

# FND RESULT SERVER

Software Detailed Design

01.06.2011

FND Software

Ahmet AKYOL e1408558

Gürkan KUŞ e1502558

Hakan EMEKÇİ e1502327

Rıdvan TANIK e1502707

## Table of Content

Table of Content .....	2
1. Introduction.....	4
1.1 Problem Definition .....	4
1.2 Purpose.....	5
1.3 Scope .....	5
1.4 Definitions, Acronyms and Abbreviations .....	5
1.5 References .....	5
2. System Overview .....	6
3. Design Considerations.....	9
3.1 Design Assumptions, Dependencies and Constraints .....	9
3.1.1 Portability .....	9
3.1.2 Reliability .....	9
3.1.3 Maintainability .....	9
3.1.4 Security .....	10
4.1.2 Availability.....	10
4. Data Design .....	10
4.1 Data Description and Data Flow Diagram .....	10
4.1.1 Entity Relation Diagram.....	10
4.1.2 Data Flow Diagrams.....	15
4.2 Data Dictionary .....	18
5. System Architecture.....	19
5.1 Architectural Design.....	19
5.2 Description of Components.....	20
5.2.1 Component “Admin Tool” .....	21
5.2.2 Component “FND Resul Server” .....	23
6. User Interface Design.....	24
6.1 Overview of User Interface.....	24
6.2 Screen Images.....	25
7. Detailed Design.....	26
7.1 FND Result Server Node .....	26
7.2 Admin Tool Client.....	28
7.3 System RDBMS.....	29
7.4 NoSQL Data Store .....	30

7.5 Data Store Manager.....	31
7.6 Admin Tool.....	33
7.6.1 JSF Module.....	34
7.6.2 JPA Module.....	36
7.6.3 EJB Module.....	37
7.7 RDBMS for Admin Tool.....	38
8. Libraries and Tools.....	39
8.1 JSF (Java Server Faces) .....	39
8.2 Netbeans .....	39
8.3 Jboss Netty.....	40
8.4 Membase.....	40
8.5 Java.....	41
9. Time Planning (Gannt Chart) .....	42
9.1 Term-1 Gannt Chart .....	42
9.2 Term-2 Gannt Chart .....	43
10. Conclusion .....	44

## Software Detailed Design Report

### 1. Introduction

This report is intended to describe the detailed design of "FND Result Server" project of group FND. It provides clarification and design of the project in a way that not only we- the project owners and implementers- but also any IT professional can understand the road map.

#### 1.1 Problem Definition

With the developments of technology in the web platforms many different services are offered to people to provide accessibility of information, they need, whenever and wherever they want. But of course, although the improvements of technology in this area, there are still extreme but encountered situations or needs that even this improved technology can not meet such as extremely high number of clients trying to request information from your site at the same time.

Many web solutions use general purpose databases to store information and general purpose servers to response the requests of the clients. It is very logical to use these systems in many applications but if there is possibility for your application to encounter extreme situations, then the response that these general purpose systems provides can not be offered to client at a reasonable time. Considering this, we decided to develop this project to make possible that people can get the information they need even in extreme situations.

## 1.2 Purpose

FND Result Server is a system that makes possible web applications

- ✚ to response to considerably more numbers of requests than general purpose systems do.
- ✚ to response to each client request faster than any general purpose system.

## 1.3 Scope

FND Result Server is a system that can be collaborated with any web application which uses general purpose databases such as Oracle, MySQL, MSSQL...

## 1.4 Definitions, Acronyms and Abbreviations

FND: Feasible Network Devolopment

http : Hyper Text Protocol

JSF : Java Server Faces

JPA : Java Persistance API

EJB : Enterprise Java Beans

NIO : New I/O

UI : User Interface

RDBMS : Relational Database Management System

## 1.5 References

Netbeans is the integrating development tool that we are going to use for implementing the FND Result Server : <http://en.wikipedia.org/wiki/Netbeans>

Scrum is an iterative, incremental methodology for project management often seen in agile software development:

[http://en.wikipedia.org/wiki/Scrum\\_%28development%29](http://en.wikipedia.org/wiki/Scrum_%28development%29)

MySQL is a relational database management system (RDBMS)[1] that runs as a server providing multi-user access to a number of databases :

<http://en.wikipedia.org/wiki/Mysql>

*The Netty project* is an effort to provide an asynchronous event-driven network application framework and tools for rapid development of maintainable high performance & high scalability protocol servers & clients: <http://www.jboss.org/netty>

Membase is a distributed key-value database management system, optimized for storing data behind interactive web applications : <http://www.membase.org/>

## 2. System Overview

System has following components:

### Administration Tool

In order to make system aware about the database tables which will be included in the system, an administration tool is going to be created. This administration tool which can be used by only the registered admins is going to help admins to retrieve the necessary data from database and insert it into the membase in this system. This tool is also going to make admins embed the pre-prepared fancy user interface

templates into the system, so that every application which uses this system will have the freedom of using its own user interface. It means that there will be no restriction of using a standard template. In addition by using this tool, admins will be able to make configurations on the server side such as setting the number of FND Result Servers which will be used in system to balance the load of the requests, setting the ip's of these FND Result Servers.

### MySQL Database

This database will have tables to hold the information about users who are admins of the system, information about the configuration of the FND Result Servers and user interface templates which will be embedded to the system in each project. Also it holds information about the projects which are created before so that by editing the properties of these projects they can be made use of in the future.

### FND Result Servers

These servers will be specialized servers to meet the needs of the extreme client request cases. If the clients' requests can scale, there must be a way to scale the abilities or numbers of these servers. So, on demand their number can be increased depending on the number of requests. This property although each FND Result Server is able to send responses to very large number of client requests , makes it possible to create multiples of the responses which each FND Result Server sends. In addition, each of these FND Result Servers is able to response to each client request. For instance, say that in our system which we created by using administration tool there

are 3 different FND Result Servers. Any client request can be reposed by any of the FND0, FND1 and FND2. Also there must be a way to stop unnecessary servers.

FND Result Server contains following three components:

➤ Admin Tool Client

This tool is different from administration tool. Every FND Result Server has one admin tool client. It is going to be used to monitor each of the FND Result Server. It will be possible to start, restart or stop a specific FND Result Server. Also this tool is going to give system the ability to see how much memory and resources each FND Result Server uses.

➤ FND Http Server

These servers will be different than general purpose servers which allow very limited number of connections to be opened. Because we are trying to optimize the system to response more client requests, these FND http servers will have the capability of opening considerably more number of connections than general purpose servers do. So, by this property each of the FND Result Servers is going to answer more numbers of client requests successfully. It means that clients will not see "Server Connection Failure" or "Connection to Server Timed Out" warnings.

➤ Membase DataStore

As it is known, general purpose databases save data into a file and retrieve it back from the same file. Because of these very time consuming file operations, the server can send response to client very late in terms of computing timing. In case of an extremely high number of clients trying to make queries from database, these general purpose databases will queue the queries and response each of them one by one and the last transactions will be executed very late. So either the requesting client cannot have a response or the response sent to the client is very late. By using Membase

Datastore, we guarantee that the query will be returned to the client very fast because there will be no mention of file operations.

## 3. Design Considerations

### 3.1 Design Assumptions, Dependencies and Constraints

#### 3.1.1 Portability

FND servers' admin tool will develop with java language thus the system can work on any operating system java virtual machine. There will have installation process on Linux machine because of performance issues for FND server nodes. It can be easily installed for others.

#### 3.1.2 Reliability

When user wants to call the FND system over a given period of time, the system should correctly deliver services as expected by the user. The reliability of the system will be satisfying if it delivers services as specified.

#### 3.1.3 Maintainability

When the FND system is in use, new requirements may emerge. When these requirements emerge, the system should be changeable to accommodate these requirements for maintaining the usefulness of the system. If the system is not maintainable, then the system cannot be modified for new requirements. In this

situation, a new system should be developed to provide new requirements. The maintainability is important in order to avoid from high cost.

### **3.1.4 Security**

The FND system should resist accidental or deliberate intrusions, when users operate on the system. If the system should not resist accidental or deliberate intrusions, then important data may be stolen by hackers. Thus, security of the system shall be low and trust of users shall be ruined. So, security of the system is very important for users.

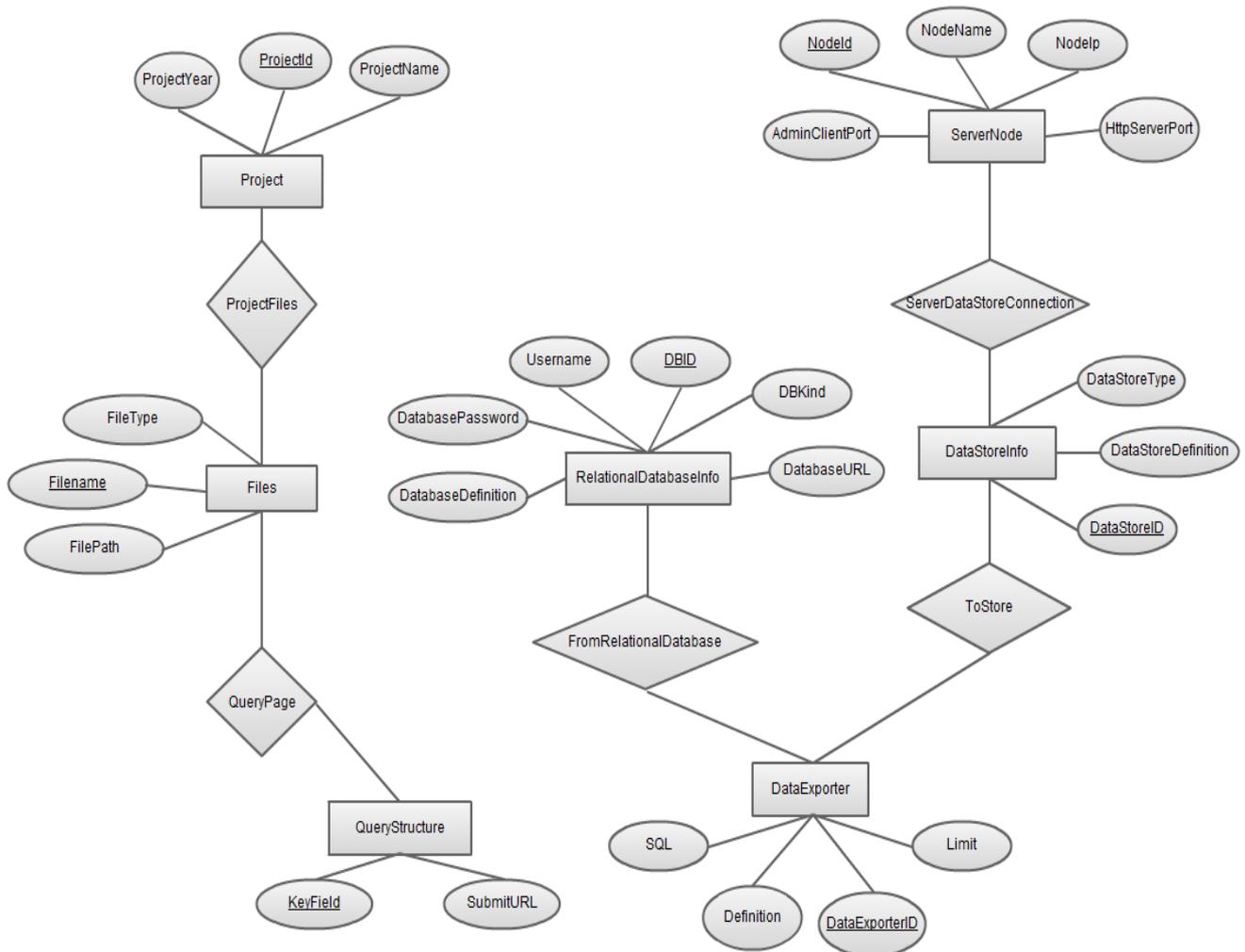
### **4.1.2 Availability**

When the FND system has any request at any given time, system should be available, it should be up and running and able to deliver useful service at this time. The availability of the system shall be good if it delivers services when it is requested. Otherwise, if requests are not responded at any given time then it implies poor availability.

## **4. Data Design**

### **4.1 Data Description and Data Flow Diagram**

#### **4.1.1 Entity Relation Diagram**



[online diagramming & design] [creately.com](https://creately.com)

Here is some important points related to above Entity Relation Diagram:

In order to use the system we need at least two databases. One of them is going to be our system's database named "FNDAdminDB" which will be used to hold information about newly created projects, user interfaces, FND server ip's, query structures and server nodes. The "FNDAdminDB" database will consist of 7 entities and 5 relations. The names of entities and relations and their attributes are specified below. The other databases mentioned above is the databases which the system will be used to retrieve the information are the databases of the companies who wants to use our system.

- "Project" Entity

- i) ProjectId
- ii) ProjectName
- iii) ProjectYear

“Project ” entity is going to be used to hold the data information about the newly created project namely its name, its year and its Id which is the primary key of the entity.

- “Files” Entity
  - i. FileName
  - ii. FileType
  - iii. FilePath

“Files ” entity is going to be used to hold the data information about the query pages to be used namely its name which is the primary key, its type and its path which will tell FND Result Servers where to get the query pages.

- QueryStructure
  - i. KeyField
  - ii. SubmitURL

“QueryStructure ” entity is going to be used to hold the data information about the query structure in the pages namely its KeyField which is the primary key and SubmitURL which is going to specify which page to be sent the form data.

- “ServerNode” Entity
  - i. NodeId
  - ii. NodeName
  - iii. NodeIp
  - iv. AdminClientPort
  - v. HttpServerPort

“ServerNode” entity is going to be used to hold the data information about the Nodes of FND Result Servers which are primary key NodeId, NodeName, NodeIp, AdminClientPort and HttpServerPort.

- “DataStoreInfo” Entity
  - i. DataStoreId
  - ii. DataStoreType
  - iii. DataStoreDefinition

“DataStoreInfo” entity is going to be used to hold the data information about the data retrieved from database and stored in the memory namely the primary key DataStoreId, DataStoreType and DataStoreDefinition.

- “DataExporter” Entity
  - i. DataExporterId
  - ii. SQL
  - iii. Limit
  - iv. Definition

“DataExporter” entity is going to be used to hold the data information about SQL which will be used to retrieve data from database, limit of the returned query rows and definition of the exporter.

- “RelationalDatabaseInfo” Entity
  - i. DBID
  - ii. DBKind
  - iii. Username
  - iv. DatabasePassword
  - v. DatabaseDefinition
  - vi. DatabaseURL

"RelationalDatabaseInfo" entity is going to be used to hold the data information about the database which the data is going to be retrieved. It consists of 6 attributes which are the primary key DBID, DBKind, Username, DatabasePassword, DatabaseDefinition, DatabaseURL.

- "ProjectFiles" Relation

"ProjectFiles" relation holds the relations between "Project" and "Files" entities.

- "QueryPage" Relation

"QueryPage" relation holds the relations between "QueryStructure" and "Files" entities.

- "FromRelationalDatabase" Relation

"FromRelationalDatabase" relation holds the relations between "RelationalDatabaseInfo" and "DataExporter" entities.

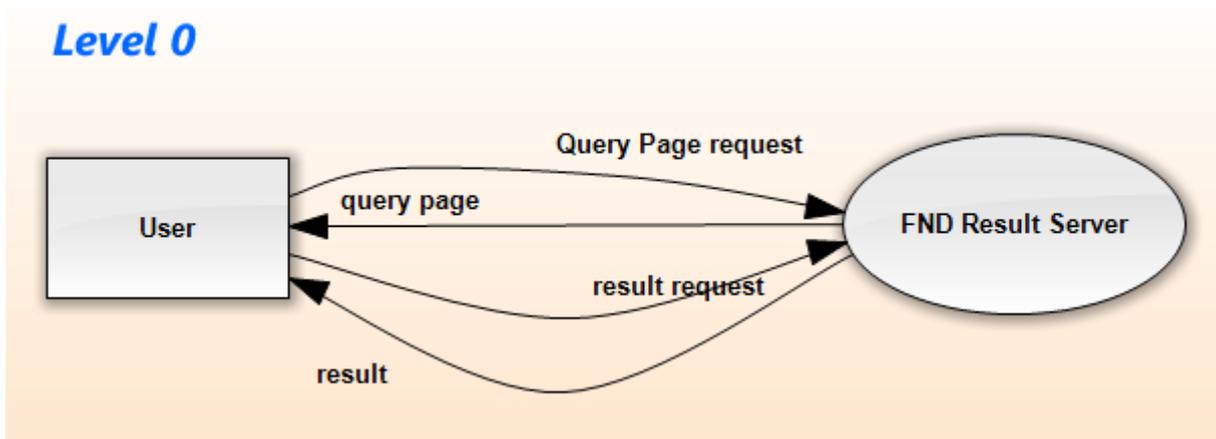
- "ServerDataStoreConnection" Relation

"ServerDataStoreConnection" relation holds the relations between "ServerNode" and "DataStoreInfo" entities.

- "ToStore" Relation

"ToStore" relation holds the relations between "DataExporter" and "DataStoreInfo" entities.

#### 4.1.2 Data Flow Diagrams



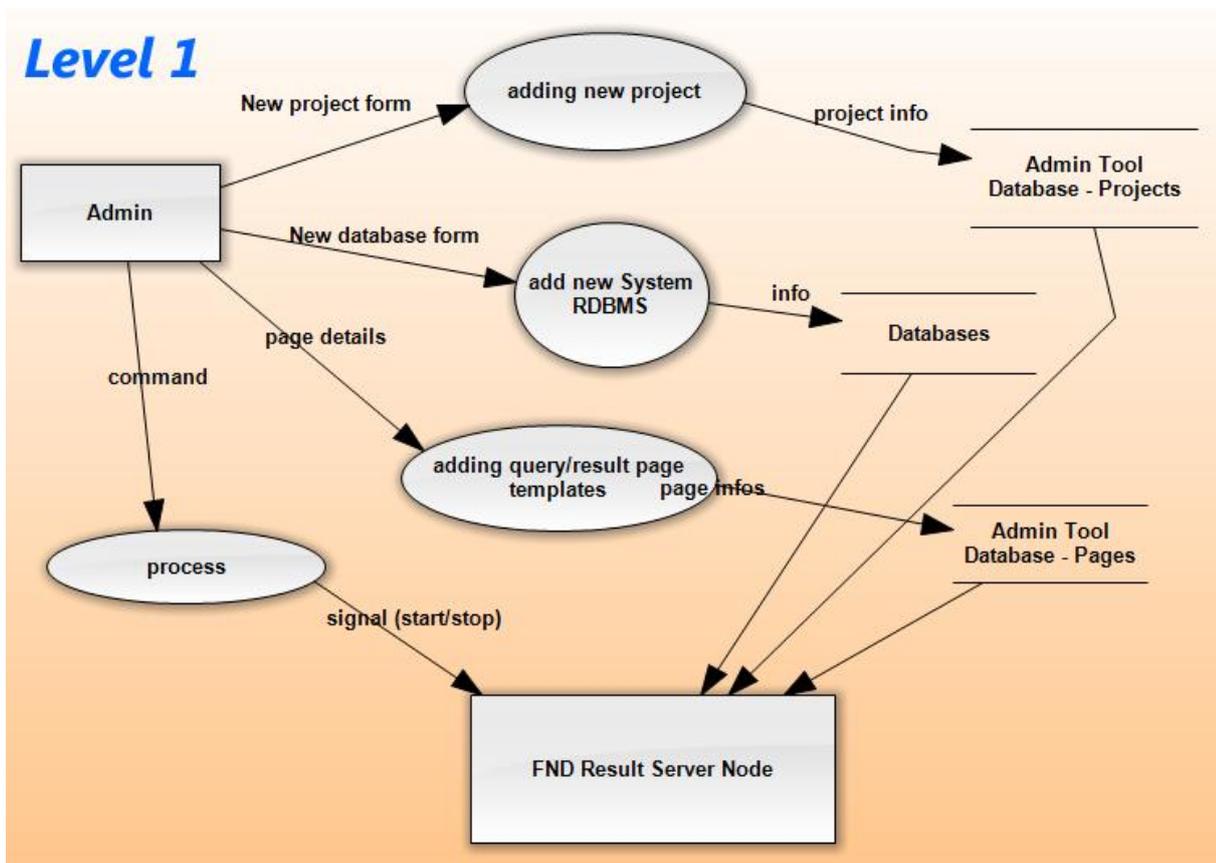
Level 0 Data Flow Diagram represents the flow of data in the process of usage of the system by an end user. In this Data flow diagram actually there are two parts. First one is to show the user the query page, the second one is the result page.

#### Query Page

When user connects to the web page FND Result Servers will get the related template query page stored in the database and prepare it with the related KeyFields and SubmitURL and return it to end user as an http response so that the user can see the page on his/her browser.

## Result Page

When user fills the required textfields and pushes the submit button in the query page, FND Result Servers get the information entered by the user to the known textfields, retrieves the necessary data from memory and gets the related template result page url which is stored in the result servers and prepare its fields with retrieved data and return it to user as an http response so that user can see his/her query results on the browser.



Level 1 Data Flow Diagram represents the flow of data while the admin uses the administration tool. This Data Flow Diagram consists of 4 parts.

### 1.New Project Form

We said that admins can create new projects. After filling the related parts in the admin tool's interface, the information will be added to "Project" entity as a new row.

### 2.New Database Form

Admins can also add new database information to the system so that other databases can be used to retrieve data. This database information will be added to the database.

### 3.Adding Query/Result Page Templates

It will be possible for the admins to create new query/result pages and save it to the database. This speciality gives facility to change the user interface of the system without changing the actual project.

### 4.Process

By pressing the start button admins will be able to execute the whole system. Data will be exported to memory from specified database, and according the ip's in the database the FND Result Servers which are nodes will be ready to get requests and responses. Also the query and result pages' paths will be exported to memory so that they will not be read from database every time servers get a request. By pressing the stop button whole system will reset itself and FND Result Servers is not going to be able to response to the requests of the users.

## 4.2 Data Dictionary

<b>Name</b>	<b>User Output</b>
<b>Input To</b>	Admin Node
<b>Output From</b>	Computer of Admin
<b>Description</b>	Sends information to Admin Panel via Internet

<b>Name</b>	<b>Command</b>
<b>Input To</b>	Process
<b>Output From</b>	Admin Node
<b>Description</b>	Getting related command and interprets the output signal

<b>Name</b>	<b>Page Details</b>
<b>Input To</b>	Adding query/result page templates
<b>Output From</b>	Admin Node
<b>Description</b>	Sends information about templates

<b>Name</b>	<b>New database form</b>
<b>Input To</b>	System RDBMS
<b>Output From</b>	Admin Node
<b>Description</b>	Sends information about database such as type

<b>Name</b>	<b>New project form</b>
<b>Input To</b>	Adding New Project Node
<b>Output From</b>	Admin Node
<b>Description</b>	Sends information about projects

<b>Name</b>	<b>Signal(start/stop)</b>
<b>Input To</b>	FND result server node
<b>Output From</b>	Process
<b>Description</b>	Alarm to servers

<b>Name</b>	<b>Page infos</b>
<b>Input To</b>	Admin Tool(Databas-Pages)
<b>Output From</b>	Adding query/result page templates
<b>Description</b>	Sends Information about pages

<b>Name</b>	<b>Database Information</b>
<b>Input To</b>	Databases
<b>Output From</b>	System RDBMS

<b>Description</b>	RDBMS sends required information about Databases
--------------------	--

<b>Name</b>	<b>Project Info</b>
<b>Input To</b>	Admin Tool Databases – Projects
<b>Output From</b>	Adding New Project Node
<b>Description</b>	Sends projects information.

<b>Name</b>	<b>Recreation Database</b>
<b>Input To</b>	FND Result Server Node
<b>Output From</b>	Databases
<b>Description</b>	Creates new noSQL database and drag to in FND Result Server Node

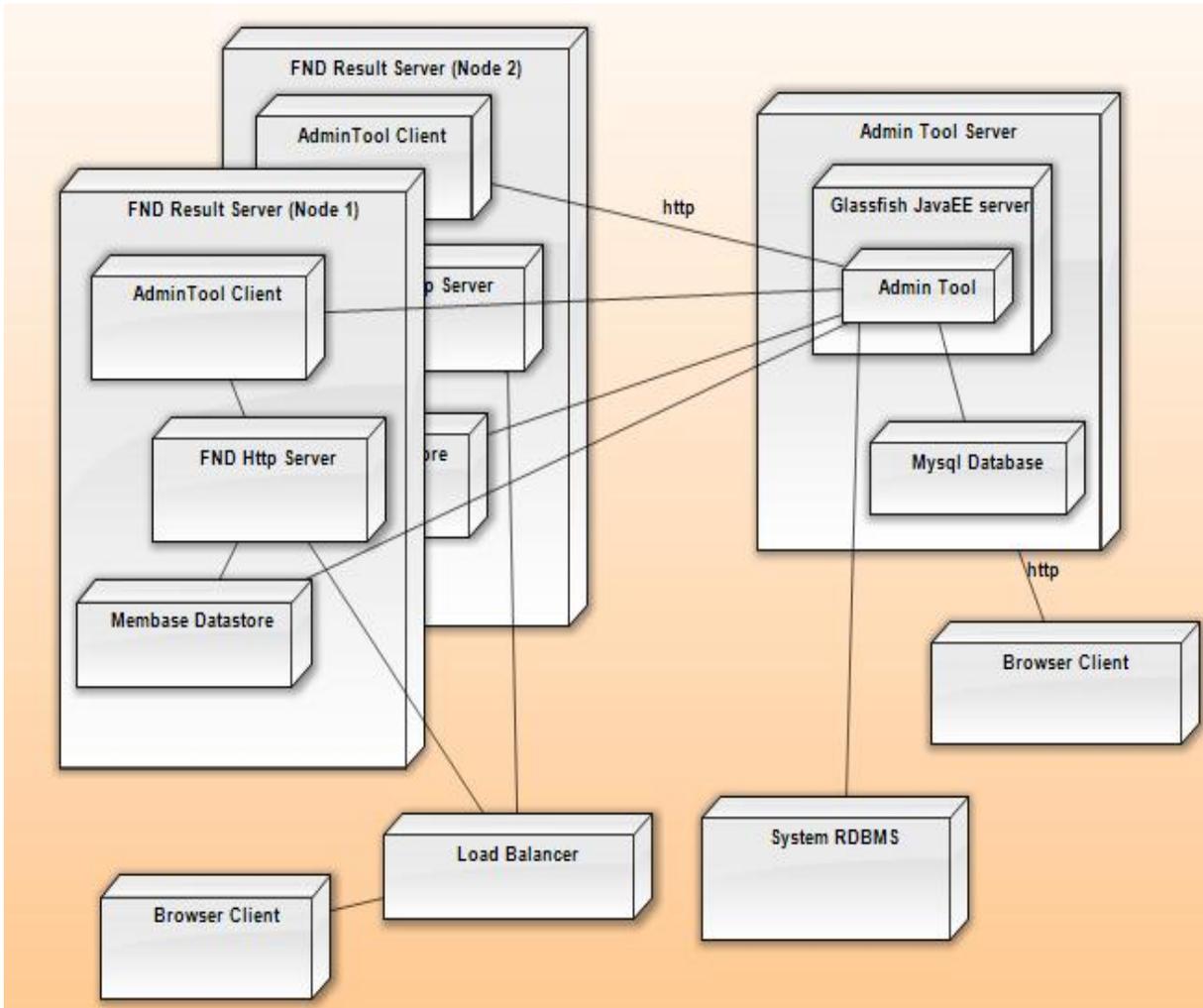
<b>Name</b>	<b>Recreation Pages</b>
<b>Input To</b>	FND Result Server Node
<b>Output From</b>	Admin Tool Database-Pages
<b>Description</b>	Creates new quick page templates in FND result server node

## 5. System Architecture

### 5.1 Architectural Design

Major components of the project are a result server which handles “Query Page” and “Result Page” traffic which works under a specialized Linux environment and an administration tool. The administration tool consists data translation which exports DBMS based data to a NoSQL Datastore System and templates of Result Page and Query Page.

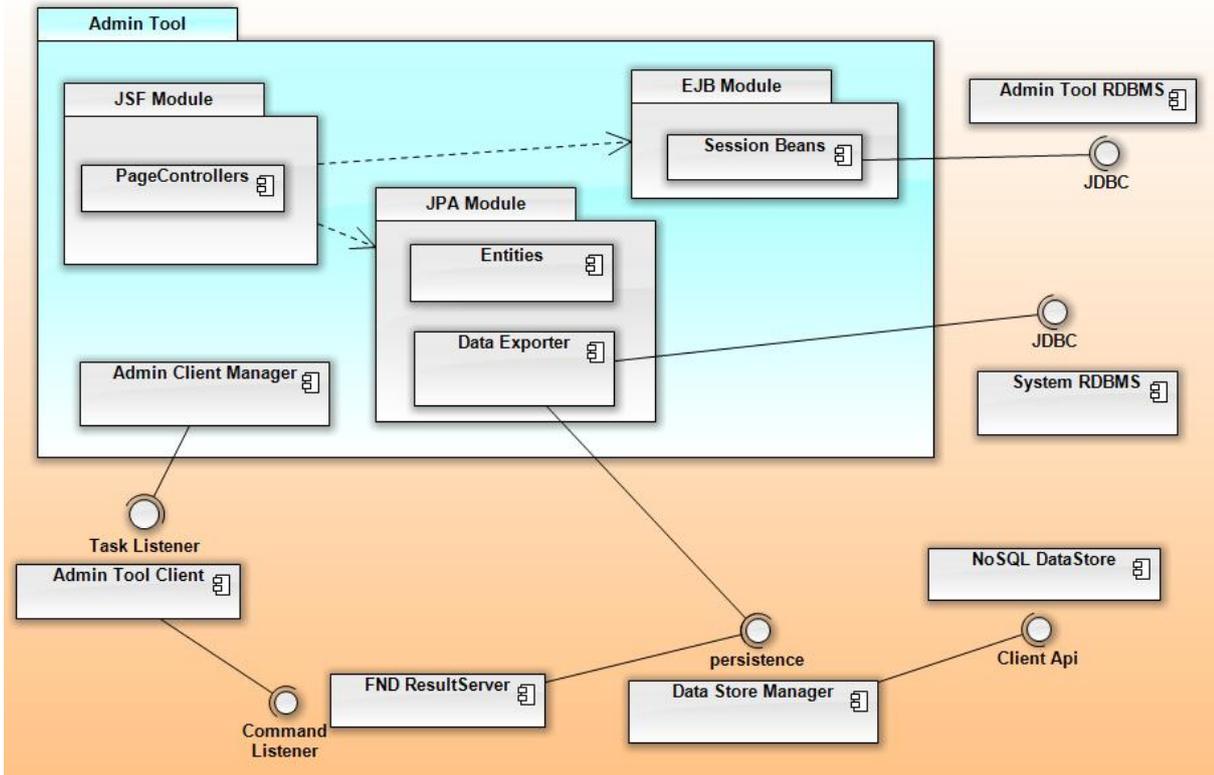
The diagram is for better understanding of the main idea of our product. All components, interconnections and relations are analyzed in detail in the next sections of this document.



## 5.2 Description of Components

Admin Tool is used for controlling the whole system. It provides to admins opening/closing the FND Result Servers, transporting the data from RDBMS to NoSQL Datastores, creating result and query page templates.

FND Result Server is supposed to handle excessive requests coming from end users. It works with NoSQL Datastores such as Membase which contains data retrieved from RDBMS.



### 5.2.1 Component “Admin Tool”

Admin Tool is going to be implemented by JSF, JPA, EJB. It is going to be used by admins only to create project and control this project.

#### 5.2.1.1 Processing narrative for component “Admin Tool”

Admin will use this tool to create a project firstly. After that admin will choose the required data from RDBMS which is also chosen by admin to transport data to NoSQL Datastore. Admin can give command to Admin Tool Client which lies on the FND Result Server to start or stop the desired server nodes.

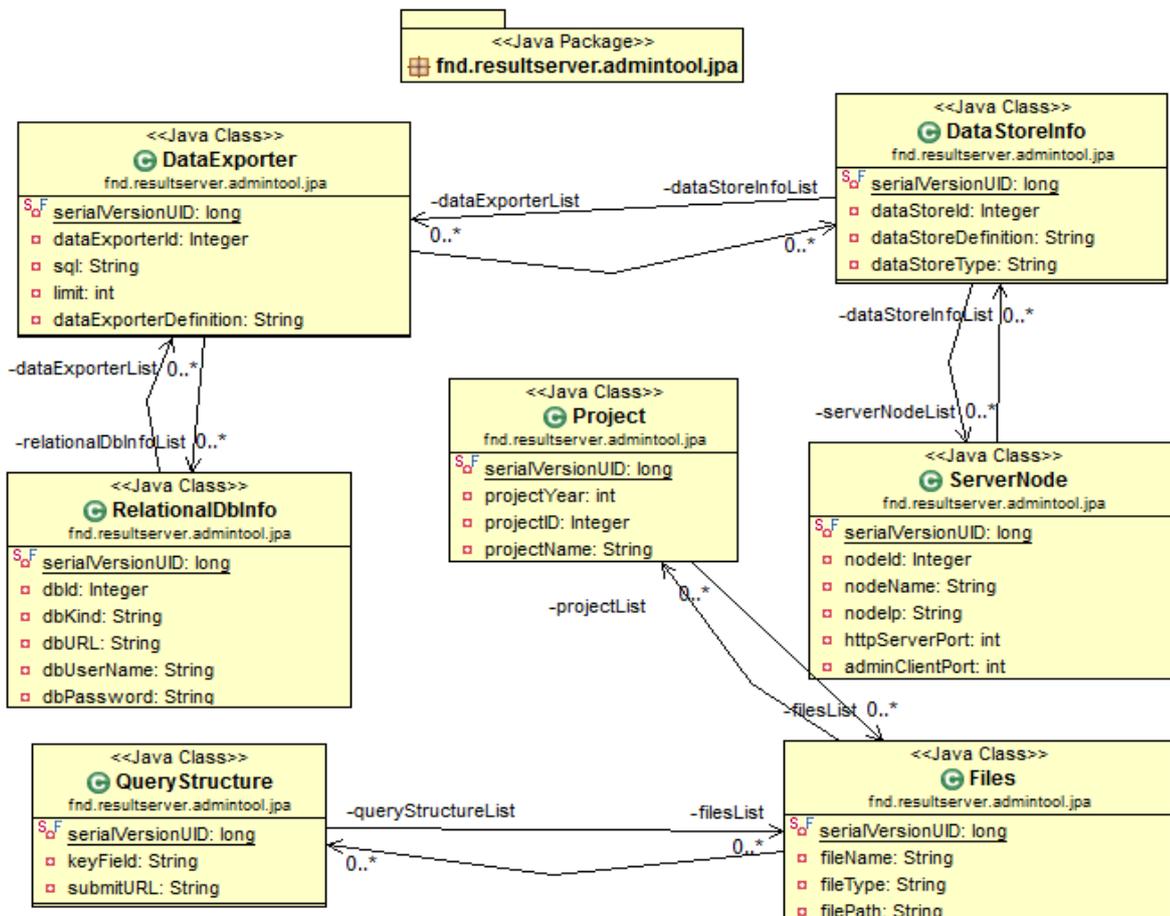
#### 5.2.1.2 Component “Admin Tool” interface description

Admin Tool will has simple, user friendly interface. It is going to be implemented by JSF which is a Java-based Web application framework intended to simplify

development integration of web-based user interfaces. Due to the concern of design issues we also want to use Prime Faces which is a lightweight open source component suite for Java Server Faces 2.0 featuring 100+ rich set of JSF components.

### 5.2.1.3 Component "Admin Tool" processing detail

As seen in the figure at the below Admin can transport data from RDBMS which represented as RelationalDBInfo to DataStoreInfo by using DataExporter. Admin can create project by using project class which is directly related with files which are used for creating query and result pages, represented as QueryStr



## 5.2.2 Component “FND Result Server”

FND Result Server is going to be implemented by Jboss Netty. It is only going to be controlled by Admin Tool. It is responsible to answer excessive number of request coming from end users of our application.

### 5.2.2.1 Processing narrative for component “FND Result Server”

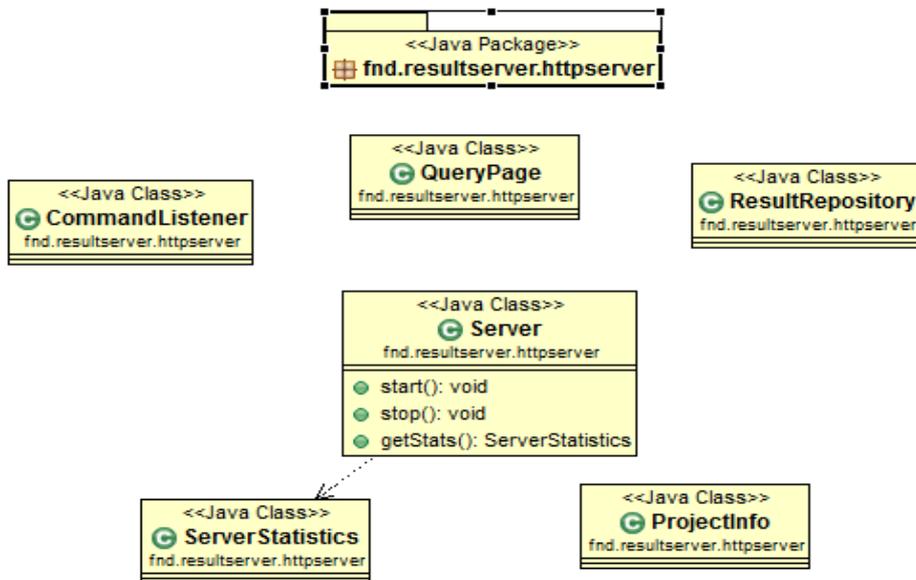
FND Result Server’s life will start with the command of Admin Tool : start. After being alive FND Result Server is ready to answer requests. Its main duty is respond this request in reasonable time. It retrieves requested data from NoSQL Datastore and send it to the demandant.

### 5.2.2.2 Component “FND Result Server” interface description

Since FND Result Server is going to be controlled by Admin Tool there is no interface to the users.

### 5.2.2.3 Component “FND Result Server” processing detail

As seen in the figure at the below, FND Result Server will get commands from Admin Tool by the help of CommandListener. It will retrieve data from NoSQL Datastore by the help of ResultRepository. It will also serve the result in the form of a web page whose object oriented corresponds is QueryPage.



## 6. User Interface Design

### 6.1 Overview of User Interface

There are two kind of users – Admins and end users. Admin kind users will encounter Admin Tool interfaces. There are various kind of operation that an admin can perform via Admin Tool. Therefore there are a lot of interface for admins. At below we serve some of them. On the other hand, for end users corresponding interfaces are going to be designed by admins via Admin Tool.

## 6.2 Screen Images

### ADMIN TOOL

- Create the project
- Define the database
- Define the result server nodes
- Export system data
- Specify Query / Result pages
- Upload templates
- Configure FND http servers
- Start / Monitor / Stop

### DEFINE RELATIONAL DATABASE

URL :

KIND :

ID :

USERNAME :

PASSWORD :

## Add New Project

Name :

ID :

Year :  -  -  at  /   

Ay      Gün      Yıl      Saat      Dakika

## EDIT RELATIONAL DATABASE

URL :

KIND :

ID :

USERNAME :

PASSWORD :

## 7. Detailed Design

This part contains the internal details of each design entity/component.

### 7.1 FND Result Server Node

**Classification:** Subsystem

**Definition:** This component is a subsystem of the whole system which includes the FND Http Server.

**Responsibilities:** The responsibility of this component is actually the most important part of whole project. It evaluates the request coming from clients then prepares a response to be sent back to client.

**Constraints:** The main constraint for this subsystem is that it can serve limited number of connections to clients because of the limitations on operating systems. They allow limited number of connections to be opened. Using TCP/IP tuning we are going to increase the number of connections.

**Composition:** FND Http Server is a part of this subsystem. Actual work is done through this FND Http Servers.

**Uses/Interactions:** This subsystem will be in interaction with another two subsystems "Admin Tool Client" and "Data Store Manager" through "Command Listener Interface" and "Persistence Interface".

**Resources:** As resources, this component will use memory to retrieve information and processor to evaluate the requests. Because of that all the information will be loaded to memory from database no database is going to be used directly by this subsystem.

**Processing:** The request which is got from load balancer is going to be evaluated in the FND Http Server. According to the request server is going to get data from memory through "Persistence Interface" and create necessary response and return it to the client. Also this component is going to be listening admin tool client to start, stop or monitor the server through "Command Listener Interface".

**Interface/Exports:** There will be no interfaces for this component because it is actually the processing part of the system. As it comes to its exports, it is going to create the http response which will be composed of Membase information, web page template and exports this response to be sent to client.

## 7.2 Admin Tool Client

**Classification:** Subsystem

**Definition:** This component is going to be used to start, stop and monitor the FND Result Servers.

**Responsibilities:** It will be responsible for starting, stopping and monitoring the FND Result Servers through the interface named "Command Listener Interface".

**Constraints:** The constraint it has that it will be only capable of starting, stopping and monitoring the servers. It will have no other authority on the system.

**Composition:** There is no other component which composes this subsystem.

**Uses/Interactions:** Admin Tool Client will be in interaction with "Admin Tool" subsystem's module "Admin Client Manager" and "FND Result Server".

**Resources:** What it is going to use as resource is only processor to catch the incoming commands from "Admin Client Manager".

**Processing:** The processing part is the part which the user can communicate with the "FND Result Servers". Admin Tool Client will be waiting for a command to start, stop or monitor the servers. When a specific command is received from "Admin Client

Manager” through interface “Task Listener”, Admin Tool Client will make a process accordingly. After the end of process, this component is going to be ready and will be waiting for a new command from “Admin Client Manager”.

**Interface/Exports:** This component is not going to have any interfaces. Its visuals are going to be included in “Admin Tool” and the communication will be achieved through “Admin Client Manager”. As an export, this component will end up with a result which will start, stop or monitor any or all of the “FND Result Servers”.

### 7.3 System RDBMS

**Classification:** Subsystem

**Definition:** The database where data will be retrieved from. It is owned by institution who wants to use “FND Result Server” system.

**Responsibilities:** This subsystem’s responsibility is to keep any information and provide opportunity to retrieve that information whenever it is required.

**Constraints:** The constraint of this subsystem is actually the reason which caused this project arise. It does not return the required data in a very reasonable time. It is fast actually but the speed is not enough to be used at some times.

**Composition:** This component will be composed of files which includes the necessary data to be retrieved.

**Uses/Interactions:** It’s use is simple. It just saves the information and gives it back when it is asked for it. It will be in interaction with “Data Exporter” package of “Admin Tool” s “JPA Module”.

**Resources:** What it consumes is processor to process the coming "SQL Query" and return answer for the call, memory and disk to save the information to be retrieved back later on.

**Processing:** When this component gets an "SQL Query" it will find the related information in the pre-saved files and prepare an sql result and return it to "Data Exporter" package.

**Interface/Exports:** This component is not going to have any interfaces. Its visuals are going to be included in "Admin Tool" and the communication will be achieved through "Data Exporter". The "Data Exporter" is going to use the JDBC Interface of Java. As an export, this component will end up with a result which is the information in response to "SQL Query".

## 7.4 NoSQL Data Store

**Classification:** Subsystem

**Definition:** This component will be Redis or Membase system to hold the necessary information.

**Responsibilities:** Its responsibility is to hold the information in the memory and return that information immediately when it is asked to do so.

**Constraints:** The main constraint on this component is memory space. Because memory will be limited, the whole database cannot be loaded in to memory system Redis or Membase.

**Composition:** The component will be composed of a "Redis System" or "Membase System".

**Uses/Interactions:** It will be in interaction with only "Data Store Manager" through a "client api". Its usage is only to keep data in memory so that it can be returned to client immediately.

**Resources:** As resource, it is going to use memory only.

**Processing:** There are three steps in this component's process.

1- On Admin's Start Command:

On this command, related data in "System RDBMS" will be loaded into memory system.

2- On Admin's Stop Command:

Related references in the memory system will be freed so that the space occupied by this project will be released and will be ready for the new project.

3- In "FND Result Server" 's run time:

The required information will be returned to server so that "FND Result Server" is going to prepare a http response and return it to client.

**Interface/Exports:** This component is not going to have any interfaces. Its visuals are going to be included in "Admin Tool" and the communication will be achieved through "Admin Client Manager". What it exports is just the information required by "FND Result Server" through usage of "Persistence Interface".

## 7.5 Data Store Manager

**Classification:** Subsystem

**Definition:** This is component that makes possible the communication between "FND Result Server", "Data Exporter" and "NoSQL DataStore".

**Responsibilities:** Its responsibility is to transmit the related "FND Result Server" request to "NoSQL DataStore" and also pass the data coming from "Data Exporter" to "NoSQL DataStore".

**Constraints:** There is no constraint for this component.

**Composition:** It will be composed of some Java Classes which will handle the specified operations above.

**Uses/Interactions:** This component will be in interaction with the "FND Result Server", "Data Exporter" and "NoSQL DataStore". It will be used to provide the communication between "NoSQL DataStore", "Data Exporter" and "FND Result Server".

**Resources:** As resource it is going to use memory and processor to process the request and keep the data to be passed to "NoSQL DataStore".

**Processing:** On Admin's start command the "Data Exporter" is going to make a connection to the specified "System RDBMS" and retrieve related data. This data will be transmitted to "Data Store Manager" through "Persistence Interface". This subsystem is going to understand that the data is coming from "Data Exporter" and this is a start operation. Then it will inform "NoSQL DataStore" about this and pass the data to it. Another possibility is that "FND Result Server" will request some data. Then this component is going to understand that the request is coming from "FND Result Server" and pass that request to "NoSQL DataStore". Also it will transmit the returned data from "NoSQL DataStore" to "FND Result Server" back.

**Interface/Exports:** This component has neither an interface nor an export. The its job is just to transmit the requests and responses between other components.

## 7.6 Admin Tool

**Classification:** Subsystem

**Definition:** This is the component which is responsible for getting the project details and making system to start or stop processing.

**Responsibilities:** Its responsibility is to control both the servers and relational databases.

**Constraints:** There is no constraint for this component.

**Composition:** It will be composed of below subsystems and the Java Classes which are handle the all control operations.

**Uses/Interactions:** : This component will be in interaction with the "FND Result Server", "Data Exporter", "Data Importer" and "NoSQL DataStore".

**Resources:** It is going to use memory and processor to execute and deploy related pages.

**Processing:** This process commands "FND Result Servers" by the gathering the wishes from admin user. When admin user chooses a task these tools sends a message related node. Moreover the admin can get relational database, changes types, and transform NoSql to servers.

**Interface/Exports:** This node is has an interface which is accessible from anywhere. It exports NoSQL queries to FND result servers.

### 7.6.1 JSF Module

**Classification:** Subsystem

**Definition:** JSF is a request-driven MVC web framework based on component driven UI design model, using XML files called view templates or Facelets views. Requests are processed by the FacesServlet, which loads the appropriate view template, builds a component tree, processes events, and renders the response (typically HTML) to the client

**Responsibilities:** Provides basic JSF templates and backing seam components to be able to display an interface that either use direct HTTP buffering or the RTSP-based URL that plays well with a server instance.

**Constraints:** It is needed that a suitable server such as glassfish.

**Composition:** FacesServlet, html pages , java classes

**Uses/Interactions:** This subsystem is always interacted with admin tool.

**Resources:** It is going to uses servers , remote machine and some memory for deployment.

**Processing:** The JSF life cycle has six phases as defined by the JSF specification:

**Restore View:** In this phase, the JSF implementation restores the objects and data structures that represent the view of the request. Of course, if this is the client's first visit to a page, the JSF implementation must create the view. When a JSF implementation creates and renders a JSF-enabled page, it creates UI objects for each view component. The components are stored in a component tree, and the state of the UI view is saved for subsequent requests. If this is a subsequent request, the previously saved UI view is retrieved for the processing of the current request.

- **Apply Request Values:** Any data that was sent as part of the request is passed to the appropriate UI objects that compose the view. Those objects update their state with the data values. Data can come from input fields in a web form, from cookies sent as part of the request, or from request headers. Data for some components, such as components that create HTML input fields, is validated at this time. Note that this does not yet update the business objects that compose the model. It updates only the UI components with the new data.
- **Process Validations:** The data that was submitted with the form is validated (if it was not validated in the previous phase). As with the previous phase, this does not yet update the business objects in the application. This is because if the JSF implementation began to update the business objects as data was validated, and a piece of data failed validation, the model would be partially updated and in an invalid state.
- **Update Model Values:** After all validations are complete, the business objects that make up the application are updated with the validated data from the request. In addition, if any of the data needs to be converted to a different format to update the model (for example, converting a String to a Date object), the conversion occurs in this phase. Conversion is needed when the data type of a property is not a String or a Java primitive.
- **Invoke Application:** During this phase, the action method of any command button or link that was activated is called. In addition, any events that were generated during previous phases and that have not yet been handled are passed to the web application so that it can complete any other processing of the request that is required.
- **Render Response:** The response UI components are rendered, and the response is sent to the client. The state of the UI components is saved so that the component tree

can be restored when the client sends another request. For a JSF-enabled application, the thread of execution for a request/response cycle can flow through each phase, in the order listed here.

**Interface/Exports:** This deploys a java server page.

### 7.6.2 JPA Module

**Classification:** Subsystem

**Definition:** The Java Persistence API deals with the way relational data is mapped to Java objects ("persistent entities"), the way that these objects are stored in a relational database so that they can be accessed at a later time, and the continued existence of an entity's state even after the application that uses it ends. In addition to simplifying the entity persistence model, the Java Persistence API standardizes object-relational mapping.

**Responsibilities:** The Java Persistence API replaces the persistence solution of EJB 2.0 CMP (Container Managed Persistence). The Java Persistence API specifies relational persistence (ORM: Object relational mapping) only for RDBMS (although providers exists who support other datastores). The Java Data Objects specification(s) provides relational persistence (ORM), as well as persistence to other types of datastores.

**Constraints:** There is no constraint.

**Composition:** Relational Database related java classes.

**Uses/Interactions:**

It has relation to Enterprise JavaBeans ,the Java Data Objects API ,the Service Data Object API **Resources:** It is going to uses some memory for instances of classes.

**Processing:** It procees simply like any java classes.

**Interface/Exports:** There is no interface or exports.

### 7.6.3 EJB Module

**Classification:** Subsystem

**Definition:** This Module uses simply Enterprise JavaBeans (EJB) technology which is the server-side component architecture for Java Platform, Enterprise Edition (Java EE). EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.

**Responsibilities:** An EJB container provider is responsible for implementing the runtime support for enterprise beans. He provides the tools that are required for the deployment of the beans. He must ensure that the development environment transaction is enabled, scalable, and secure.

**Constraints:** This module uses glassfish server and Java EE.

**Composition:** It will be composed of some Java Classes which will handle the specified operations above.

**Uses/Interactions:** This has always an interaction with JSF and JPA

**Resources:** As a resource it uses some memory and a server.

**Processing:** Admin creates a session bean instance using one of the create<Method> methods defined in the session bean's home interface. The session bean's home interface is provided by the bean developer; its implementation is generated by the deployment tools provided by the container provider. The container creates an instance of a session bean in three steps: Calls the bean class' newInstance method to create a new session bean instance. Calls the setSessionContext method to pass the context object to the instance. Calls the instance's ejbCreate<METHOD> method whose

signature matches the signature of the create<METHOD> method invoked by the client.

**Interface/Exports:** This has neither interface nor exports.

## 7.7 RDBMS for Admin Tool

**Classification:** Subsystem

**Definition:** A short definition of an RDBMS may be a DBMS in which data is stored in the form of tables and the relationship among the data is also stored in the form of tables.

**Responsibilities:** It has some responsibilities namely, creating databases, backing up and recovering databases, controlling security and access to the database and its objects, monitoring and tuning the performance of a database as well as the applications that access it, having table, index, and view design, multi-user considerations, networking, and loading, converting, and unloading data and ensuring data availability, accuracy, completeness, integrity, and consistency.

**Constraints:** Some entry limitations which are changing according to chosen database.

**Composition:** It consist of database tables.

**Uses/Interactions:** This module has an interaction with admin tool. Whenever admin tool want to change database to NoSQL , this provides related informations.

**Resources:** It needs some memory.

**Processing:** The RDMS just a container for data keeping module which is exist in servers of costumer.

**Interface/Exports:** This exports all data to Admin tool to change type and transform the FND Result Servers.

## 8. Libraries and Tools

### 8.1 JSF (Java Server Faces)

JavaServer Faces (JSF) is a Java-based Web application framework intended to simplify development integration of web-based user interfaces.

JSF is a request-driven MVC web framework based on component driven UI design model, using XML files called view templates or Facelets views. Requests are processed by the FacesServlet, which loads the appropriate view template, builds a component tree, processes events, and renders the response (typically HTML) to the client. The state of UI components (and some other objects) is saved at the end of each request (called stateSaving (note: transient true)), and restored upon next creation of that view. Several types of state-saving are available, including Client-side and Server-side state saving. Out of the box, JSF1.x uses JavaServer Pages (JSP) for its display technology, but can also accommodate other technologies (such as XUL and Facelets). JSF2 uses Facelets by default for this purpose. Facelets is a more efficient, simple, and yet more powerful view description language (VDL).

### 8.2 Netbeans

NetBeans refers to both a platform framework for Java desktop applications, and an integrated development environment (IDE) for developing with Java, JavaScript, PHP, Python, Ruby, Groovy, C, C++, Scala, Clojure, and others. The NetBeans IDE is written in Java and runs everywhere where a JVM is installed, including Windows, Mac

OS, Linux, and Solaris. A JDK is required for Java development functionality, but is not required for development in other programming languages. The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans platform (including the NetBeans IDE) can be extended by third party developers.

### 8.3 Jboss Netty

Netty is a NIO client server framework which enables quick and easy development of network applications such as protocol servers and clients. It greatly simplifies and streamlines network programming such as TCP and UDP socket server.

'Quick and easy' doesn't mean that a resulting application will suffer from a maintainability or a performance issue. Netty has been designed carefully with the experiences earned from the implementation of a lot of protocols such as FTP, SMTP, HTTP, and various binary and text-based legacy protocols. As a result, Netty has succeeded to find a way to achieve ease of development, performance, stability, and flexibility without a compromise.

### 8.4 Membase

Membase is a distributed key-value database management system best suited for applications that require predictable, low-latency, random access to data with high sustained throughput.

It is protocol-compatible with Memcached (both text and binary protocols), so if an application is already using Memcached, Membase can be dropped in without any change to application code or client configuration.

Membase is extremely easy to scale up, and down, making it attractive for applications in which data set size or data access volume can rapidly change. Membase scales linearly from a single server, to a cluster of hundreds of servers.

## 8.5 Java

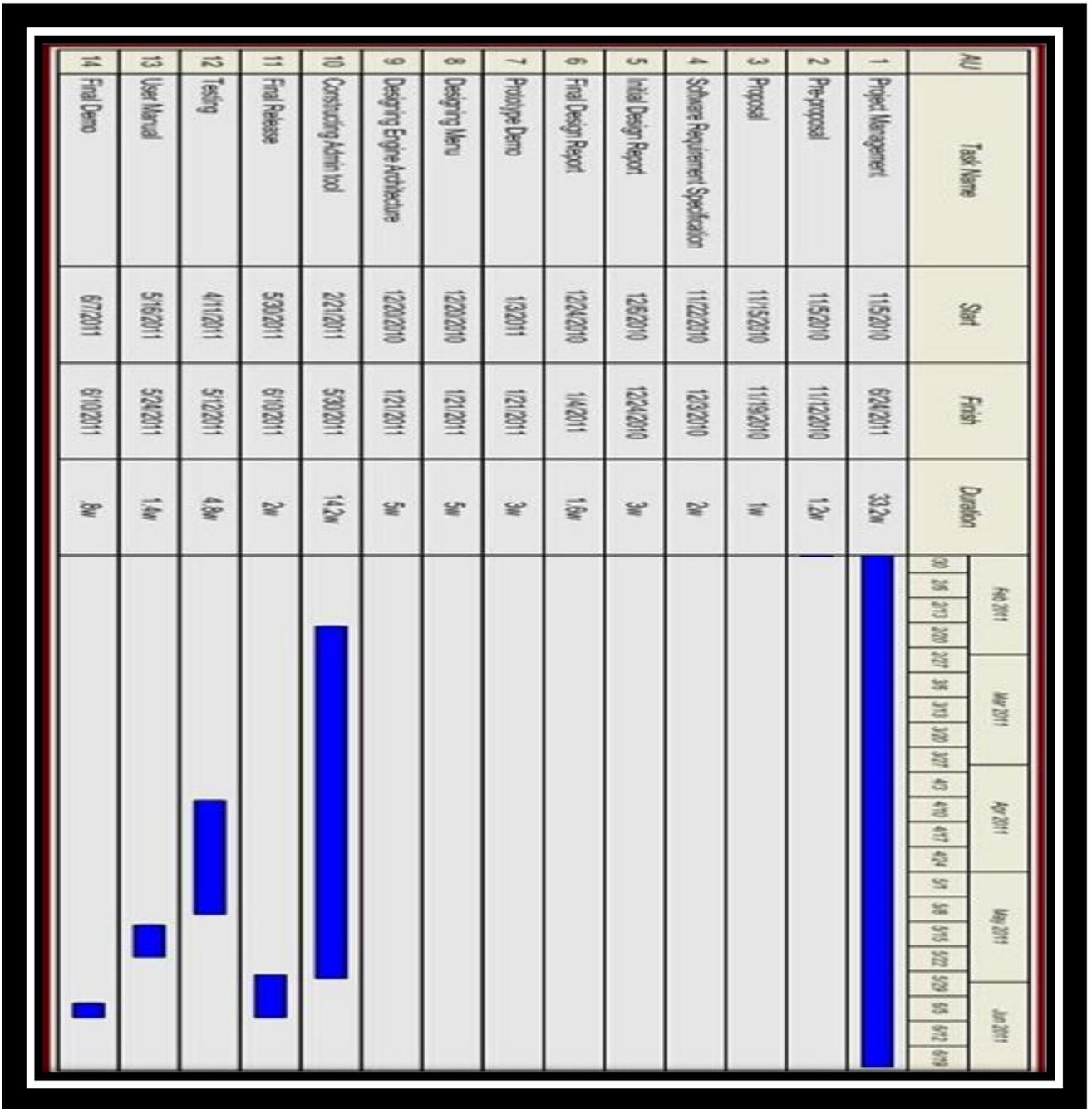
Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet.

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

- Write software on one platform and run it on virtually any other platform
- Create programs to run within a Web browser and Web services
- Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
- Combine applications or services using the Java language to create highly customized applications or services
- Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat



### 9.2 Term-2 Gantt Chart



## 10. Conclusion

The Initial Design Report for "FND Result Server" gives the definition, purpose and scope of the project. It is designed in detail that the "Admin Tool" and Server Nodes. The design decision will be used during developing the project.

It includes data flow models, class diagrams, entity relationship diagrams, possible use cases. It is explained the works that we have done so far and within the schedule we give the future work to be done.