CHALMERS

FT WEB 3.0

Designing and implementing web application for field test

Master of Science Thesis (Secure and Dependable Computer Systems)

ALEXANDER PERSSON

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, October 2012

Designing and implementing web application for field test at Volvo 3P.

Alexander Persson

Examiner: Sven-Arne Andreasson

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden October 2012

Abstract

The field test group is validating and verifying trucks. The field testers are working in groups and with projects often globally. All field tests result in a lot of data stored in file binders. This makes the globally work very difficult.
The purpose was to digitalize the old way to work and create a solution to store this work in some sort of database. This requires creating an application connected to a database. The purpose was also to create a more efficient way to work.
The thesis work began with an investigation about which platform, database and tools that should be the best for this purpose. The requirements were identified, both functional and non-functional. The next step was to investigate all the options available and take decisions for what platform, database and tool to use. The applications was designed and implemented by agile software development.
The result showed to be satisfied, the computer were used by 60-120 persons the day the master thesis work ended.

Keywords
PHP, Oracle, Web application

Acknowledge

This master thesis was made at Volvo 3P by Alexander Persson written for Chalmers University of Technology. First I would like to thank my two supervisors Sven-Arne Andreasson at Chalmers and Kimmo at Volvo 3P this could not been performed without you. I also want to thank the field test group at Volvo 3P, specially Thomas and Richard to coming to the meetings and discuss what to do in my thesis work.

List of figures

List of tables

Contents

1. Introduction

1.1 Background

Volvo 3P forms a part of Volvo AB as a business unit. The three P stands for product planning, product development and purchasing.

The verification and validation group of 3P is working with the brands Volvo, Renault trucks, UD trucks and Mack trucks. They are working in projects that can be a field test; the field test can consist of validation or verification of a specific truck or a unit in a truck. A unit can be everything from a new mug holder to a new engine or a gearbox. A field test generates a lot of data; this data can come from different persons working in the field test. These persons can work at the same place in the world or in different places in the world with different time zones. Examples of those places are Australia, Sweden, France and India etc. A field test may have long test periods; 2-3 years are common.

Today lots of data are stored in file binders. The verification and validation team are working with vehicles, haulers, drivers, dealers. A field test consists of data from vehicles, haulers, drivers, dealers which are registered. The data is sometimes recurring in new field tests and the data must therefore be stored to easy receive the data next time a new field test is created.

A field test generates a lot of new data; the data often involves the four base units (vehicle, drivers, haulers and dealers). During a field test a lot of actions are taking place. The actions must be logged; today this is made in text document stored at the local desktop by the person that performed the action.

Examples of actions that need to be logged or data about a field tests that need to be stored are:

- Mileage for vehicles within the field test
- Standstill for vehicles within the field test
- Diary post for happenings or other important information within a test
- Information related data about the specific field test such as persons involved, vehicles, reports, accounts registered to the field test and involved brands etc.
- Service planner data
- Consumption data

FT Web

Versions of FT Web have been released during a period of 6 years. The first version FT Web system was made by Volvo IT. The first version did almost only provide a consumption following up system for all vehicles in a field test. The system did not gather any other information.

The second version of FT Web provides more functions for data storing for vehicles, dealers, haulers and drivers. The system did also provide some functions for storing information about field tests.

1.2 Problem

Data from field test projects is often stored and gathered for 2-3 years; therefore it is important to gather all data at the same place for different field tests. It is also very important to store data from field tests in the same way because some tests are overlapping. The tests are also spread geographical all over the world with different time zones etc. A system for validation and verification of trucks creates a prerequisite for standardization of test methods, which will lead to efficiency opportunities and opportunities to the development of future FP core business. It is important to have the opportunity to go back and look at older tests.

A system is needed to store and process data from the field tests. The third version of FT Web will provide:

- Store data for vehicles, haulers, dealers and drivers
- Store and process project data
- Mileage follow-up
- Consumption follow-up
- Generate project status reports(graphs and informative views)
- Distribute global messages
- Distribute data in an efficient way
- Standstill follow-up
- Store diary for vehicles and projects
- Service planner
- Protus connection to count all points for vehicles and objects

All code must be recoded and MVC will be used as a design pattern to make the development easier in the future.

1.3 Method

This master thesis work is mainly focused on designing and implementing a new computer system.

Literary study

The literary study started to investigate the proper tools to design a new system for the field test group. It is important to investigate the advantages and disadvantages with the different types of language and the different types of database used for the design of the new computer system.

The literary study is focused on how to implement MVC (Model View Controller) in PHP. This is a little bit tricky because PHP is a script programming language. There are many guides of how to implement such patterns, by first an investigation is made then a MVC pattern structure can be implemented.

User interface design

It is important to design a system that could be recognized as a Volvo system, therefore when the system is designed a research about how the current Volvo systems are designed is made. The research will be about some internal system designs and the home page for Volvo, Volvo trucks and Volvo 3P.

Implementation

When the design of the user interface is finished implementation of the functions will be started. Regularly meetings with the stakeholders will be held before a new function is implemented, in this meeting an evaluation will be held and then the requirements for the next function to implement will be discussed. Agile software development style will be used.

Testing

The testing will be performed from the beginning to the end. Because the use of agile software development it will always ensure that the stakeholders' requirements will be met.

2. Web tools

2.1 Website languages and databases

Website languages and databases are the keys to build websites.

2.1.1 Web development languages

Web development languages are languages made to developing the graphical user interface for a websites. Programming languages often includes functions for connecting the graphical interface to a database.

PHP

PHP is an open-source programming language mainly used for creating dynamic web pages. It was released in the 1990s and is today widely used by more than 14 million websites [1] PHP can be used as server-side scripting language, as command line scripting and to writing desktop applications. PHP has a library for functions and classes that will make the programming more easy, secure and efficient. There are many MVC frameworks written in PHP, examples of frameworks are Zend, Symfony and CakePHP. PHP can be installed on most operating systems as Windows, Linux, UNIX, OS X and more.

Server-side scripting is the most common way to use PHP. The PHP code is often embedded into HTML code and will be interpreted and executed by the web server then send the dynamic page to the web browser that asked for the web page.
PHP code can be executed in a terminal as a command line. The interpreter will interpret and execute locally at the computer and then write the result to the terminal.
There is a way to write desktop applications in PHP, an extension called PHP-GTK is created to make this more efficient. [2]

ASP

ASP stands for Active Server Pages, and is a server side scripting language. It is made by Microsoft and can be downloaded for all Microsoft operating systems. The scripts is running when a web browser has requested a web page, the execution of the scripts are done before sending the page to the web browser client. The ASP code is embedded into HTML code.

.NET framework

.NET is the counterpart of Suns' Java for Microsoft's operating system. It is a system component in the Windows operating system. .NET provides a framework with a huge class library of solutions to security, database connection, network connections and much more. .NET is standardized to CLI (Common Language Infrastructure). [3] The class library of .NET makes it very easy and quick to create applications. Microsoft provides two environments to create applications in .NET Visual Studio and Visual Web Developer. [4]

ASP .NET

ASP .NET makes it possible to create web sites and web applications. Web applications are more advance web sites. ASP .NET is an event programming language this makes it possible to retrieve new information from the web server without reloading the whole page. [4] ASP .NET also provides a solution for MVC ready for use for the programmer.

ASP .NET VS PHP

The comparison between ASP .NET and PHP can be seen as a little bit unfair because ASP .NET is a framework and the applications in ASP .NET can be compiled and PHP is a pure server-side scripting language and cannot compile code. It is also hard to compare the two languages because there are so many factors to be considered to make a full comparison. The performance between ASP .NET and PHP is different for different scenarios. [5]

There are big websites written in both PHP and ASP .NET as Word Press and Wikipedia are written in PHP and Live.com and MSN.com are written in ASP .NET. Google is written in both, some parts are written in PHP and some ports in ASP .NET, this shows that both languages has their own effective fields in different scenarios.

Scalability
There is more a question about the programmers experience when comparing the scalability and maintaining of the software. A more experienced programmer will create better code which will scale better and be easier to maintain.

Speed
The speed comparison is the field where the factors and scenarios must be considered the most. For example when a website fetch data from a database it is not only the connection layer between the website and the database that must be considered, it is also a question about which database will give the fastest result by a SQL-query. Therefore in a scenario when a website fetching and storing a lot of data a comparison between MSSQL and MySQL and Oracle and MSSQL should be made, I will describe such comparison later. The speed in database also depends on the SQL queries, experienced programmers will write better and faster SQL queries.

Another factor to be considered is when the website is accessing files at the file system. PHP is often installed at Linux server with the file system ext4 and ASP .NET is often installed at Windows server with NTFS file system. Comparison between ext4 and NTFS shows that ext4 performs better when accessing files. [5]

ASP .NET using C#, due the fact that C# can be compiled then ASP .NET is faster than PHP because PHP is never compiled. The major different can be seen if a website is doing for example 2.000.000 loops, the compiled code in C# will be faster than a script solution in PHP. But another aspect, how many websites needs a loop looping 2.000.000 times?

Cost
Open source programs are free to use and upgrade, ASP .NET needs license from Microsoft. There are free solutions to use PHP both for Windows platform and Linux platform. WAMP is a program for Windows platform containing Apache, MySQL and PHP, this program can be installed and then allows the computer to be a web server to host websites. LAMP is the name when Apache, MySQL and PHP are installed at a Linux platform. Apache MySQL and PHP are open source programs and all use and upgrades are free.

ASP .NET runs on Windows server, MSSQL needs to be installed for database connection. A license of Microsoft Server Bus server cost around 750 Euros. [6]

There are solutions for running ASP .NET under Linux [7] but the performance will not be the same as at the Windows platform and some features are missing.

Time to implement
This is also a question about the programmer’s experience. The programmer should choose the language that he or she is most experienced in. The implementation time also depends on which language that requires most written lines when making advanced websites. ASP .NET requires more written lines than PHP when advanced websites are built. [5]

Security
The security question very much depends on the experience of the programmer. There exist websites programmed either in ASP .NET or in PHP that suffer from vulnerabilities. It is more a question about create safe code that protect against security threats like SQL injections and cross-site scripting.

Conclusion about what language to use for an implementation
The choice between PHP and ASP .NET depends on the programmers’ experience. PHP should be used if the programmers have more experience in PHP than ASP .NET and ASP .NET should be chosen if the programmers have more experience in ASP .NET. The choice also depends on the platforms that are available. PHP is platform independent and will work on Linux, UNIX, Windows, Mac OS and Solaris ASP .NET can only be installed at Windows platform. [8]

2.1.2 Databases
A database is a place where data can be stored organized.

Oracle
The development of the first version of Oracle database started in 1977. The data is stored in data files at the computer running the Oracle database software. The database has support for SQL and row lock. Oracle also allows data procedures from for example Java. In order to make the execution faster Oracle use parallel execution. Oracle can be installed on most platforms Linux, UNIX, OS X, Windows and more.

MySQL
MySQL is an open source database management system and is very common on Linux platform but can be installed at most platforms. MySQL was bought 2008 by Oracle. MySQL support SQL and is a very fast software using multithreaded functions. It allows multi-user and has support for command line execution.

Microsoft SQL Server
MSSQL exist in many versions that can be customized from small business to very large business with a lot of data traffic to and from the database. MSSQL can only be installed at a Windows operation system. In MSSQL readers do not block writers due new lock mechanism. MSSQL use Btree for indexing and provides backup options. [9]

2.2 Software vulnerabilities
Software vulnerability is a safety risk in software, the software suffer from a weakness that can be exploit.

2.2.1 SQL-injections
Structure Query Language is a language used for database connections, used for example Oracle or MySQL. Example code for a SQL query is:

“SELECT * FROM foo WHERE id=’$id’ AND pwd=’$pwd’”

SQL-injection is a way to exploit security flaws in the programming design. For example this query could be a query used for login. The problem arise if the variable $id and $pwd is a direct input from the intruder, as for the example above two inputs from a web form. The input can change the SQL query structure. For example if $id could be admin and $pwd be ‘ OR ‘’ = ‘ this will lead to the SQL query:

“SELECT * FROM user WHERE id=’admin’ AND pwd=’’ OR ‘’=’’”

OR ‘’=’’ will ensure that the SQL-query always result in a given row.

The intruder will be logged in as admin with admin user rights.

There two ways of protecting against SQL-injections parameterized statements and escaping. In FT Web I have choose to use escaping.

3. Requirements

3.1 General requirements

3.1.1 Security

To make sure that the system is secure the system must be coded to be protected against PHP injections, cross-site scripting and other threats.

3.1.2 User classes

The system provides much data and many functions that change and manipulate the data. It is important that only those users that are allowed to for example add, modify or delete data are able to do that. Therefore there are four user classes all with different privileges:

user	read	write	Admin-data	Developer-data
read	X			
Read-write	X	X		
Admin	X	X	X	
Developer	X	X	X	X

Read-only
These users can only read the data; all data except the admin and developer related data are visible for the read-only users.

Read-Write
The read-write users can read, add, modify and delete all data except the admin and developer related data.

Admin
The admin users can read, add, modify and delete all data except the developer related data.

Developer
The developer users has full access to the system, developer users can add, modify and delete all data.

3.1.3 Dependable

Due lack of support and teaching, the system must be very dependable. There will be very costly if changes must be made when the master thesis work has ended. The system must always provide user friendly error messages. The system must be checked for errors and bugs in a bigger scale.

3.1.4 Volvo accounts

All employees of Volvo have one unique text string called t0-number. Volvo also keeps information about the employees by this t0-number such as name, organization, departure, address etc. All employees also have a password used for the login of Volvos all applications and the computers at Volvo.
There is a requirement that the FT Web system must use this t0-number. An integration module was created for fetching t0-number information. Integration was also created to use a login script written in PHP used for Volvo applications.

Because many users are interacting together in the system while working with for example a field test there is a use of a function that returns information for a user of FT Web. The FT Web system keep track of which user that update or create something this is shown at those pages where people can modify or create new records of something like the profile pages. Therefore I made a function that let users click on a name and then a popup is shown with contact details for the user. To make the FT Web more personally one requirement was to make it able to upload a picture for all users of FT Web. This picture will also be shown in the contact details pop-up

3.2 Functional requirements

- System shall provide add/modify/delete vehicles
- System shall provide add/modify/delete dealers
- System shall provide add/modify/delete haulers
- System shall provide add/modify/delete drivers
- System shall provide add/modify/delete users
- System shall provide add/modify/delete field tests
- System shall provide add/modify/delete entries for mileage data
- System shall provide add/modify/delete entries for standstill data
- System shall provide add/modify/delete entries for diary data
- System shall provide add/modify/delete entries for consumption data
- System shall provide add/modify/delete entries for mileage data
- System shall provide functions for generating status reports for the field tests
- System shall provide functions for distribute news globally
- System shall present data in informative views
- Use the Volvo t0-number account for the user system in FT Web

3.3 Non-functional requirements

- System must provide user classes with different privileges to hide or protect sensitive data
- System must be available on all Volvo sites
- System must be backed up on regular basis

- The system must provide information about the workflow, in other words the system must be able to give good explanations to users that not have backgrounds for computer technical systems.

3.4 Future requirements

- Flexibility, the work flow may change

4. Analysis

4.1 Platform

There was lack of options when choosing tools and platforms because of Volvo 3Ps limitations of different platforms and tools.

4.1.1 PHP

The choice to use PHP is simple; I have much more experience in PHP than in ASP .NET. I contacted Volvo IT and I get access to a Linux server that had apache and PHP installed.

4.1.2 Oracle

I did not have the opportunity to choose between different databases. The only database that I could get an access to was a connection to an Oracle database. Despite the lack of other database options I think that an Oracle database would be the best chose for database. Foreign keys helped me a lot in the work of always get correct data stored in the database tables.

4.1.3 Workstation

Almost all workstation computers at Volvo run some version of Windows operating system. This is not something I can have an influence of. As a master thesis worker I got a computer prepared with Windows XP and I know that I can code PHP and HTML in this environment.

4.1.4 Linux server

The oracle database and the apache web server were located at a Linux server, I could not change this but I still think this is a great solution.

4.2 Tools

4.2.1 Oracle SQL Developer

Oracle SQL developer is Oracles own program for maintaining Oracles database. I have choosing this program because I think it is a very efficient way to work. Oracle SQL editor gives a very good overview of all tables and provides fast information about for example keys and constraints. It is very easy to create, update and delete tables. The program can also execute pure SQL queries that will affect the database.

4.2.2 DBDesigner

DBDesigner is a program for creating ER-diagrams for Oracle databases. Foreign keys and constrains can be recognized. The program can also generate SQL queries for creating all tables from the diagram, I did not test this function but it seems like an efficient function.

4.3 Domain analysis

4.3.1 ER-diagram

The ER-diagram shows the database scheme.

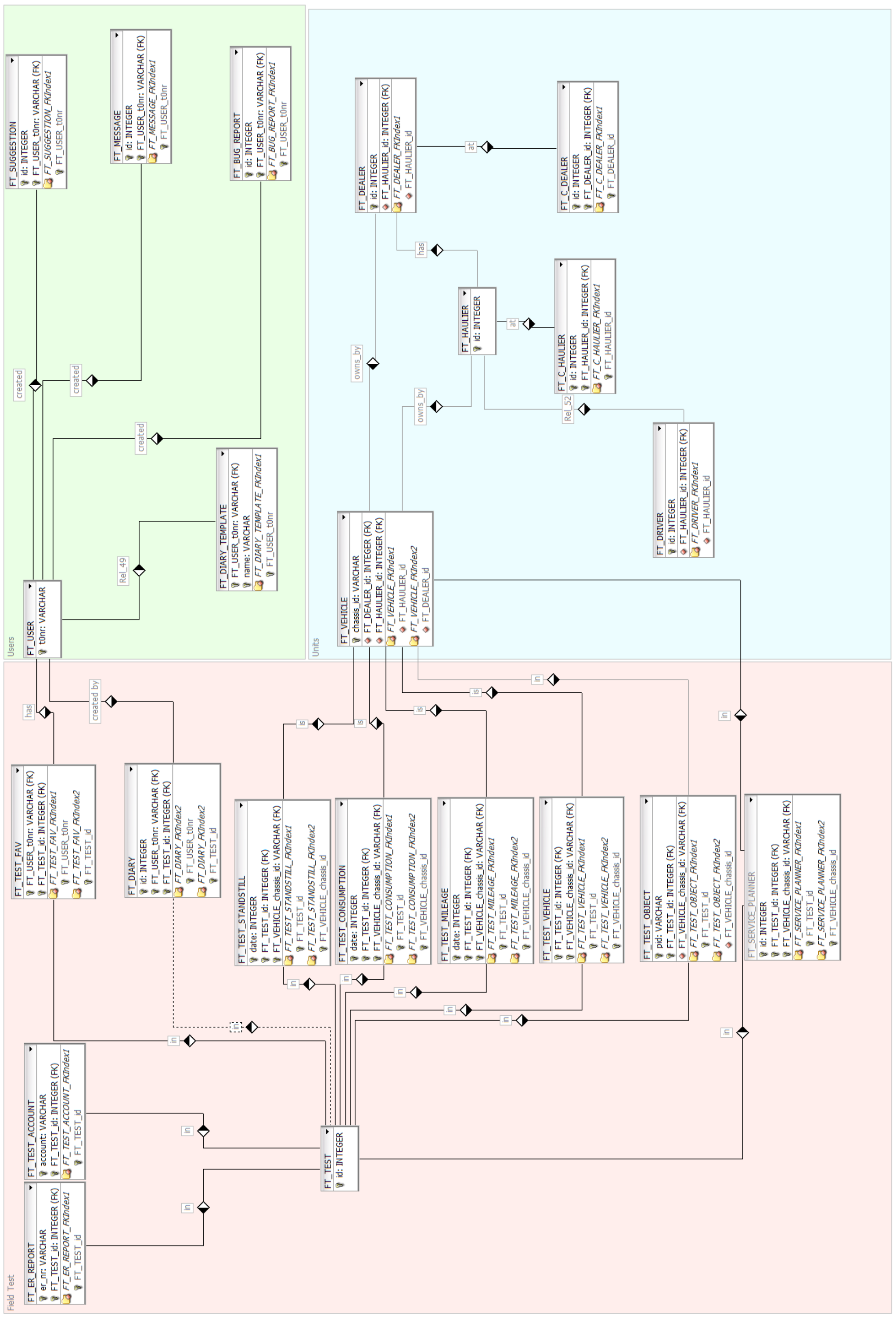

Figure1. Database scheme

4.3.2 Relations

Tables are marked by upper case letters in following descriptions and attributes are marked by lower case letters. Example FT_TEST_id, id is an attribute in table FT_TEST.

FT_ER_REPORT – FT_TEST

FT_ER_REPORT keys: er_nr and FT_TEST_id, the table saves data of all ER-reports

FT_TEST keys: id, the table saves general data of all field tests

An ER-report will be created within a field test; an ER-report cannot exist without a test. The FT_ER_REPORT table is a child table and the FT_TEST table is a parent table. The reference FT_TEST_id is a foreign key. The relation is a many to one relation, in other words one test can be have many ER-reports but an ER-report can only be attached to one field test.

FT_TEST_ACCOUNT – FT_TEST

FT_TEST_ACCOUNT keys: account and FT_TEST_id, the table saves data of all accounts

FT_TEST keys: id, the table saves general data of all field tests

An account will be attached to a field test; an account can exist without a test but cannot exist in the FT Web system without to be attached to a test. The FT_ACCOUNT table is a child table and the FT_TEST table is a parent table. The reference FT_TEST_id is a foreign key. The relation is a many to one relation, in other words one test can be have many accounts but an account can only be attached to one field test.

FT_TEST – FT_TEST_STANDSTILL – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_TEST_STANDSILL keys: date, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves general data of standstills for vehicles

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

A standstill happened by a vehicle in a field test; a standstill cannot exist without a vehicle and a field test. FT_STANDSTILL is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_STANDSTILL table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many vehicles with many standstills, and a vehicle can exist in many field tests and can have many standstills. But a specific standstill can only belong to one field test and one vehicle.

FT_TEST – FT_TEST_CONSUMPTION – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_TEST_CONSUMPTION keys: date, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves data of consumption inputs for vehicles

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

A consumption input is logged by the driver and will be transferred into FT Web to make a graph table to study the consumption for different vehicles under periods of time; a consumption input cannot exist without a vehicle or a field test. FT_TEST_CONSUMPTION is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_TEST_CONSUMPTION table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many vehicles with many consumption inputs, and a vehicle can exist in many field tests and can have many consumption inputs. But a specific consumption input can only belong to one field test and one vehicle.

FT_TEST – FT_SERVICE_PLANNER – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_SERVICE_PLANNER keys: id, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves data of consumption inputs for vehicles

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

A service planner input is created to plan a service for a vehicle. The service can contain for example oil change. A service planner input cannot exist without a vehicle or a field test. FT_SERVICE_PLANNER is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_ SERVICE_PLANNER table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many vehicles with many service planner inputs, and a vehicle can exist in many field tests and can have many service planner inputs. But a specific service planner input can only belong to one field test and one vehicle.

FT_TEST – FT_TEST_MILEAGE – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_TEST_MILEAGE keys: date, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves mileage inputs for vehicles

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

A mileage input is logged by the driver and will be transferred into FT Web to make a graph table to study the mileage for different vehicles under periods of time; a mileage input cannot exist without a vehicle or a field test. FT_TEST_MILEAGE is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_TEST_MILEAGE table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many vehicles with many mileage inputs, and a vehicle can exist in many field tests and can have many mileage inputs. But a specific mileage input can only belong to one field test and one vehicle.

FT_TEST – FT_TEST_VEHICLE – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_TEST_VEHICLE keys: date, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves data for vehicles in field tests and will tell the system which vehicles that belongs to which field tests

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

Vehicles can be registered to field tests; a test vehicle cannot exist without a vehicle or a field test. FT_TEST_VEHCILE is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_TEST_VEHICLE table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many vehicles and a vehicle can belong to many field tests. But a test vehicle can only belong to one field test and have reference to one vehicle.

FT_TEST – FT_TEST_OBJECT – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_TEST_OBJECT keys: date, FT_TEST_id and FT_VEHICLE_chassis_id, the table saves data for objects in field tests

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

Objects can be registered to field tests; a test object cannot exist without a vehicle or a field test. FT_TEST_OBJECT is a child table and FT_TEST and FT_VEHICLE are parent tables. The FT_TEST_OBJECT table has two foreign keys both to the parent tables. The relations are many to one relation, in other words a field test can have many objects and a vehicle can belong to many objects. But a test specific object can only belong to one field test and have reference to one vehicle.

FT_VEHICLE – FT_DEALER

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

FT_DEALER keys: id, the table saves data of dealers

Dealers owns vehicles, a vehicle can exist in FT Web without to be registered to a dealer. The relation is a many to one relation, in other words a vehicle can only be owned by only one dealer but a dealer can own many vehicles.

FT_VEHICLE – FT_HAULIER

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

FT_HAULIER keys: id, the table saves data of hauler

Haulers owns vehicles, a vehicle can exist in FT Web without to be registered to a hauler. The relation is a many to one relation, in other words a vehicle can only be owned by only one hauler but a hauler can own many vehicles.

FT_HAULIER – FT_DEALER

FT_DEALER keys: id, the table saves data of dealers

FT_HAULIER keys: id, the table saves data of hauler

Dealers are registered to haulers; a dealer can exist in FT Web without to be registered to a hauler. The relation is a many to one relation, in other words a dealer can only be register to one hauler but a hauler can be registered by many dealers.

FT_C_DEALER – FT_DEALER

FT_DEALER keys: id, the table saves data of dealers

FT_C_DEALER keys: id, FT_DEALER_id, the table saves data of contact persons of dealers

Contact persons are registered to dealers, a contact person cannot exist in FT Web without to be registered to a dealer. The relation is a many to one relation, in other words a contact person can only be register to one dealer but a dealer can be registered by many contact persons. The FT_DEALER_id is a foreign key and FT_C_DEALER is a child table and FT_DEALER is a parent table.

FT_C_HAULIER – FT_HAULIER

FT_HAULIER keys: id, the table saves data of hauler

FT_C_HAULIER keys: id, FT_HAULIER_id, the table saves data of contact persons of haulers

Contact persons are registered to haulers, a contact person cannot exist in FT Web without to be registered to a hauler. The relation is a many to one relation, in other words a contact person can only be register to one hauler but a hauler can be registered by many contact persons. The FT_HAULIER_id is a foreign key and FT_C_HAULIER is a child table and FT_HAULIER is a parent table.

FT_DRIVER – FT_HAULIER

FT_DRIVER keys: id, the table saves data of drivers

FT_HAULIER keys: id, the table saves data of hauler

Drivers are registered to haulers; a driver can exist in FT Web without to be registered to a hauler. The relation is a many to one relation, in other words a driver can only be register to one hauler but a hauler can be registered by many drivers.

FT_TEST – FT_TEST_FAV – FT_USER

FT_USER keys: id, the table saves data of the users, for example privileges for users

FT_DIARY_FAV keys: FT_TEST_id and FT_USER_t0nr, the table is a reference table

FT_USER is the parent table for the child table FT_DIARY_TEMPLATE; a template cannot exist without a user.

FT_DIARY_FAV is a child table and FT_USER and FT_TEST are both parent tables.

FT_TEST – FT_DIARY – FT_USER – FT_VEHICLE

FT_TEST keys: id, the table saves general data of all field tests

FT_USER keys: id, the table saves data of the users, for example privileges for users

FT_DIARY keys: id, FT_TEST_id, FT_USER_t0nr, the table saves data of the messages

FT_VEHICLE keys: chassis_id, the table saves general data of all vehicles

FT_DIARY is a child table and FT_USER and FT_TEST are both parent tables. The vehicle reference in the FT_DIARY table is not mandatory.

FT_USER – FT_DIARY_TEMPLATE

FT_USER keys: id, the table saves data of the users, for example privileges for users

FT_DIARY_TEMPLATE keys: name and FT_USER_t0nr, the table saves templates for diary posts

FT_USER is the parent table for the child table FT_DIARY_TEMPLATE; a template cannot exist without a user.

5. Design

5.1 Design pattern

Model view controller is a fairly common design pattern and is used in many applications. Because the fact that PHP is a server-side scripting language the MVC pattern is implement a little bit different then in for example the very object-oriented language Java.

Model – Handling the database connection and processing data

View – Handling the visual like HTML code

Controller – Handling the combinatorics

5.2 The code for the MVC pattern:

5.2.1 index.php

index.php loading all server configuration files like correct file path. The PHP's auto loads function allows to automatic load modules and correct files by try to load the modules.

Index.php file call the interpreter for the web address (e.g.www.example.com?d=foo&c=foo2&id=1). The interpreter tells which modules too load.

The index.php create the model object and the controller object and at last calls the index() function for the controller to execute the combinatorics.

```
/**** index.php ****/

/*
server_config
loading correct files
autoload();
*/

/** Create the model **/
$oModel = new ftModel();

/** Create the controller **/
$oController = new ftController($oModel,$_GET);

/** Call function in controller to do the combinatorics **/
$oController->index();

/**** END ****/
```

5.2.2 Controller

The controller class creates the view object and when the index() function is called the combinatorics are executed and in the end of the index function the view's display() function is called.

```
/**** controller.php ****/

/** Controller class **/
class ftController extends controllerModule
{
function __construct($oModel,$get)
{
/** Create the view **/
$cView = new indexView($oModel);
parent::__construct($cView,$oModel,$get);
}
function index()
{
/** Combinatorics goes here**/
/**
ex.
If($_GET['var'] == 0)
                    $var = 'Hello world';
else
                    $var = 'Good bye world';
**/

/** Call function that **/
$this->oView->display();
}
}

/**** END ****/
```

5.2.3 View

The view has a function display() which contains the HTML code for rendering the visually content.

```
/**** view.php ****/

/** View class **/

Class indexView extends viewModule
{
private $oModel;
function __construct($oModel)
{
parent::__construct($oModel);
$this->oModel=$oModel;
}
function display($data,$misc_data)
{
/** HTML code goes here **/
}
}
/**** END ****/
```

5.2.4 Model

The model contains the database specific functions for the specific page of the system. The model also taking care of the access rights, every time a user loads a new page the model is checking if the user has rights to access the page

```
/**** model.php ****/

/** Model class **/
class ftModel extends modelModule
{
publicfunction __construct()
{
parent::__construct();
}
}

/**** END ****/
```

5.2.5 All

The controller, view and model extends from parent classes. This will reduce the number line of codes. It will also make it easier to get changes of all pages just to modify one file. One major advantage of this is to make it easier to manage a login system. The login system may be change over time or need to be recoded for security purpose.

5.2.6 Abstract classes

The controller and the view has one abstract function each index() and display(). It creates a database object and will be responsible for all connection to the database. The model will be used from the controller's combinatorics each time a database connection is needed.

```
/** Abstract classes extends by all controllers **/

Abstract Class controllerModule
{
public $oView;

public $oModel;

public function __construct($oView,$oModel,$get)
{
$this->oView=$oView;
$this->oModel=$oModel;
}
/** Returning the database object**/
functionget_db_manager()
{
return$db_manager;
}
/** All controllers must declare an index function**/
abstractfunction index();
}
```

```
/**** END ****/

/** Abstract classes extends by all controllers **/

AbstractClassmodelModule
{
public$db_manager;
public$current_user;
publicfunction __construct()
{
/** Create the data base manager object **/
$this->db_manager=newDBManager("127.0.0.1","root","","test");
}
}
/**** END ****/



/** Abstract class extends by all views **/
AbstractClassviewModule
{
private $oModel;

public function __construct($oModel)
{
$this->oModel = $oModel;
}
/** All controllers must declare a display function**/
abstract function display();
}
/**** END ****/
```

6. Solutions

6.1 Protus

Field tests are made for testing vehicles or parts of a vehicle. Field tests result in knowing new errors in the testing objects. These errors needed to be reported and therefore a system called Protus is used for reporting. Once a tester finds a new error this tester describe the error into the Protus system. The error reporting will describe what type, the priority, the points (high points means high malfunction), which people are involved etc.

The Field test group is stating that there is a use for combining Protus and FT Web. This combining function will have a better overview for vehicles in field tests. The view for showing points will be designed by the users' requirements.

Problem

A problem arise when spooking to the Protus team, they did not allow any access from FT Web to Protus. There is a way to solve this problem. PHP have a library for reading and writing XML files, when a XML file is read into PHP the function DOMDocument::load is called:

```
$dom = DOMDocument::load( $FILE”);

$rows = $dom->getElementsByTagName( ‘Row’ );

$cells = $row->getElementsByTagName( ‘Cell’ );
```

The Protus connection can be solved by the PHP library for reading and writing XML files. First the admin of FT Web saves the Protus data into a XML file, and then the admin upload this XML file to FT Web. The FT Web system will read all the attributes and insert them into a database table. Every time a XML file is read into the FT Web system the Protus database table data is deleted before the new data is inserted. This is made because the FT Web system does not know which new data is or which old data in the new XML file is. There are some drawbacks like new data written to Protus is now shown in FT Web before the admin has downloaded the new XML file from Protus and uploaded it to the FT Web system. Another drawback is that an admin need to do this update on regular basis and this requires time.

6.2 Solving the future flexibility

Flexibility for the future is a very important topic for FT WEB. The field test team is increasing and more places and more features are used in field tests. There are today 4 different field test sites Lyon, Gothenburg, Brisbane and London. In the future there will be even more places where the field tests will take place.

When a new unit (vehicle, field test, dealer, driver or hauler) is added to the system many attributes are registered together with the unit. As for example for a vehicle, the attributes country, owner and chassis number are registered together with the vehicle. These attributes must be pre-typed to make the database search able for specific pre-typed keys.

The key for solving this are drop-lists and references. When designing the system groups the drop-lists are not hard coded. The system will fetch all names available for the drop-list from the database; these names will be able to choose for the user of the system. Only the users in the admin or the developer class have privileges to add new names to the system.

6.3 Report generating function

The field test team working with their field tests over long time periods. Under these time periods there is a need of status reports. A status report contains information of the current state for the field test. The current state of a field test describes by information about the vehicles in the field test. The information about the vehicles is information about the vehicles mileage and the vehicles days of stand still. The report also reports about the important happenings. In the FT Web system these important happenings are reported by the diary function. The FT Web system provides an auto generating status report function. The purpose for such a function is that the report will always have a standard style and it will save all the time it takes for the field test team to write a status reports.

The status report will contain following data:

- Field test name
- Date
- Issuer
- List of all vehicles registered to the field test. For every vehicle the objects registered to the vehicle is listed together with information about application cycle, Protus points, number of Protus reports, mileage and standstill data.
- Important diary posts posted for the field test.

6.4 The vehicle status problem

All vehicles have a status which will be displayed on different pages. There is problem if a vehicle has the same status for all field tests. The status is changed by for example a standstill posts. A problem will arise if two users one from each field tests posts a standstill posts on different days. The status field will not be consistent. Therefore all vehicles have one status for each field test. If one error exists in a field test, this error will not affect the other field tests.

There are 4 statuses:

- Inactive, the vehicle is registered in a field test but not start running.
- Running, the vehicle is registered and running within a field test.
- Standstill, the vehicle has gone from running to standstill within a field test.
- Planned standstill, the vehicle has gone from running to standstill within a field test.
- Stopped, the vehicle has stopped

6.5 MVC workflow

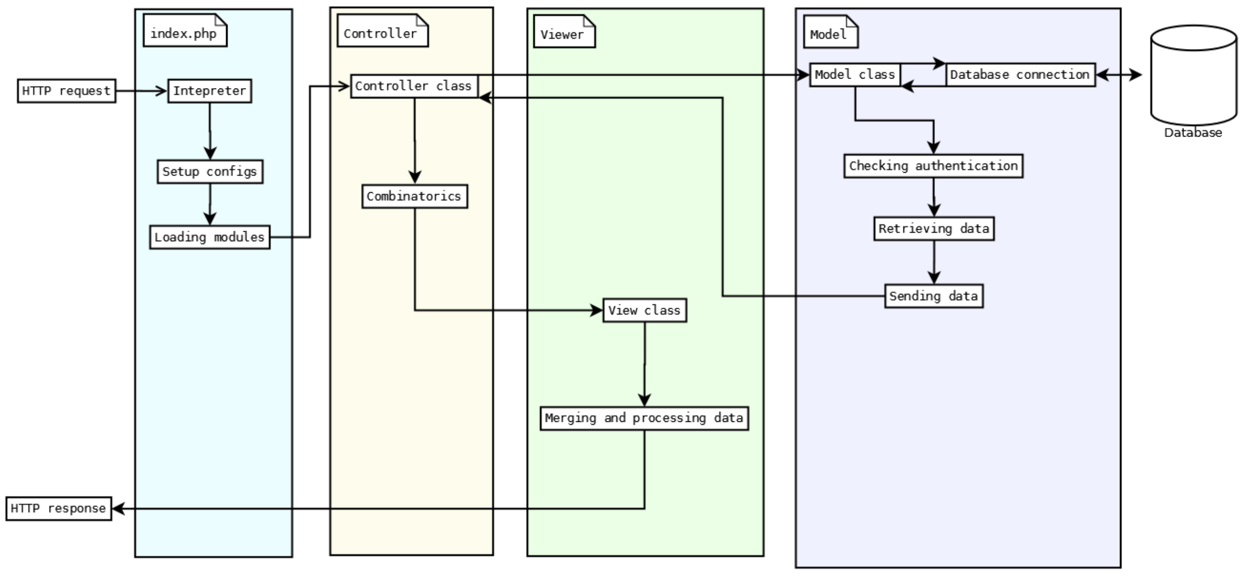

Figure 2. MVC workflow

The figure 2 shows how the modules in the MVC pattern are interacting with each other. Each web page has its own view, controller and model in different files. The requests always evoke the index.php file where the Interpreter is called. The interpreter decides which modules to load. The configuration files for the server is loaded for example the root path is defined as a global variable. The auto_load() function helps the index.php load correct files for all modules. The index.php file creates a model object and a controller object, the controller object get the model object as a parameter. When the controller object is created the controller first creates the view object. The flow of data can be seen in the picture.

7. Description of the user interface

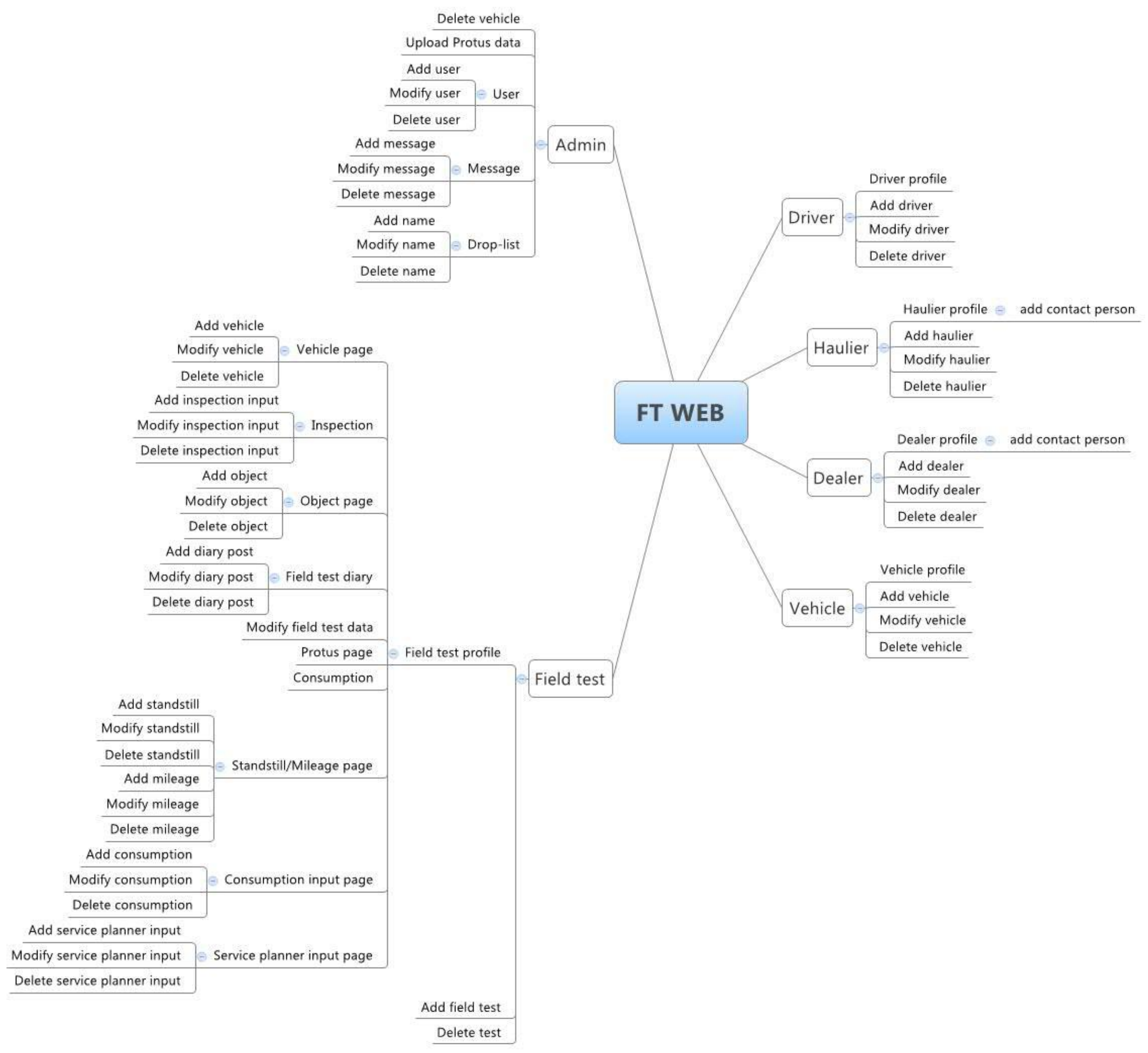

Figure 3. Interface scheme

"Add" pages
All add pages simply add a unit to the system
"Modify" pages
All add pages simply modify a unit to the system
"Delete" pages
All add pages simply modify a unit to the system

7.1 Vehicle

Listing all registered vehicles stored in the database, attributes as chassis number, vehicle id, owner, serial number, dealer and hauler name etc. There is one attribute that shows the current status for the vehicle, the states are running, stand-still, planned standstill or stopped.

Vehicle profile
The profile page is a structured page for showing one specific vehicle. It is possible to upload an image for the vehicle.

7.2 Hauler

Listing all registered haulers stored in the database, attributes as country, brands, name, fleet size and organization number etc.

Hauler profile
The profile page is a structured page for showing one specific hauler together with the contact persons for the hauler. It is possible to upload an image for the hauler and add or remove contact persons.

7.3 Dealer

Listing all registered dealers stored in the database, attributes as country, brands, name, website address and organization number etc.

Dealer profile
The profile page is a structured page for showing one specific dealer together with the contact persons for the dealer. It is possible to upload an image for the dealer and add or remove contact persons.

7.4 Driver

Listing all registered drivers stored in the database, attributes as country, brand experience, name, weight and length etc.

Driver profile
The profile page is a structured page for showing one specific driver together with the contact info for the driver. It is possible to upload an image for the driver.

7.5 Admin

Admin page controls all functions.

Users
The system only allows registered users to use the system. It is possible to add, modify and delete users from the system at the users' page.

Messages
The start page for the system shows news from the administrators. It is possible to add, modify and delete messages from the system at the messages page.

Drop-list
The drop-list page shows all drop-list names for the system. It is possible to add, modify and delete names from the system at the drop-list page.

Delete vehicle
Foreign keys preventing data corruption in the database. This means that a vehicle cannot be deleted before the other data connected to the vehicle is deleted. Example of such data are

mileage data, standstill data etc. It is possible to delete a vehicle and all its data from this page. This will make it easier to delete a vehicle.

Upload Protus data
Due the limitation of connection to Protus the admin have an upload function. The admin can download data from Protus in XML form and then upload this XML file into the FT web system.

7.6 Field test

Field test profile
A field test consist of vehicles, objects involved people.

Data gathering:

- Mileage
- Standstill
- Diary posts
- Consumption
- ER-reports

Presentation of:

- Protus reports
- Vehicle status
- Consumption over time
- Mileage over time
- Information about the field test over time

Visible at the profile page for field tests:

The profile page shows the current status for the test, there are five states:

- *Inquiry* – When the test is created then it is in the first state “Inquiry”.
- *Planning* – A test needs to have at least one test account to be in status PLANNING
- *On-going* – A test needs to have at least one vehicle with start date to be in status ON-GOING
- *Termination* – A test needs to have all vehicles with values start and end dates to be in status TERMINATION
- *Ended* – A test needs to have one ER report with FINAL report value to be in status ENDED

A summary table is shown where all vehicles are listed together with Protus points, mileage summary, objects attached to vehicle and status (ex. Running).

Vehicles
This page lists all the vehicles in the field tests.

Test objects
This page lists all the objects in the field tests.

Mileage standstill
This page consists of two views, one view lists the mileage and standstill over time and the second view lists all mileage information for all vehicles.

When adding a standstill it is possible to add information about the standstill like reason of the standstill, mileage, if there is a loan vehicle, if there were a planned standstill.

Protus reports
This page shows all Protus reports for the registered objects for the field test. When the user hover the mouse pointer over one of the reports the system will fetch the information of this report and show the report in a popup. The report contains of TODO

Diary
The diary page shows the diary posts and standstill posts.

A diary post contains:

- Diary text
- Create date
- Creator
- Mileage (not mandatory)
- Vehicle (not mandatory)

A standstill post contains:

- Text
- Create date
- Start for standstill
- End for standstill
- Vehicle
- If there is a loan vehicle
- If the standstill were planned
- Creator
- Mileage

When adding a diary post it is possible to submit as an announcement and the diary post will then be visible on the profile page for the field test.

Consumption

The consumption page has two views, one view for listing consumption data for all vehicles in one table and one view for adding, modifying and deleting consumption inputs. The second view also listing all consumption inputs in an informative table and calculate data for the vehicle like liter/km consumption, oil consumption and urea consumption.

Service planner
The service planner page will show all inspections for the vehicles, an inspection input can be a service planner input or other inspection task. An inspection input will be scheduled with a date or a mileage.

8. Result

The goal for this master thesis was to design and implement a computer system for a group at Volvo 3P. There was a demand for a new system that should lead to a more efficient way to work with field tests and make other tasks easier to fulfill.
The implementation has gone as plan and FT Web was released in the end of my master thesis. The development has focused on flexibility to make it easier to maintain and the support the system in the future.
The first users were the field test group in Gothenburg, in the future the users will be spread all over the world.
The planned number of users after the release was estimated to:

Role	People
Field testers around the world	30-60
Other	30-60

Table 1. Users

The future work will be for the local IT group at 3P to maintain and support the system.

9. References

[1] Coggeshall, JC. (2004) PHP 5 Unleashed. Indiana: Sams.
[2] PHP.net, Intro what can PHP do
http://se2.php.net/manual/en/intro-whatcando.php (Accesed: Oktober 2011)
[3] .NET Framework 4
http://msdn.microsoft.com/en-us/library/w0x726c2.aspx (Accesed: November 2011)
[4] ASP .NET vs PHP
http://www.whr.se/blog/programmering/asp-net-vs-php (Accesed: Oktober 2011)
[5] PHP vsASP .NET comparision
http://www.comentum.com/php-vs-asp.net-comparison.html (Accesed: Oktober 2011)
[6] Komplett
http://www.komplett.se (Accesed: Oktober 2011)
[7] Mono project
http://www.mono-project.com (Accesed: Oktober 2011)
[8] PHP
http://www.php.net/downloads.php (Accesed: November 2011)
[9] Oracle vs. SQL Server
http://www.dba-oracle.com/oracle_tips_oracle_v_sql_server.htm
(Accesed: November 2011)