all of them about their available budget every quarter. In our solution, partners have the initiative on their side.

Currency

Another complex topic to deal with is currency system inside the application. There are many possible solutions, including automatic currency rate calculator and showing both Dell and partners their primary currency, but this solution is not precise because of currency rate changes. With multiple currencies in the system, Budgets and generally all amounts of money would change with changing rates, either for Dell or for partners. We have decided to work with one binding currency for both parties, which is Euro.

Conclusion

By Andreas

We have gone above and beyond for this project and really invested a lot of time into the application, this has resulted in an extremely thoroughly built application that we are proud to say is our creation. We had new ideas coming to us every day while working and we wish that we could've implemented them all, but with an application like this we would never be finished. All of the core functionalities has been made and implemented, which is the most important, but still we would have liked these additional features to make it into the final version. Given the relatively short span of development we have accomplished an amazing product that we are confident can hold up in the real world, and would love to see it be used.

Install instructions

Setup

1. Save attached zip named "Dellproject.zip"

2. From Netbeans:
    a. Go to file in upper left corner and click ‘Import from zip’
    b. Browse for the zip file where you saved the zip in step 1
    c. ‘Folder’, remember the directory you extract to
    d. Import
3. The following steps are important
4. Right click the project in the ‘Projects’ navigation pane to the left in netbeans
5. Click properties
6. Under Build and Compiling, disable ‘Compile on save’
7. Browse to the folder where the project is extracted, from point 2.c
8. Navigate to [Project Path]\web\WEB-INF\poe-folder and copy the path
9. In the file Controller, which is in the Domain package, go to line 20
10. Paste in your copied path so that the variable POE_FOLDER now is your path
11. Make sure that backslashes are escaped if you are using windows, Netbeans will do this automatically
12. Click run (F6)

Using the application

When the project has ran, it should open in your default browser. If not, open your browser and go to http://localhost:8080.

You should now be at the login screen, on the left side you will see 3 different login credentials, we recommend starting with the first one. When logged in with the given credentials you are encouraged to browse and explore the whole site using the navigation in the top.

We recommend using incognito/private mode in your browser to easily swap between Dell user and Partner.

Things to try

Try the filters

When logged in as a Dell user (Claus), click the 3 different boxes in the top to change list according to the text in the box you're clicking.

Create a partner

Click the 'Partners' link in the top while logged in as a Dell user. Click 'Add partner', put in a name and then select a country, if you put in a real company name you should now see 4 suggestions for a logo that should match you company name. Click on one of the images to use that as a logo, or you can upload your own.

After creating the company you are redirected to its page, you can now add a user for this company, click on 'Add user'. See next headline below for instructions.

Create a user

Go to Users, click 'Add user'. Enter the details and use an email that you have access to, when user is created you should receive an email with a link, if you follow the link you can set a password for this newly created user. Make sure you are logged out before clicking the link as it will not work if you are already logged in. You can log out by hovering you mouse over the name of the user in the top right corner and clicking 'Log out'.

Try the search

While on Dashboard, click on the search field in the top right. Put in whatever you like, after 3 characters you should see some magic.

Move a project from start to finish

As a partner, click the Project Request button on the top, fill out details and click 'Send Request'. Now looking at the Dashboard from a Dell user you will see the newly created project, click on it and approve it. Continue this, alternating user and moving to the next step to simulate a project lifecycle.

Upload and view PoE

When logged in as a partner, go to a project that is marked 'Project Approved', you should see an upload form. Try uploading different kind of files and see your file appear in a nice thumbnail. If it's an image it will have a preview and a slideshow with all uploaded images.

Send a message

Go into any project and in the bottom you can put in a message that you can submit as a comment.

Set a budget

Click budgets in the top while logged in as Dell user. You can either click an existing budget or create a new one. While on budget edit page you can change the value and save.

Check out the stats page

While logged in as Dell, click on Stats page in the header. You can now see a lot of graphs and charts.

Process report

Project Backlog

By Jan

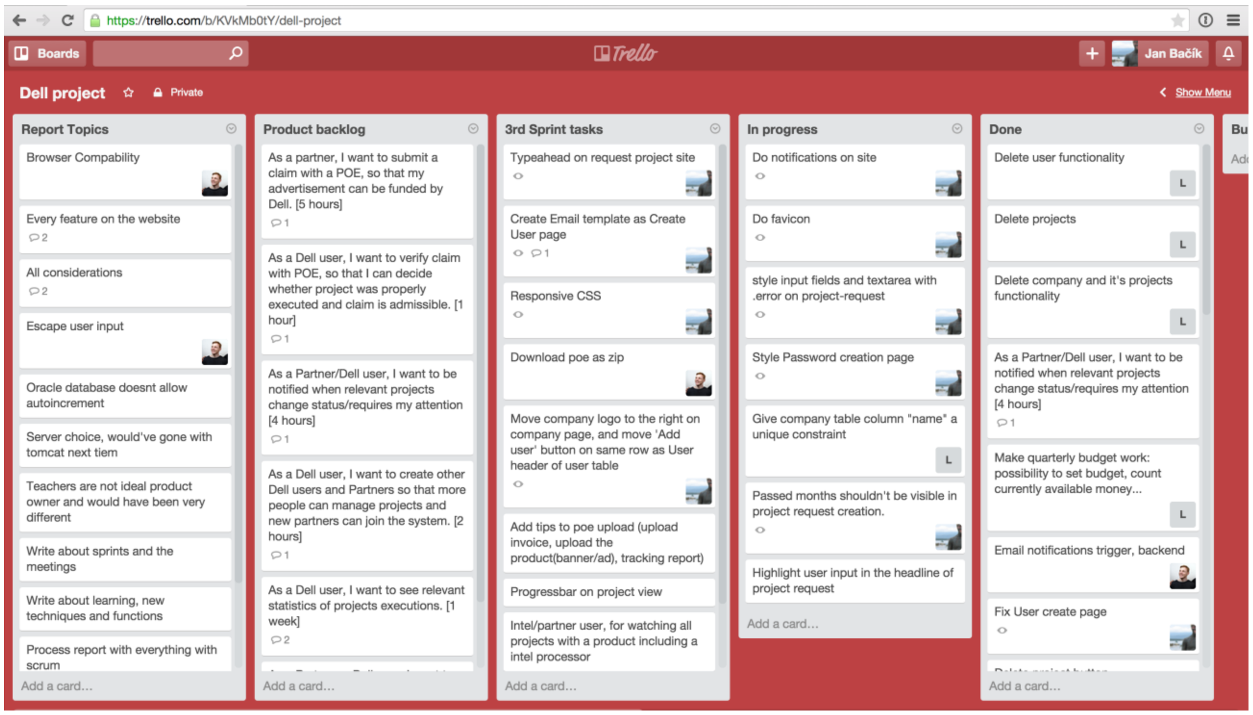
In the beginning of the project, we have created all backlog items (user stories) we saw necessary, and these definitions proved to be solid in the process of development, therefore we did not need to alter any of the user stories in the following sprints.

Nevertheless, the attributes of time and sprint selection were sometimes subjects of change. For example, our initial time estimate for Statistics user story was set to one week, but user story was then finished in the time of just two days. It is also true that the opposite case when we underestimated user story’s difficulty was more usual.

Tasks

In the process of application development, we have decided not to directly connect all the sprint tasks with related user stories, because we worked with high number of “scaffolding” tasks, which are needed to be done to make backlog related tasks. These scaffolding tasks included mostly database-related activities and building of user interface with bigger scope than needed just for fulfilling backlog item.

Therefore, our tasks for each sprint are collectively aimed to fulfill current backlog items, but not all of them can be referenced as subpart of user story.


https://trello.com/b/KVkMb0tY/dell-project

Dell project — Private

Report Topics
- Browser Compability
- Every feature on the website 2
- All considerations 2
- Escape user input
- Oracle database doesnt allow autoincrement
- Server choice, would've gone with tomcat next tiem
- Teachers are not ideal product owner and would have been very different
- Write about sprints and the meetings
- Write about learning, new techniques and functions
- Process report with everything with scrum

Product backlog
- As a partner, I want to submit a claim with a POE, so that my advertisement can be funded by Dell. [5 hours] 1
- As a Dell user, I want to verify claim with POE, so that I can decide whether project was properly executed and claim is admissible. [1 hour] 1
- As a Partner/Dell user, I want to be notified when relevant projects change status/requires my attention [4 hours] 1
- As a Dell user, I want to create other Dell users and Partners so that more people can manage projects and new partners can join the system. [2 hours] 1
- As a Dell user, I want to see relevant statistics of projects executions. [1 week] 2

3rd Sprint tasks
- Typeahead on request project site
- Create Email template as Create User page 1
- Responsive CSS
- Download poe as zip
- Move company logo to the right on company page, and move 'Add user' button on same row as User header of user table
- Add tips to poe upload (upload invoice, upload the product(banner/ad), tracking report)
- Progressbar on project view
- Intel/partner user, for watching all projects with a product including a intel processor

In progress
- Do notifications on site
- Do favicon
- style input fields and textarea with .error on project-request
- Style Password creation page
- Give company table column "name" a unique constraint
- Passed months shouldn't be visible in project request creation.
- Highlight user input in the headline of project request

Done
- Delete user functionality
- Delete projects
- Delete company and it's projects functionality
- As a Partner/Dell user, I want to be notified when relevant projects change status/requires my attention [4 hours] 1
- Make quarterly budget work: possibility to set budget, count currently available money...
- Email notifications trigger, backend
- Fix User create page

Add a card...

Figure 33 - Tasks in scrum board during 3. sprint period

Planning sprints

As seen bellow, our group worked on described user stories in order of their priority. Anyhow, there were some exceptions, like the user story mentioning "dashboard" view of categorized projects. Some tasks and user stories proved to be more effectively done when it makes programmatically sense, rather then when it's their turn based solely on priority.

User story	Time	Priority	Sprint
As a partner, I want to create project request, so that project can be verified and I can execute a project.	12 h	100	1
As a Dell user, I want to verify project, so that partner can proceed to the next step.	5 h	90	1
As a partner, I want to submit a claim with a POE, so that Dell can fund my advertisement.	8 h	80	2
As a Dell user, I want to verify claim with POE, so that I can decide whether project was properly executed and claim is admissible.	2h	75	2
As a partner, I want to be able to contact Dell during and before project execution.	6h	60	2

As a Dell user, I want to end project to let partner know that we will reimburse and project is finished.	2h	55	3
As a Partner/Dell user, I want to be notified when relevant projects change status/requires my attention	6h	50	3
As a Dell user, I want to have overview of all active projects and it's states.	3h	40	1
As a Dell user, I want to create other Dell users and Partners so that more people can manage projects and new partners can join the system.	5h	35	3
As a Dell user, I want to see relevant statistics of projects executions.	10h	30	3
As a Partner or Dell user, I want to have access to the system and all it's functions on mobile.	10h	25	3

Retrospective

By Andreas

1st sprint

7.04 - 16.04

Since this was the group's first time working together on a subject that was relatively new to us we undersold ourselves and didn't aim high enough for the amount of features we could implement in the first spring. Therefore, we managed to come quite far in the first week and had the basic and core functionality working within the end of the week. When we had finished our spring backlog we started began integrating agile development by pulling extra features from the product backlog.

Product Owner Meeting - with TUE

At this meeting we had come surprisingly far with our product and had a very functioning demo to show. We went through the process of creating a project and it's flow from start to finish which TUE seemed very pleased with. He didn't have a lot of comments on our program but did clear up a couple of questions we had collected through the week as he acted as the product owner.

Technical Review - with HAU

We were not very prepared at this meeting as we had not seen the requirements of what to bring to the meeting and it showed. We expected it to be a more technical review of our solution but it turned out to be very diagram/model heavy which we lacked in. We all had a common vision of the goal and how the program should work which allowed us to work without the help of diagrams. Henrik was not pleased with what we had and we still promised ourselves to be more prepared to the next meeting.

Group retrospective meeting

Not hitting on target with velocity is quite normal with new groups and projects and we didn't think anything about it, we now could aim more precise for the following sprints. We are quite satisfied with the progress we made this week and our group cooperation.

2nd sprint

17.04 – 23.04

Technical Review (and short PO meeting) - with HAU

Today the product owner meeting was unfortunately cancelled because of sickness but Henrik took over and acted as product owner for 5 minutes at the start of the meeting. For first part of the meeting we showed the user stories we had in our sprint backlog for this week and we followed the "How-to-demo" instructions we had set at the start of the week. At this stage the core of our program is mostly done and Henrik was very satisfied with our progress and approved of the finished user stories. The rest of the meeting was a technical review to which we this time were more prepared for. We had the right diagrams, which were all done well, we could show Henrik very precisely the structure of our program and had all the answers to his questions on the diagrams.

Overall, Henrik was surprised and content with what we had done, which was a big improvement from the previous meeting.

Group retrospective meeting

At this sprint we had a more precise view of our sprint points and velocity, this allowed us to come closer within the expectations. We were perhaps a bit too confident resulting in us slightly overshooting this time and we did not manage to complete the whole sprint backlog.

3rd sprint

27.04 – 1.5

Group retrospective meeting

We got a late start this sprint and on the first day back we realized we were quite busy if we wanted to do all the remaining features. But we were consistent and motivated to complete the application wholly so we had long days making sure we did our days' worth of work. The last few days the stress of not finishing cleared and we were sure we would complete the features we wanted from the beginning of the sprint.

Post-sprint

Even after the last sprint we were devoted to adding more valuable features and we ended up spending some of our spare time as well as some supposed report writing allocated time to develop the application further. Due to this devotion to the application we had a late start on report writing, and we got quite busy and demotivated for the first two days. Fortunately, we calmed down and worked hard to write a good chunk each day.

Group co-operation

By Jan & Andreas

Our group was teaming up from time-proven collaboration of Jan and Andreas, and for some time before project started, our intention was to try to make the team of two, even rejecting a couple of requests for joining the group. However, before the project began we agreed with Lasse to join the group, which shown as great addition, making our workflow easier and enabling our group to do much more in given time.

We weren't given ourselves specific roles in the team, and didn't have any hierarchy, as our workflow was based on collective trust and individual responsibility. It is only possible in small teams, and this approach worked for us without problems. It did not take long to learn about the skill level of each member and quickly gained trust in each other that they could perform individually.

Instead of roles, we have naturally chosen our responsibilities in development; example would be Jan's responsibility for presentation layer, Lasse's for data source layer and mostly Andreas' responsibility for domain layer.

For more detailed overview of team member's programming activities, there is detailed table of files and members who	**Contributor**	**Contributor**	**File**

contributed most to them. Files are color-coded based on their architecture layer. Purple for Data layer, blue for Domain layer, green for Presentation layer.**Contributor**			
Lasse			BudgetMapper
Lasse		Jan	CompanyMapper
Lasse	Andreas		DatabaseConnection
Lasse	Andreas	Jan	DatabaseFacade
	Andreas		MessageMapper
	Andreas		NonceMapper
Lasse			PoeMapper
Lasse			PoeMapperTest
Lasse	Andreas	Jan	ProjectMapper
	Andreas		StatisticsGetter
Lasse	Andreas	Jan	UserMapper
Lasse			Budget
	Andreas		Company
Lasse	Andreas	Jan	**Controller**
	Andreas		DisplayProject
Lasse			FileHandling
	Andreas		IdGenerator
	Andreas		JSONTranslator
	Andreas		Login
	Andreas		Message
	Andreas		Nonce
Lasse	Andreas	Jan	Notifications
Lasse			Poe
	Andreas		Project

	Andreas		Result
	Andreas		ResultContainer
	Andreas		Stage
Lasse			User
Lasse	Andreas	Jan	**PresentationServlet**
		Jan	*.js
		Jan	*.css
		Jan	Budgets
		Jan	Create-budget
		Jan	Create-company
		Jan	Create-user
	Andreas	Jan	Create-project
		Jan	Edit-budget
	Andreas	Jan	Header
	Andreas	Jan	Index
		Jan	Login
		Jan	Partner
		Jan	Partners
	Andreas	Jan	Project
		Jan	Reset
	Andreas	Jan	Statistics
		Jan	User
		Jan	Users

Report authors

Table of individual sections' authors of this report is below to see.

Author	Author	Author	Section
	Andreas		Business case
		Jan	Vision, Traceability

	Andreas		Activity diagram, Domain model
Lasse			Architecture and design patterns
	Andreas		Design class diagram
Lasse			Sequence diagram
		Jan	E/R diagram
	Andreas		Relational schema
	Andreas	Jan	User Interface
Lasse	Andreas		Complex solutions
		Jan	SQL Queries
		Jan	Features
Lasse			Tests
Lasse			Known errors
	Andreas		Future implementations
		Jan	Considerations
	Andreas		Product conclusion
	Andreas		Install instructions
		Jan	Project backlog
	Andreas		Retrospective
		Jan	Group co-operation
		Jan	Group techniques
	Andreas		Conclusion

Group techniques

By Jan

Our group followed the main principles behind Scrum and Agile development methods, which helped us with organizational decisions during application development. We take our three-person structure as an advantage in distribution of work, and administrative tasks. It is easier to go through meetings with lower group of people that needs to understand and agree on every major decision.

For the most of the sprint days we have decided to meet every day and work on the project together instead from home, to make team interactions and discussion about project's features and solutions used faster, therefore more productive.

Tools

By Jan & Andreas

GIT and Github was our tool of choice for versioning, and despite few cases of commit conflicts where Github needed manual help, we are satisfied with this solution. Anyhow, we agreed to use different solution BitBucket in future projects, which offers the same GIT functionality, but enables users to create free private repositories, which is premium feature in Github.

Task management system Trello helped us to connect the methods of Scrum with online collaborative environment. This represented big benefit in task management, as we were able to glance and change every aspect of every task or user story from anywhere.

Google Drive was used in our group to share mostly diagrams and Scrum documents. For diagrams creation, we have used Gliffy web application, for easy collaboration and seamless ability to change and rework our diagrams.

A tool not made of one and zeroes is our selves, whenever we have had a problem individually, we could consult our group members for help, instantly stopping what they were doing and devoting their undivided attention to the new problem that now affected everyone. This has been the most useful and most used tool, getting a fresh perspective and a couple of more brains to solve the problems. Passionate debates have risen from these problems that all have resulted in an agreement.

Scrum

Methods of Scrum were used extensively in the development. Daily Scrum meetings, sprint planning, sprint review and sprint retrospective meetings were held regularly to test and learn development with use of this technique. Daily "stand-up" meetings were particularly useful. It gets everyone on the same page while also provides easy way to orient about different parts of the process.

Agile development

We quickly adopted the way of agile development, not on purpose but naturally. The group focused on working code and individual productivity. Nevertheless, because of low number of team members, we did not use the agile method of pair programming. Another reason was that we felt confident in our comrades that they would accomplish solid work that we all could agree met our expectations.

Conclusion

By Andreas

This project has been very exciting for us, keeping us at school until late most days, testing our abilities and inspiring us to do and learn more. The scrum development process has been a new domain for us which we have embraced most of it, it has taught us a new tool that is useful in projects of this magnitude. It helped us working collectively towards a common goal with all of us being at the front simultaneously. The real-life simulation of a real business case have been refreshing, it makes you more motivated to do your best.

Using every bit of taught methods and qualities from class in the second semester have proven very useful. Adapting the correct design pattern since day one, knowing the scale of the application in front of us, have saved us tons of time. Further learning about web development on a personal level have given great competences, useful in a lot of different scenarios.

Fortunately, we have not have any disputes within the groups, letting us work seamlessly throughout the whole project. We believe we have been very lucky with the group that is built around trust in each other. From this, we cannot learn much about group cooperation when all of us were on the same page since the very first day. We have been very self-dependent and have not once gone to a teacher for advice, instead resorting to the World Wide Web for help, which has proven success time again. Our internal discussions have given us so much and taught us not to be afraid of relying on others.

Authors

Report

Table of individual sections' authors of this report is below to see.

Section	Author	Author	Author
Product Report			
Business case		Andreas	
Vision, Traceability			Jan
Activity diagram, Domain model		Andreas	
Architecture and design patterns	Lasse		
Design class diagram		Andreas	
Sequence diagram	Lasse		
E/R diagram			Jan
Relational schema		Andreas	
User Interface		Andreas	Jan
Complex solutions	Lasse	Andreas	
SQL Queries			Jan
Features		Andreas	Jan
Tests	Lasse		
Known errors	Lasse		
Future implementations		Andreas	
Considerations			Jan
Product Conclusion		Andreas	
Process Report			
Project Backlog			Jan
Retrospective		Andreas	
Group co-operation		Andreas	Jan
Group techniques		Andreas	Jan
Process conclusion		Andreas	

Code

🟪 Data layer

🟦 Domain layer

🟩 Presentation layer

File	Contributor	Contributor	Contributor
BudgetMapper	Lasse		
CompanyMapper	Lasse		Jan
DatabaseConnection	Lasse	Andreas	
DatabaseFacade	Lasse	Andreas	Jan
MessageMapper		Andreas	
NonceMapper		Andreas	
PoeMapper	Lasse		
PoeMapperTest	Lasse		
ProjectMapper	Lasse	Andreas	Jan
StatisticsGetter		Andreas	
UserMapper	Lasse	Andreas	Jan
Budget	Lasse		
Company		Andreas	
Controller	Lasse	Andreas	Jan
DisplayProject		Andreas	
FileHandling	Lasse		
IdGenerator		Andreas	
JSONTranslator		Andreas	
Login		Andreas	
Message		Andreas	
Nonce		Andreas	
Notifications	Lasse	Andreas	Jan

Poe	Lasse		
Project		Andreas	
Result		Andreas	
ResultContainer		Andreas	
Stage		Andreas	
User	Lasse		
PresentationServlet	Lasse	Andreas	Jan
*.js			Jan
*.css			Jan
Budgets			Jan
Create-budget			Jan
Create-company			Jan
Create-user			Jan
Create-project		Andreas	Jan
Edit-budget			Jan
Header		Andreas	Jan
Index/Dashboard		Andreas	Jan
Login			Jan
Partner			Jan
Partners			Jan
Project		Andreas	Jan
Reset			Jan
Statistics		Andreas	Jan
User			Jan
Users			Jan