

**Hogeschool van Utrecht**

Faculty of science and technology  
International software engineering for business applications

# **Web Application Framework**

Final project proposal (ver.3)

*Authors:*

**Daniel Rozsnyó**

Brno University of Technology, Faculty of information technologies  
*rozsnyo@kn.vutbr.cz*

**Pavel Cadorský**

Brno University of Technology, Faculty of information technologies  
*xcader00@stud.fit.vutbr.cz*

## Contents

Contents.....	1
Company outline .....	2
Position of students.....	2
Overall description.....	2
Global formulation.....	2
Back-end system.....	3
Multidimensional file system.....	3
Persistent applications.....	4
Front-end system.....	4
Applications .....	5
Development techniques.....	5
Project activities.....	5
Produced products.....	6
Scope of the project.....	7
Project planning.....	8
Challenges.....	9

## Company outline

We, the undersigned authors, have decided to work on this project on our own. There are several reasons for this choice, just to mention one - we get the opportunity to realize our ideas, visions and dreams in a larger scale than usually during the school. We have a lot of own small development experience from the past, but we were unable to make a meaningful product out of them due to lack of free time.

Another reason is that we imagine our future not on the bottom of the work chain, as programmers, but rather as the leaders of a larger development team. If we get the chance to do the project just on our own, we also get a direct experience with project management which is incomparable with the management skills required for a project realized under someone else's control.

## Position of students

Since we will not work in a company, this point can look as quite useless. But it isn't. We will work as a small company. The primary function of a company is to achieve some aims - our primary aim is get the projected system to be useable and working.

To keep on a right way we need a kind of steering force. We are two for the project and it will be definitely divided into small parts or modules. The responsibility for them will be divided between both of us equally. Always the non-responsible person will provide the steering force needed.

## Overall description

As the name of the project indicates, the aim is to create a framework for web applications. This framework will be not only a tool-box with various classes waiting to be used, but it will behave as an operating system with an advanced graphic user interface accessible from any modern web browser. We will create a basic set of system applications and management tools (more specified in chapter Applications).

Further usage of the framework will involve network management tools, an information system development tool and a new programming environment which will be independent on used programming language. These applications will be not fully realized in this project but we start to design them since we would like to continue in the developing after we return to our home country.

## Global formulation

The project is based on several technologies developed fully by the authors such as cross-platform user interface and the MDFS. These technologies are generally not implemented yet, but various tests and checks were made to verify the feasibility.

This section contains the description of building blocks of the framework and also some more information about our own technologies that we want to implement.

## Back-end system

The back-end system is the core of the framework. It consists of a storage place called multidimensional file system and an application runtime environment which is responsible for running the applications on the back-end server. Since all the applications are running on the back-end it has to be a very stable, secure and scalable solution. The framework is designed as multi-tier application and the back-end system includes the database and application logic tier.

The back-end system will be implemented using PHP scripting language for its easy usage. The code of the back-end processes and applications will be partly created on the fly - from the multidimensional file system. With this approach we can replace in the future the Apache web server and PHP scripting language with our own application server written in C++. This will give us a dramatic rise of performance because the C++ program will be compiled into a machine code which runs much faster than PHP scripts.

## Multidimensional file system

One of our own technologies developed especially for this kind of projects and used first time is this central storage place. The multidimensional file system provides storage of various kinds of data based on their meaning, not on their notation (syntax). When some data is needed, the meaning is converted into the desired notation. Since there can be several notations of the same data, the file system got the name multidimensional.

Since the MDFFS should be compatible with the today's technologies, the data model is based on XML. It says that the elementary data is a node which has a name, namespace and optional value. Furthermore, the node can have attributes which are named with name and namespace and always have a value. The MDFFS data model is not exactly the same as XML's. In MDFFS is allowed the existence of multiple root nodes and also the relations between nodes are not so strict - two nodes have a one way relation. This means that a node can be shared by multiple parent nodes, in XML the node has only one parent node. The advantage of this feature is that you can create much more complex structures in MDFFS, e.g. in ACL (access control lists) the user can belong to more than one group, so the management will be easier. Another improvement is that it saves storage place - if we have a footer/header on every page, we can link them directly into more pages - without using shortcuts or references which need additional code.

The components of the MDFFS are the database abstraction layer, the core, the transformation engine and the dimension cache.

The database abstraction layer is the smallest part, but enables the MDFFS to use different underlying relational databases. Firstly we make support only for the MySQL open-source database.

Above the database abstraction layer is the core of MDFFS. It has two parts where one is responsible for storing, lookup and retrieving the data and the other

one represents a node and makes possible to access its value, attributes and child nodes.

The transformation engine makes conversions of data stored in MDFS (accessible as XML) into another format. This engine can be used not only for a simple conversion (converting a database metadata from XML into SQL create database/table statements), but also for generating some data parameterized by the input XML (creating a C++ or ObjectPascal/Delphi list class from a simple declaration of the list element containing the properties of the element).

The dimension cache will store the result of the transformation and when there will be a request for transforming again, the stored value will be used instead of doing the conversion.

### **Persistent applications**

As we mentioned before, the applications run on the back-end server, so the functionality of our framework does not depend on the clients and/or on the quality of their connection. Every user will run a set of applications in a session. This session will be persistent so the applications will not be terminated when a communication error occurs between the client and the back-end system.

As an extension of this functionality, we get a lot of freedom. The user can disconnect from a session and continue its work at another time, from another place, another machine and using another host operating system.

But there are more features which will be possible using the above mentioned system. One is that the user can run as much sessions as he wants. Another one is important for companies - using the framework, a company can create a centralized virtual working place for employees. All the applications and the documents will be easily accessible for everyone who will meet the security settings, of course.

### **Front-end system**

The front-end system is also known as the presentation tier and acts as a client for the back-end system. It provides a cross-platform graphical user interface based on web standards and is easy to use for application user, because the controls used will be well known from modern operating systems (windows, buttons, input boxes and other).

The best way how to create a today's state-of-the-art user interface is using a web browser. There are standards as XHTML and CSS which can be used and ensure that the result will look the same when using different browsers. Dynamic functionality can be achieved by using DHTML and JavaScript technologies which are also present in every modern web browser. The communication between the user interface and the back-end system will use XML format and HTTP (hypertext transfer protocol).

Implementation: The front-end system is composed from a few system components (will be similar to OS kernel and communication layer) that take care of communication with back-end server (this means composition, sending, receiving and interpretation of messages in XML format).

The main function of front-end is graphical presentation of processes running on back-end server and interaction with user. There is a set of standard features

provided by the front-end system. These are functions like dragging objects, moving, resizing windows and standard actions on button click. But the framework will provide much more functionality than conventional operating systems - e.g. sharing of applications (two or more users can work on the same project in one application, the changes made by one will be immediately visible to the other ones), sending applications from one user to another one or to a group of users. Example of sending application: You can have a text document you just finish and you need to send it to proof-reader so you just click send editor application to proof-readers group and one of the proofreaders accept this application and can make the corrections directly to your document. After he finishes, he sends it back to you. This operation in today operating system requires saving the document, creating a new mail, including the document as attachment, on the receiver side everything in reverse order and all that again for sending back the checked document.

## Applications

The framework alone is not much interesting since it is only a system which allows running applications - like the hardware without operating system. We need to make some applications which will show the power of the framework and the advantages of the technologies used in it.

There are planned three basic categories of applications - network management applications, information system development tools and application development tools. There is also a small group of system tools which are inseparable from the framework, like configuration tools (setting default behaviour of front-end, skins, colour schemes, wallpapers, icons of shortcuts on desktop).

## Development techniques

Each of us (developers) will work on his part of work, but there are also common parts (like communication between front-end and back-end) and for these we must specify standards. We will adapt similar method that is used by W3C (World Wide Web Consortium). Every standard has its own document with current version, history and comments on it. This way we produce high quality (internal) standards which are easy to understand and to implement by developers not involved into our project. The method is based on documents like proposals, drafts, recommendations, testing and final reports.

We will implement our framework using PHP server side scripting language. The reason why to use PHP and not C/C++ is that we do not have experience with writing applications of this scale so the realization of the project would be quite impossible. Using PHP and Apache web server we can focus on the problem and do not need to implement modules which were already implemented (eg. http server).

## Project activities

The common activities which will be done for every software component are:

- Making a design - using UML classes for PHP and JavaScript components, use cases and process diagrams for framework applications, ERD for database parts of framework

- Implementation - creating collection of JavaScript and PHP scripts that will form the framework
- Testing - because framework is build from components, every piece will be tested separately. If all components are bug-free, then the whole system built from these components will be bug-free
- Integrating components into the framework - to make components usable we need to create interface for each of the components and afterwards integrate these components together into one system

Among these activities there will be also a detailed documentation written. This documentation should be enough for new developers who want to improve existing components.

## Produced products

There will be various products produced - see the following list:

- software components of the framework
  - back-end system
    - multidimensional file system
    - transformations required by mdfsEditor and dimensions for demonstration (SQL, EBNF)
    - application runtime environment - communication module, message queue and message processor
  - front-end system
    - functional system core including kernel, window manager and communication layer
    - visual components - windows, buttons, tree and mdfsTree (component for presenting and editing data stored in MDFS, will be derived from tree component)
- the framework system
  - system working on Linux platform with Apache web server and PHP scripting language and Mozilla Firefox web browser
- installation manual
  - complete step-by-step guide for installing the framework system
- framework user's guide
  - guide for users, information how to work with desktop and user interface (windows, menus, buttons, ...)
- the thesis
- sample application(s)
- installation CD

## Scope of the project

As the scope of the project we give you the required functionality list:

### MDFS

- storage of nodes composed of namespace, name, optional value, attributes having namespace, name and value
- storage of relations between nodes (one-direction, M:M relationship)

### Transformation engine

- conversion of data starting with a specified node into a dimension specified by name (e.g. SQL) and allowing additional parameters to specify the details of output
- result types are specified by mime (text/plain, image/jpeg, etc)

### Dimension cache

- caching of the result, making usage statistics
- storing of dimension dependencies
- invalidating the cached values on data change using dependencies
- invalidating based on time (specified as - value valid till some time)

### Application runtime environment

- making a connection with front-end, receiving messages
- queuing the messages, allowing asking for their processing status
- executing the code based on the message - processing the message

### Front-end core

- engine for loading components
- XML parser
- kernel components
  - making a connection with back-end
  - send & receive messages (both synchronous and asynchronous)
  - message processing, instructions queuing and routing to target components
  - loadable modules support (shared libraries for user applications)
- window manager
  - showing, hiding, resizing and moving windows
  - z-order handling
- visual components
  - window
  - button, edit box, tree, mdfsEditor

### Front-end system

- desktop
- login application (authentication of users that want to use applications)
- debugging applications
  - kill
  - task manager
  - message monitor

### Requirements for applications

- applications will be stored in MDFS and retrieved on demand
- format of application is a simple XML document
  - specification of application's windows
    - application can have multiple windows
    - controls placed in windows (buttons, trees,..)
  - linked modules (shared libraries)
  - mapping of events onto actions
  - actions composed of:
    - send message to back-end
    - run a script
  - reactions - mapping of incoming messages onto actions

### The thesis

- one thesis composed of chapters written by the authors separately (each chapter will have its own author)
- during the realization period we provide finished chapters after milestones (see the planning)

## Project planning

The project will be realized in the period of 1.2.2004 - 25.5.2004. The final version of project planning is in the following table:

Month	Planned tasks
February	Back-end <ul style="list-style-type: none"><li>• Database abstraction layer</li><li>• MDFS core</li></ul>

---

	Front-end
	<ul style="list-style-type: none"> <li>• Learn about the used standards on very detailed level</li> </ul>
March	Back-end
	<ul style="list-style-type: none"> <li>• Transformation engine</li> <li>• Dimension cache</li> </ul>
	Front-end
	<ul style="list-style-type: none"> <li>• XML parser</li> <li>• front-end core (window manager &amp; communication layer)</li> <li>• visual components</li> </ul>
	Thesis
	<ul style="list-style-type: none"> <li>• All about MDFS (design &amp; implementation)</li> </ul>
April	Back-end
	<ul style="list-style-type: none"> <li>• Application runtime environment</li> </ul>
	Front-end
	<ul style="list-style-type: none"> <li>• MDFS editor &amp; applications</li> </ul>
	Thesis
	<ul style="list-style-type: none"> <li>• All About applications</li> </ul>
May	Project
	<ul style="list-style-type: none"> <li>• Finishing, testing</li> </ul>
	Thesis
	<ul style="list-style-type: none"> <li>• Use cases, Future</li> </ul>

---

## Challenges

The first challenge will be the realization of this project itself, turning the framework into a production-level system - neither a technology demonstration nor a development level system full of security holes.

Furthermore, the challenges are that the we, as the developers, need to work in one team, as one subject, and produce a software product not for our own use as we did in the past but with a respect to the needs of others, the users and our future clients.

26. 3. 2004

Utrecht, the Netherlands