SOFTWARE REQUIREMENTS SPECIFICATION

GTU BIZTALK

Version 0.1

Gebze Technical University
January 2019

Contents

1 Introduction

1.1 Purpose

The purpose of this specification document is to describe how are we going to handle BizTalk project . This document explains what features does the software will have, how the software will work, what kind of interfaces will it have, what would it require to work properly, which tools will it use and why did we preferred to use those tools. We tried to keep it as simple as possible so the users and the developers both can easily understand and participate with the software.

1.2 Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

1.3 Intended Audience and Reading Suggestions

- System administrators that administer the server
- Managers who assigns jobs and rules in company
- Employees who works for company
- Programmers who are interested in working on the project by further developing it or fix existing bugs.

1.4 Product Scope

BizTalk Server is an enterprise service bus (ESB) developed by Microsoft that can connect to various business servers that might otherwise be unable to interconnect or communicate. The BizTalk Server includes more than 25 multi-platform adapters as well as a stealth messaging infrastructure, allowing organizations to realize connectivity outside and inside their operations. It is best known for its superior integration functionality. The BizTalk Server is a form of business process management (BPM) solution. The "Biz" in BizTalk is short for business.

1.5 References

BizTalk references:
What is BizTalk
Security
Soap API
Davis Putnam Algorithm
Docker
Vert.X
Gradle
MySQL
Business Rule Engine
Postman

2 Overall Description

2.1 Product Perspective

The product is supposed to communicate with different computers used at the GTU. It is a web based system implementing client-server model. The GTU BizTalk provides mechanism for users to create Jobs, Rules and Orchestration.Simplify the planning and orchestration of jobs

2.2 Product Functions

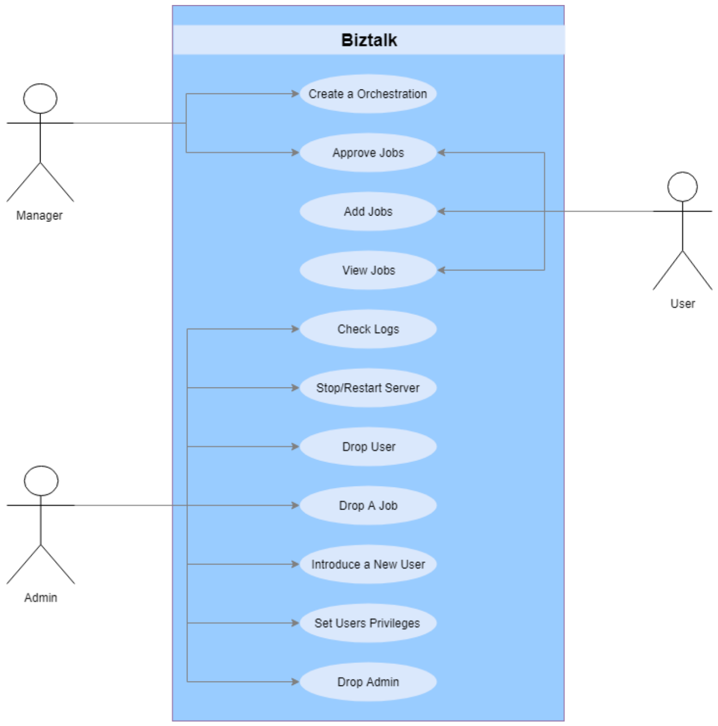


Figure 1: Use Case Diagram Of Product Functions

2.2.1 User

Add Job: The Add job section is the page that adds the user's job.

View Jobs: The job view page can display the jobs that users have added and information about these jobs.

Job Approval : In the Approve Jobs section of the User, the approvalTask

Table has the following properties: JobID, JobOwner, JobDescription, and Acceptance status.

2.2.2 Manager

Add Orchestration: The Orcestration is a job and can be defined as multiple job series and is a directed acyclic graph. Manager can add orcestration.

Add Job-Rule-End : In this section, the orcestration graph is created. Edges can be added between Job, Rule, End nodes and this node.

Manage Orchestration:All the information about these orchestrations in the database can be seen as owner, start job, date and status by the manage orchestration screen.

2.2.3 Admin

Add Manage Users: If you want to create a new user record, you can access the add users menu.

Manage Jobs: This page displays the jobs that users have added and the status in- formation for these jobs.

2.3 User Classes and Characteristics

- System administrators that administer the server
- Managers who assigns jobs and rules in company
- Employees who works for company

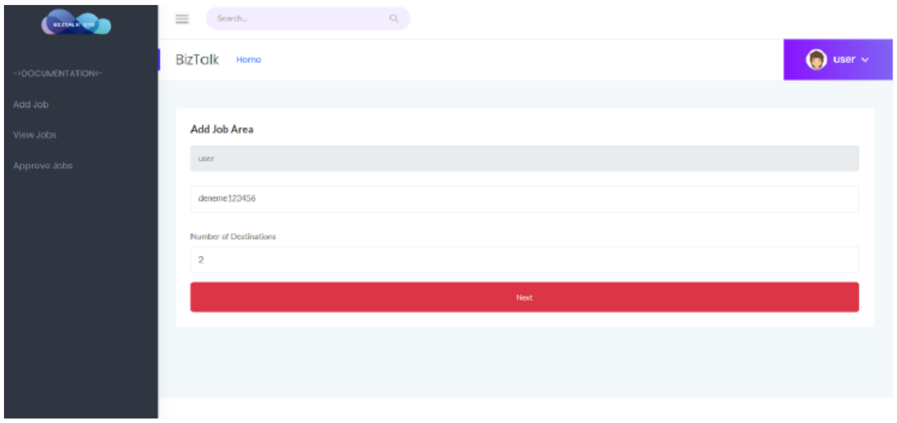
2.4 Operating Environment

- Windows
- Linux
- Mac OS

2.5 User Documentation

ADD JOB (User)

The Add job section is the page that adds the user's job. It consists of two pages in itself.



Add Job Area

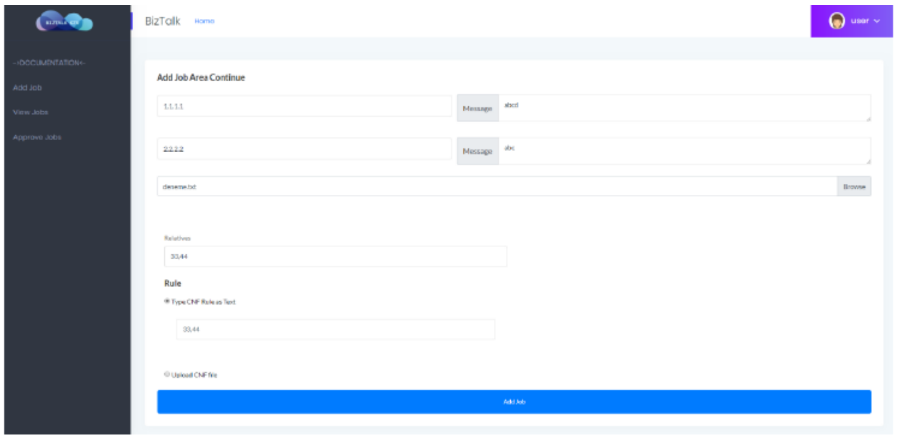
user

deneme123456

Number of Destinations

2

On the first page, the user enters the job's decription form and the number of destinations to which the file will be sent. Since the owner is himself, the owner has his own name written in the owner's section and cannot be modified. In this example, the number of destination is 2.User clicks on the next button after filling in these fields and the second screen comes.



Add Job Area Continue

1.1.1.1 Message abcd

2.2.2.2 Message abc

deneme.txt

Relatives

33,44

Rule

[✓ Type CNF Rule as Text]

33,44

Upload CNF file

In the second screen, the ips of the destinations and the messages to be sent with the file are entered. The number of ip and message text boxes is 2 because the number of 2 is entered on the first screen.

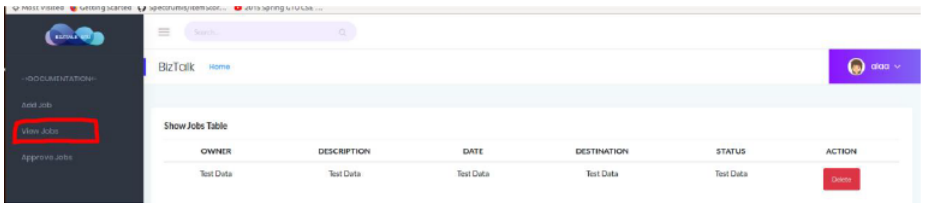
Then the upload part of the file to be sent comes. The file is selected in this section. The relays to be sent to the query will then be sent to this file. In the last part, the rule information is entered. Rule information can be entered either from text field or can be selected in file format. This selection is provided by radio buttons.

In addition, the rule must contain all relative values written in the Relatives text box. User clicks Add Job after filling all fields. The succes screen will appear if the job is added. After clicking the Add Job button in the background, the job information is saved in the tbljobs table in the phpmyadmin database.

The xml response is then populated with the information in the tbljobs table in the guinea database. This array is sent to the server with the addJobRule method using the soap structure. This section was made in addUsrJob3.php.

View Jobs(User)

The job view page can display the jobs that users have added and information about these jobs. If necessary, the job can be deleted. In this screen, a request is sent to the server side in xml format to display the job data, and the data in xml format that is reacted by the server is shown to the user as shown in the View Jobs screen.



Show Jobs Table

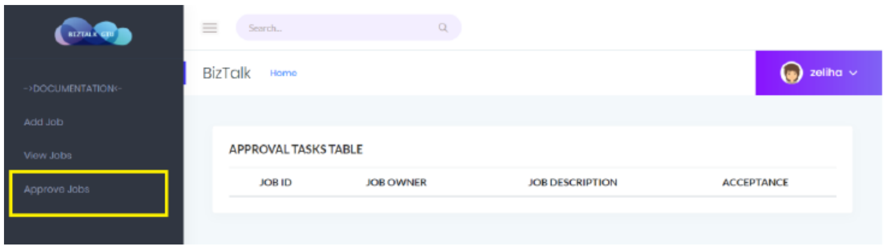
OWNER	DESCRIPTION	DATE	DESTINATION	STATUS	ACTION
Test Data	Test Data	Test Data	Test Data	Test Data	Delete

Owner: Represents user information.
Description: Represents the given description for the job during job addition.
Date: Represents the time that the job creation process was performed.
Destination: Represents the target information to which the file is to be sent by the created job. (To be sent to)
Status: Represents whether the files to be sent by Job are approved by the relevant users.
Action: If you want to delete the job, the job can be deleted with the Delete button in this section.

This page will appear blank if no job has been added.

JOB APPROVAL

In the Approve Jobs section of the User, the approvalTask Table has the following properties: JobID, JobOwner, JobDescription, and Acceptance status.



APPROVAL TASKS TABLE

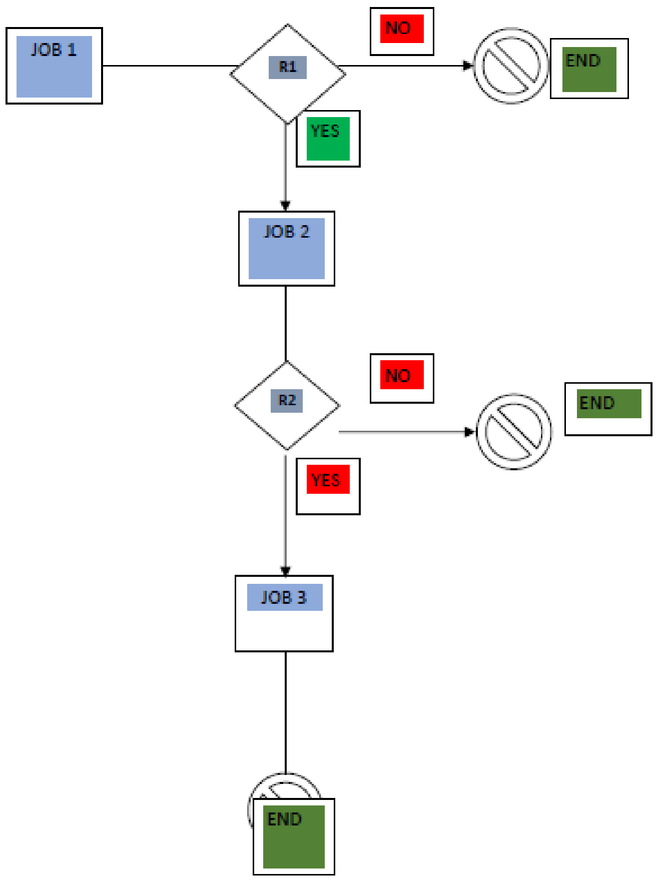
JOB ID	JOB OWNER	JOB DESCRIPTION	ACCEPTANCE

Relatives are submitted with the related rules after Job is created, the user submits the Job. Checks whether RuleEngine Rupees are correct, if the rolls are correct, the job has passed this step. The places to send the Job are the places on the destination ropes, and the people who approve Job are the relatives and not on the destination ip. When the job is submitted, the confirmationTask message is sent to the BizTalkServer relatives. The recipients confirm or cancel the job when they see this message. If the job is canceled, the BizTalkServer will not add the job to the database, and then the JobOwner will be notified whether the job has been approved.

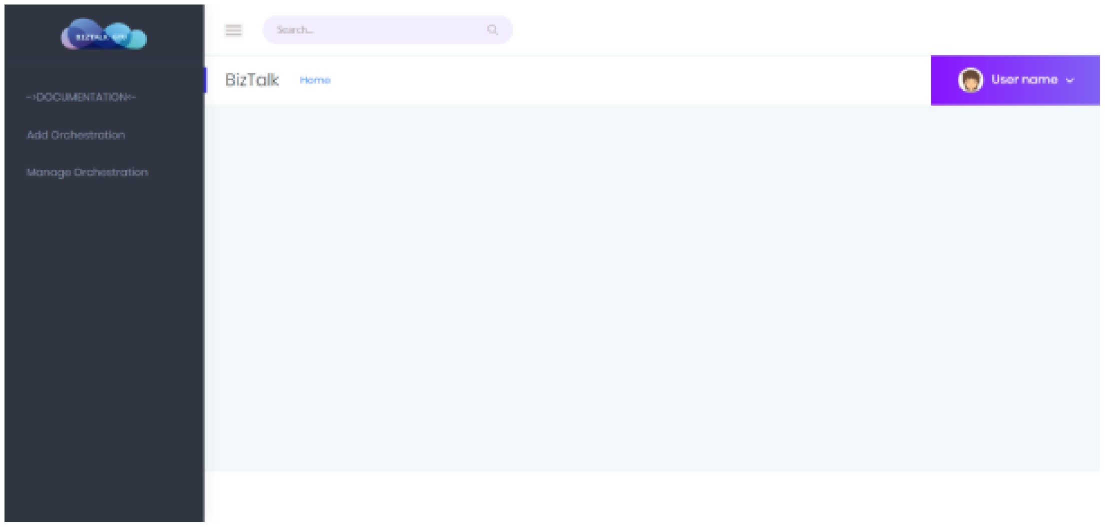
1) If Job does not approve Job's related relatives, job is disapproved and canceled.
2) If the admin cancels the job from JobBasket (drop), the job will not be approved. If not, the Rulees connected to the Job are also not approved and are canceled.

ADD ORCHESTRATION(MANAGER)

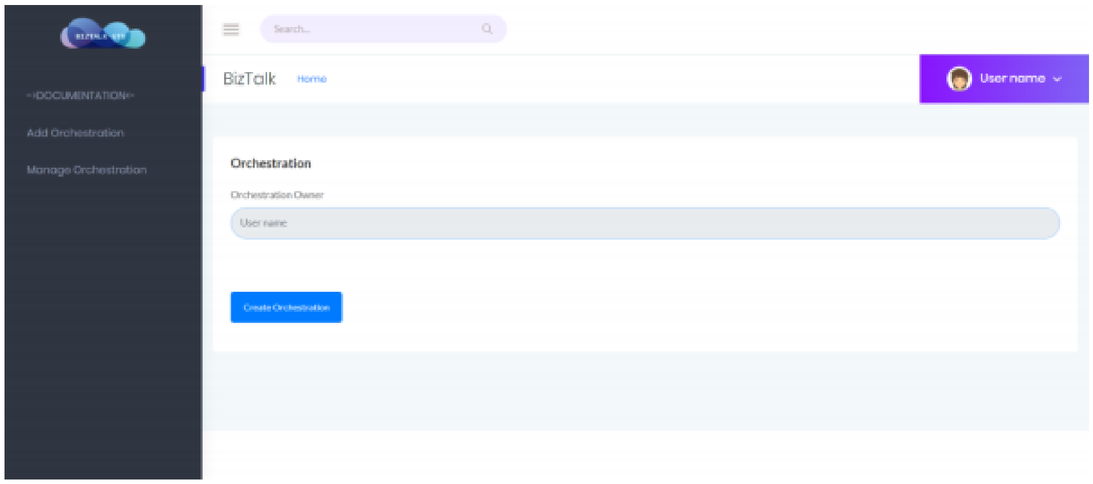
The Orcestration is a job and can be defined as multiple job series and is a directed acyclic graph. Manager can add orcestration. There are 3 nodes for Job, Rule and End, and 2 edge for Fragows and Leads. The orchestration starts with a starting job and this is followed by a job or rule starting job. The edge between the job and another job or rule is called follows. The edge between Rule and another job is called to lead. In addition, the success of the rule is successful when the added end node is added to determine whether to include yes and no leads. A direct edge can be created between the job and the job, but the rule cannot be created between the rule and the rule. According to the result from the Rule, the job can pass to the next job or stop.


JOB 1
R1
NO
END
YES
JOB 2
R2
NO
END
YES
JOB 3
END

HOME PAGE



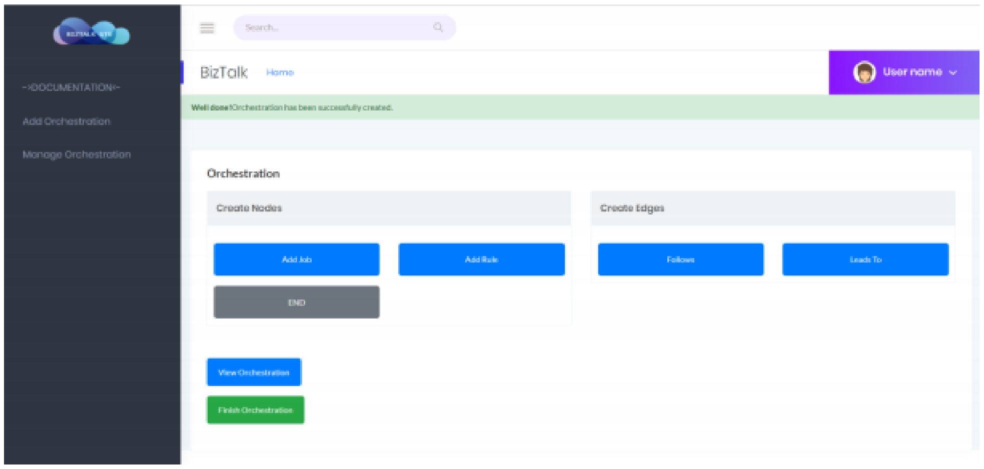
ADD ORCESTRATION



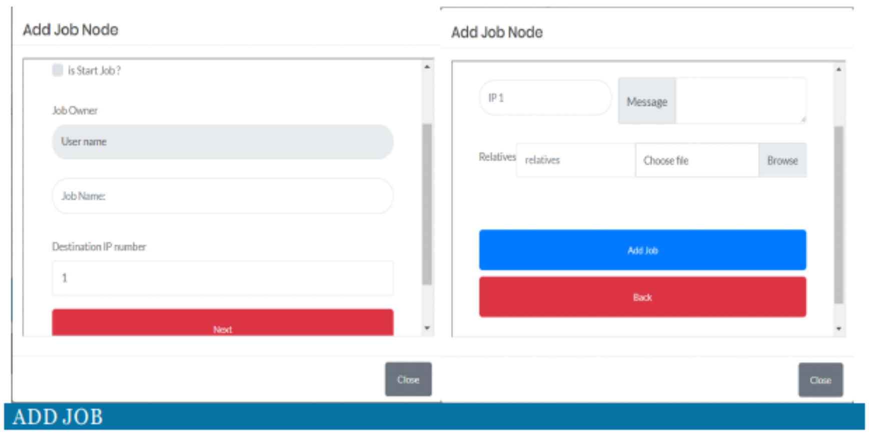
The Orchestration Owner is automatically typed for the logged in user. Orchestration is created with Create Orchestration. When the Orcestration process is completed (when the job rule end nodes are added), it is sent to the server. During the creation process, the information is kept in the gui database.

ADD JOB-RULE-END

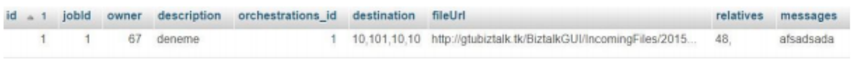
In this section, the orcestration graph is created. Edges can be added between Job, Rule, End nodes and this node. End node orchestration for finish noded. Orkestration can be viewed and deleted for any changes to the job rule, etc .. Unless the orchestration is created above, the page will continue even if it is refreshed.



Well done! Orchestration has been successfully created.



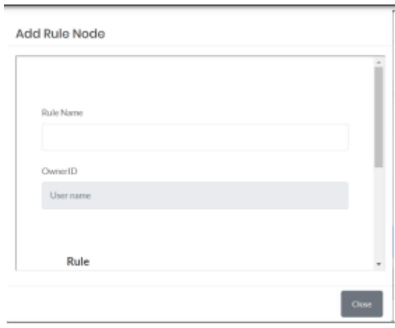
(Figure 3.1) (Figure 3.2)



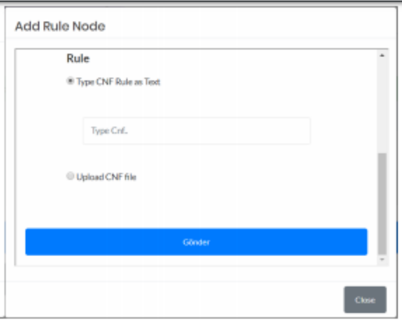
id	jobId	owner	description	orchestrations_id	destination	fileUrl	relatives	messages
1	1	67	deneme	1	10,101,10,10	http://gtubiztalk.tk/BiztalkGUI/IncomingFiles/2015...	48,	afsadsada

(Figure 3.3).

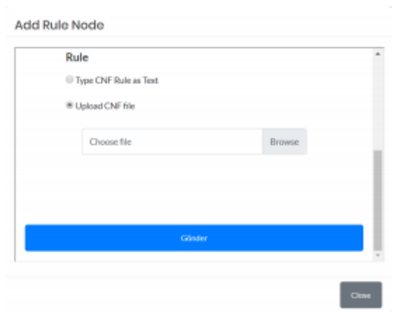
In this section job is added. The job name and number of target people are requested for the add job (Figure 3.1). A starting job must be set for each orchestration. Job is Start Job lard is marked if any of the job created is determined as the starting job. After this, the “is Start Job ecek option is disabled for any job to be added. For each orchestration, the job id and rule id 1 starts and continues. After this process, destionation ips, messages to be sent (these messages are optional) and relatives are requested (Figure 3.2). Relatives are selected from the dropdown list as input. Relatives are added to the database as user-id. And the last step is added to the desired file. When Job is added, Job is created. The information in these three steps (login user id, job-owner, job-name, job-id, destionation-ips, relatives-id, fileurl) is added to the orcJobs table (Figure 3.2). (Addorcjob3.php).



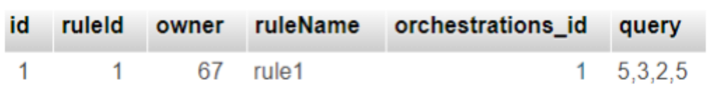
(Figure 3.3.1)



(Figure 3.3.2)



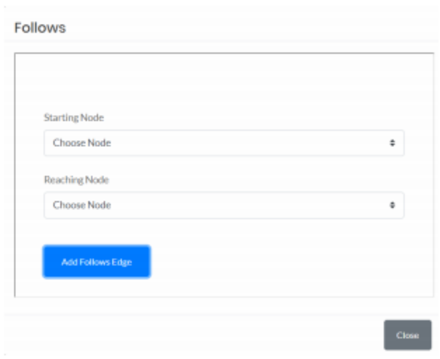
(Figure 3.3.3)



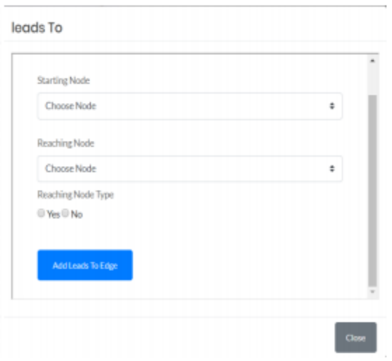
id	ruleId	owner	ruleName	orchestrations_id	query
1	1	67	rule1	1	5,3,2,5

(Figure 3.3.4) Next to the Job node, a rule can also be added. After the add rule, first you must specify how to enter the rule name (Figure 3.3.1) and cnf (Figure 3.3.2, Figure 3.3.3). Rule is created when you click Send. This information is added to the gui database. It is added to the orcRules table (Figure 3.3.4).

FOLLOWS



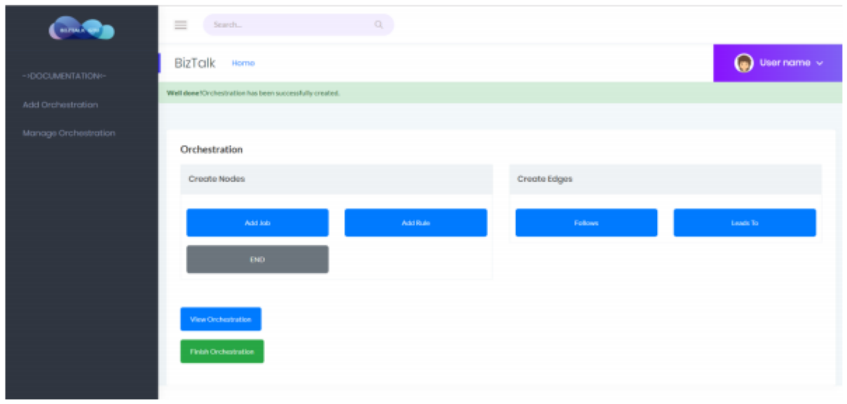
(Figure 3.5)



(Figure 3.6)

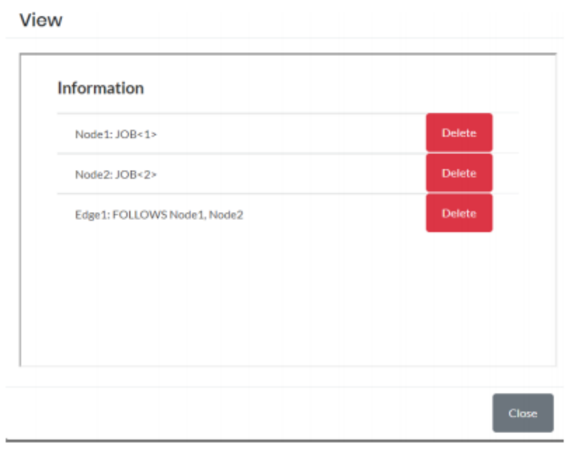
After the addition of Job and Rule, these nodes can be added to the edge. Follows can be between edge job and job, rule and end node. Dropdown lists lists pre-created jobs and routes. For Follows edge, the starting node and end node types and orchestration id information is retrieved. This information is added to the orcFollowsEdge table in the gui database. The leadsto edge takes place between the job and the end. When the rule is successful, orchestration is unsuccessful, or if there is no orchestration. Leads to initial node and end node types and orchestration id and yes no information is received. This information is added to the orcLeadstoEdge table in the gui database.

END-VIEW ORC-FINISH ORC-REQUESTS



Well done! Orchestration has been successfully created.

(Figure 4.0)



Node1: JOB<1>
Node2: JOB<2>
Edge1: FOLLOWS Node1, Node2

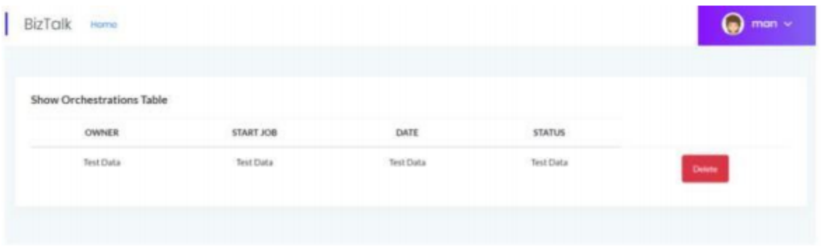
(Figure 4.1)

If you want to end the orchestration end button is added with the end node. The orc scheme created with ViewOrchestration is shown. It can be deleted if desired (Figure 4.1). The orchestration is completed with the FinishOrchestration process, and the information saved in this guideline is sent to the server. An xml-array is created for the send operation. The information generated during this array e orchestration is pulled from the gui database and added to xml-array.
$(xml - array['jobList'][\text{i}][\text{'owner'}] = rowJobs['owner'];$
....
...
)

The generated xml-array is sent to addOrchestration with the following structure (orc3.php) is fixed for all requests.

Manage Orchestration(Manager)

The orchestration created by Manage Orchestration (Manager) is kept in the database and all the information about these orchestrations in the database can be seen as owner, start job, date and status by the manage orchestration screen. With the delete button, the desired orchestration can be deleted.



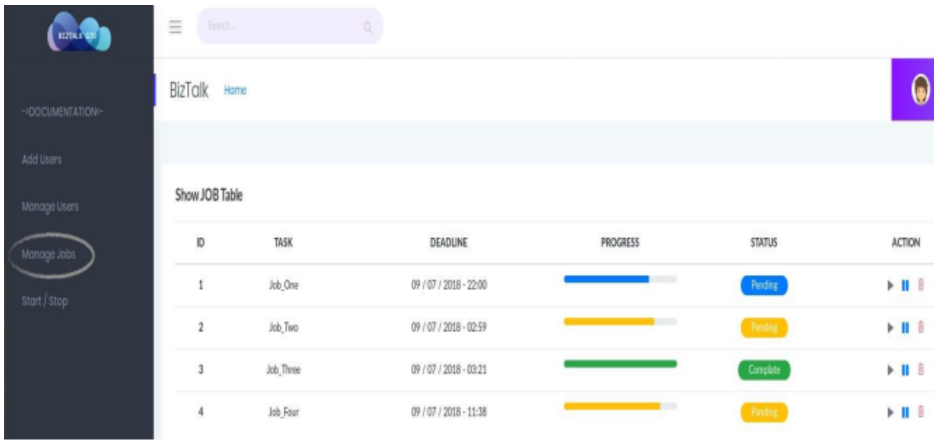
Show Orchestrations Table

OWNER	START JOB	DATE	STATUS	
Test Data	Test Data	Test Data	Test Data	Delete

Orchestrations are listed in the list; In order to manage the system, a xml request is received from the server and the return response from the server is parsed and then the orchestration is listed according to the results.

Manage Jobs(Admin)

Manage Jobs page displays the jobs that users have added and the status information for these jobs. The admin can check the status of these jobs. If the admin wants to do the job deletion can occur. Job statuses are sent to the server in xml format as request. The server displays the response paths according to the given format and is displayed on the Manage Jobs screen. Processes occur in the viewUserJobs.php file.

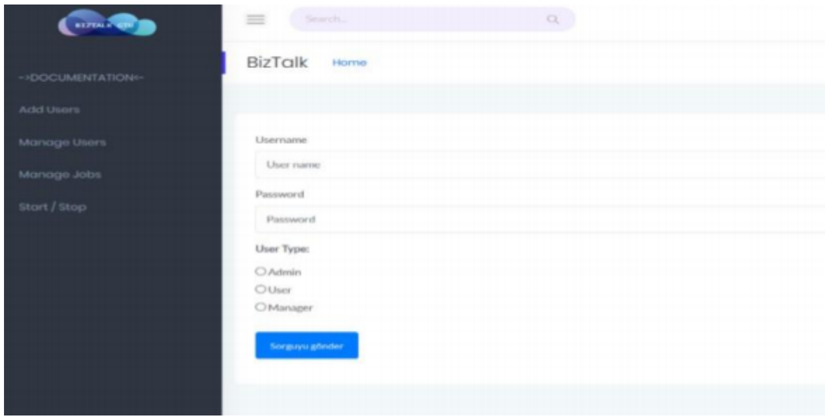


Show JOB Table

ID	TASK	DEADLINE	PROGRESS	STATUS	ACTION
1	Job_One	09 / 07 / 2018 - 22:00		Pending	
2	Job_Two	09 / 07 / 2018 - 02:59		Pending	
3	Job_Three	09 / 07 / 2018 - 03:21		Complete	
4	Job_Four	09 / 07 / 2018 - 11:38		Pending	

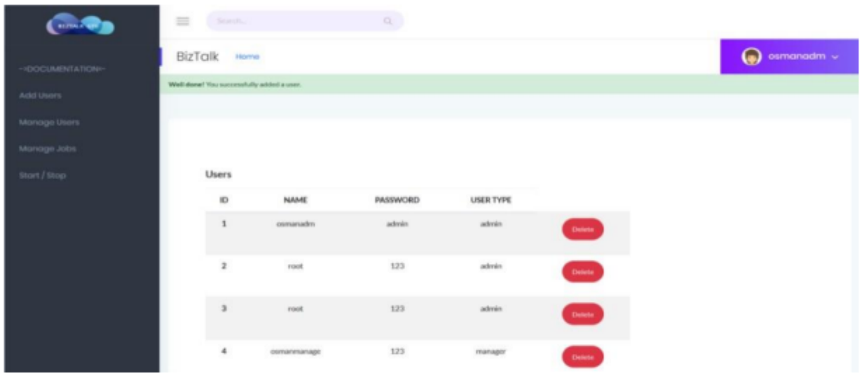
ID : Represents the ID of the created job.
TASK : Represents which job it is.
DEADLINE : Represents the timestamp of the generated job operation.
PROGRESS : Indicates the current graphical status of the generated job.
STATUS : Provides information on whether the job has been completed.
ACTION : The process of stopping, resuming or deleting the created job is done by admin.

Add – Manage Users(Admin)



Username
User name
Password
Password
User Type:
○ Admin
○ User
○ Manager

If you want to create a new user record, you can access the add users menu on the left. After providing the necessary username and password information in this section, a new record can be created by specifying the authorization class of the created user, and it can be displayed in the list in the manage users section. The list is listed in the order of registration.



Well done! You successfully added a user.

Users

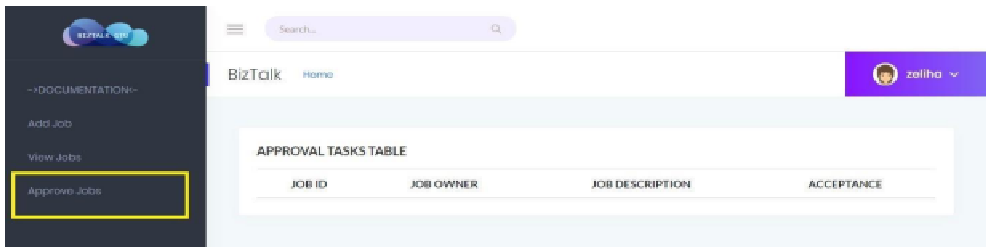
ID	NAME	PASSWORD	USER TYPE	
1	osmanadm	admin	admin	Delete
2	root	123	admin	Delete
3	root	123	admin	Delete
4	osmanmanage	123	manager	Delete

Only Admins are authorized to make changes to users. After adding a user, they can access the manage users section from the left menu. On this page you can list the details of existing users and delete them with delete button. The deletion takes place via deleteUser.php.

The user list is also kept in the database provided by phpmyadmin and the authentication is provided separately with the information from this database. The removed user registration is also deleted from the database. The information that is kept in this separate database is sent to the server via xml format determined via manageusers.php if required request is received from biztalk server.

JOB APPROVAL(User)

In the Approve Jobs section, the approvalTask Table has the following properties: JobID, JobOwner, JobDescription, and Acceptance status.



APPROVAL TASKS TABLE

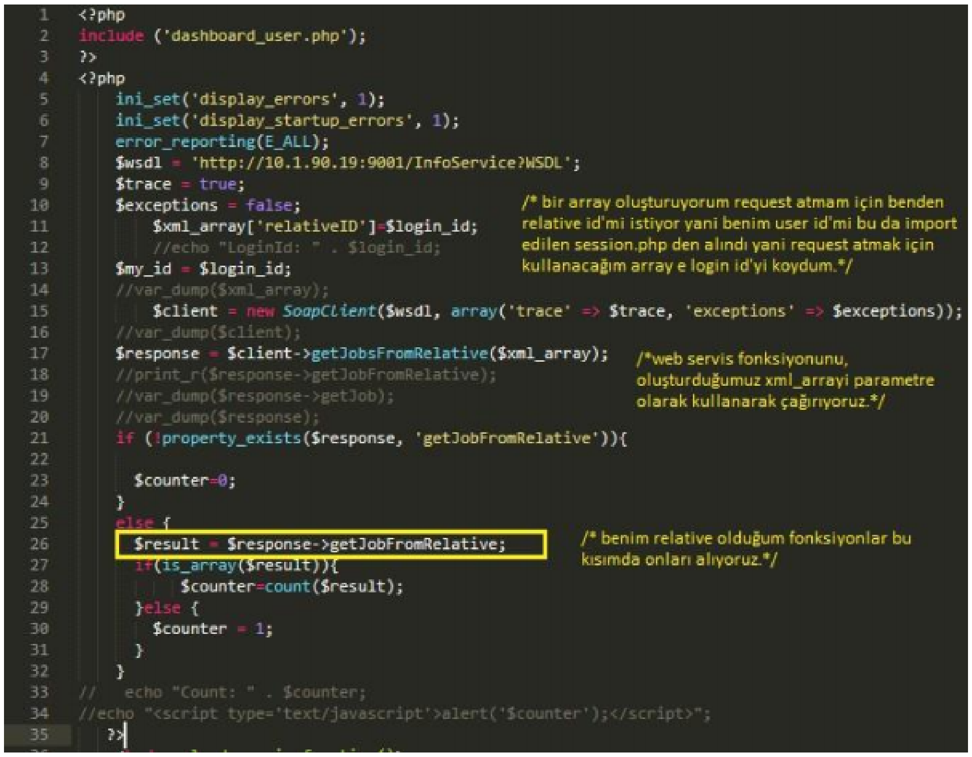
JOB ID	JOB OWNER	JOB DESCRIPTION	ACCEPTANCE

Relative s are submitted with the related rules after Job is created, the user submits the Job. RuleEngine checks whether the Rule is correct, if the rolls are correct, the job has passed this step. The places to send the Job are the places on the destination ropes, and the people who approve Job are the relatives and not on the destination ip. When the job is submitted, the confirmationTask message is sent to the BizTalkServer relatives. The recipients confirm or cancel the job when they see this message. If the job is canceled, the BizTalkServer will not add the job to the database, and then the JobOwner will be notified whether the job has been approved.

1) If Job does not approve Job's related relatives, job is disapproved and canceled.

2) If the admin cancels the job from JobBasket (drop), the job will not be approved.

If not, the Rulees connected to the Job are also not approved and are canceled.

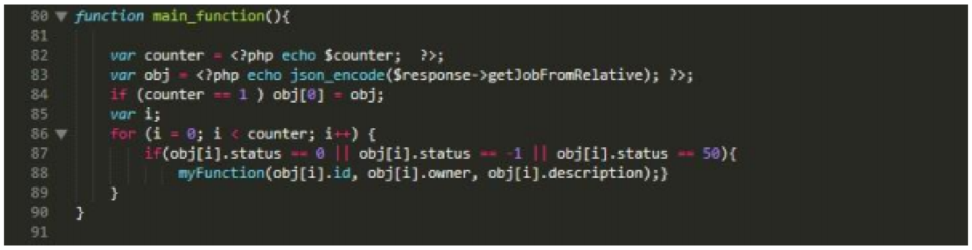


```php
<?php
include ('dashboard_user.php');
?>
<?php
    ini_set('display_errors', 1);
    ini_set('display_startup_errors', 1);
    error_reporting(E_ALL);
    $wsdl = 'http://10.1.90.19:9001/InfoService?WSDL';
    $trace = true;
    $exceptions = false;                        /* bir array oluşturuyorum request atmam için benden
        $xml_array['relativeID']=$login_id;     relative id'mi istiyor yani benim user id'mi bu da import
        //echo "LoginId: " . $login_id;         edilen session.php den alındı yani request atmak için
    $my_id = $login_id;                         kullanacağım array e login id'yi koydum.*/
    //var_dump($xml_array);
        $client = new SoapClient($wsdl, array('trace' => $trace, 'exceptions' => $exceptions));
    //var_dump($client);
    $response = $client->getJobsFromRelative($xml_array);   /*web servis fonksiyonunu,
    //print_r($response->getJobFromRelative);                oluşturduğumuz xml_arrayi parametre
    //var_dump($response->getJob);                           olarak kullanarak çağırıyoruz.*/
    //var_dump($response);
    if (!property_exists($response, 'getJobFromRelative')){

      $counter=0;
    }
    else {
      $result = $response->getJobFromRelative;         /* benim relative olduğum fonksiyonlar bu
      if(is_array($result)){                           kısımda onları alıyoruz.*/
          $counter=count($result);
      }else {
        $counter = 1;
      }
    }
//   echo "Count: " . $counter;
//echo "<script type='text/javascript'>alert('$counter');</script>";
   ?>
```

(FIGURE 1)

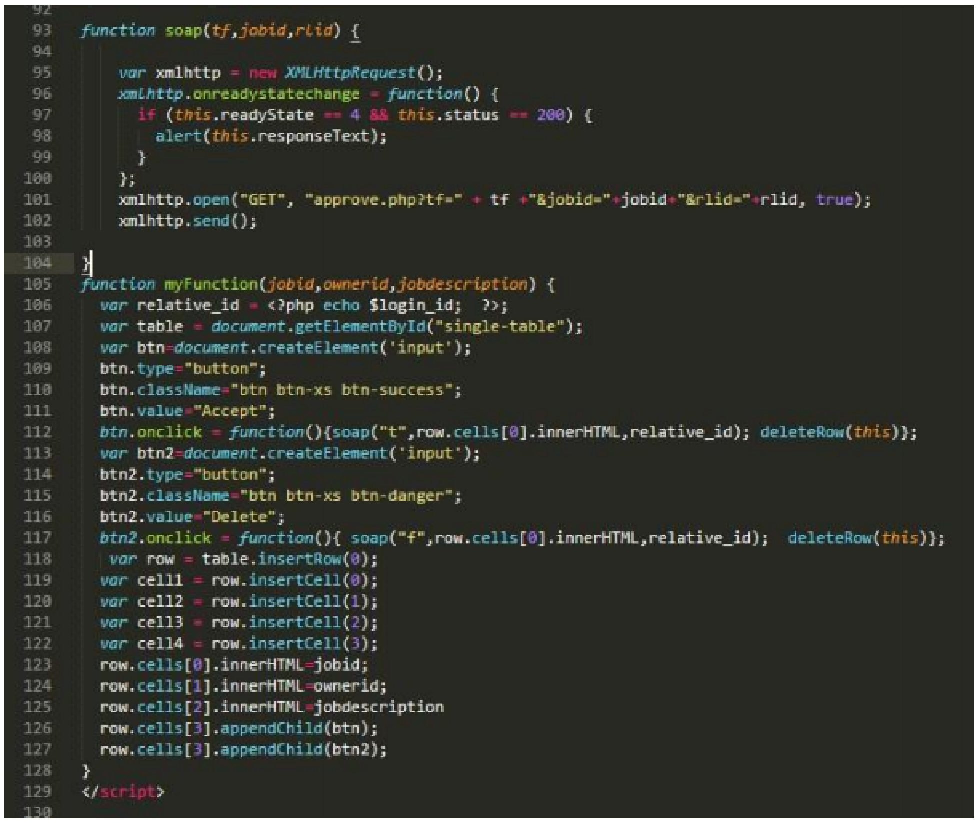
In Figure 2, we send the request from php in main function to our myFunction function, and press the screen, then add or delete the status of the user or the status of true or false according to the acceptance or rejection status of the job in the Approvel webservice according to the accept delete key on the screen. depending on the situation, we delete the job.



```javascript
function main_function(){

    var counter = <?php echo $counter;  ?>;
    var obj = <?php echo json_encode($response->getJobFromRelative); ?>;
    if (counter == 1 ) obj[0] = obj;
    var i;
    for (i = 0; i < counter; i++) {
        if(obj[i].status == 0 || obj[i].status == -1 || obj[i].status == 50){
            myFunction(obj[i].id, obj[i].owner, obj[i].description);}
    }
}
```

(FIGURE 2) The function myFunction is shown in Figure 3 below.



```
function soap(tf,jobid,rlid) {

    var xmlhttp = new XMLHttpRequest();
    xmlhttp.onreadystatechange = function() {
      if (this.readyState == 4 && this.status == 200) {
        alert(this.responseText);
      }
    };
    xmlhttp.open("GET", "approve.php?tf=" + tf +"&jobid="+jobid+"&rlid="+rlid, true);
    xmlhttp.send();

}
function myFunction(jobid,ownerid,jobdescription) {
  var relative_id = <?php echo $login_id;  ?>;
  var table = document.getElementById("single-table");
  var btn=document.createElement('input');
  btn.type="button";
  btn.className="btn btn-xs btn-success";
  btn.value="Accept";
  btn.onclick = function(){soap("t",row.cells[0].innerHTML,relative_id); deleteRow(this)};
  var btn2=document.createElement('input');
  btn2.type="button";
  btn2.className="btn btn-xs btn-danger";
  btn2.value="Delete";
  btn2.onclick = function(){ soap("f",row.cells[0].innerHTML,relative_id);  deleteRow(this)};
   var row = table.insertRow(0);
  var cell1 = row.insertCell(0);
  var cell2 = row.insertCell(1);
  var cell3 = row.insertCell(2);
  var cell4 = row.insertCell(3);
  row.cells[0].innerHTML=jobid;
  row.cells[1].innerHTML=ownerid;
  row.cells[2].innerHTML=jobdescription
  row.cells[3].appendChild(btn);
  row.cells[3].appendChild(btn2);
}
</script>
```

(FIGURE 3)

2.6 Assumptions and Dependencies

2.6.1 Product

Docker must be installed in the system where the product is installed.There is no other need because of the containerization we did.

2.6.2 Development

Java JDK v1.8 or upper,Vert.X v3.6.2 or upper ,Gradle v4.10,MySQL v5.7,Docker v18.09,FileZilla Server version 0.9.60 beta must be installed.

3 External Interface Requirements

3.1 User Interfaces

- Php
- Javascript
- PhpMyAdmin (SQL)

3.2 Hardware Interfaces

A machine with:

- 256 MB of RAM, although more than 512MB is recommended
- 10 GB of drive space (for your Docker image)

3.3 Software Interfaces

The following software installed:

- Java 8 (either a JRE or Java Development Kit (JDK) is fine)
- Docker (navigate to Get Docker at the top of the website to access the Docker download that's suitable for your platform)
- (BRE)Vert.X (Eclipse Vert.x is event driven and non blocking. This means your app can handle a lot of concurrency using a small number of kernel threads. Vert.x lets your app scale with minimal hardware.)
- (BRE)Gradle (an open-source build automation tool that is designed to be flexible enough to build almost any type of software.)
- MySQL (BRE, Server, Log)
- (BRE)MySQL (For storing rules and approvals from users)
- (LOG) MySQL (Used in order to store required details and query datas of a log)

3.4 Communications Interfaces

Rest and SOAP architecture were used together. Server-GUI communication is by sending XML over the SOAP architecture. Server-Log communication is by sending XML through Rest architecture. GUI-Log communication is by sending XML through Rest architecture. Server-BRE communication is by sending XML over the SOAP architecture.

4 System Features

This section demonstrates most prominent features of interface. This interface designed to support four different components used by GTU BizTalk culture. Which are Business Rule Engine, Log, GUI, Send Message File / Receive Message File.

All these components communicate with each other through web services. There are two types of web services; REST and SOAP.

4.1 Business Rule Engine

In summary Business Rule Engine checks and solves the given rules according to users approvals. Handles to http request with thread per request. Business Rule Engine communicates with server trough XML file.First XML file has 3 lines in it:

ruleId: ID of current rule
relatives: List of relatives for the job
rule: Rule in CNF format

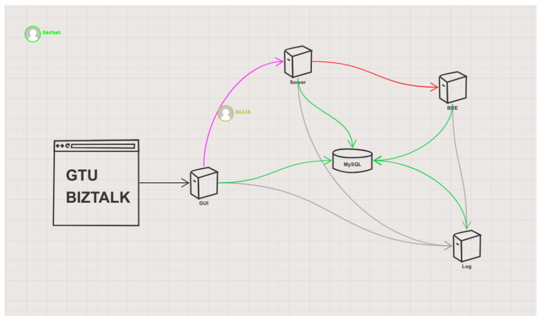


Figure 2: BizTalk Structure

XML data transforms to java strings(Rule and job relatives).Checks whether or not given Rule is in right syntax for CNF.Checks whether the literals in the rule are in the relatives, and whether the literals in the relatives are in the rule.Checks if CNF is tautology and return true immediately, in order to increase performance. Checks whether or not CNF is satisfy able. Waits for approval from job relatives.Approval is send by server with XML file. Lines in this file:

relativeID: ID of job relative
response: Response of the job relative

Solves the Rule with given Approvals from job relatives.Returns T if rule is satisfied with given approvals.Returns F if rule is unsatisfied with given approvals.Returns X if the rule cannot be resolved because there is not enough participation from job relatives(participation means approvals).Return values send to the server through XML file .Format of this file:

ruleID: ID of processed rule
status: Status of rule

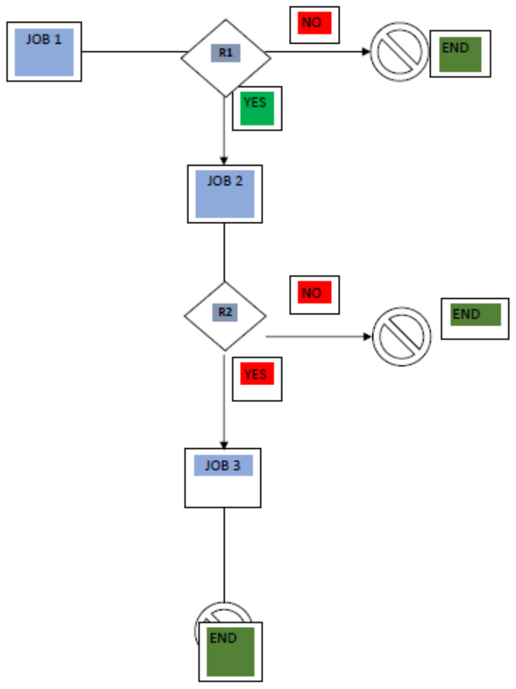
4.2 Server Features

Server system can be explained basically in two main sections: Database and Web Service. Web Service examined in three sections: Info Service, Orchestration Service, Approve Service. Info Service, transmits some information like JobId, OrchId, Job when needed. Orchestration Service, JubList and RuleList information is received and the orchestration is created background. Approve Service, rules that relative enters sent to Business Rule Engine part and processed the job by answer. At database section there is three table which created with with entered information from GUI: Orchestration Table, Rule Table, Job Table.

4.3 GUI Features

System administrators, Managers and Employees can use these features:

- Add Job
- View Jobs
- Job Approval
- Add Orchestration
- Add Job-Rule-End
- Manage Orchestration
- Add Manage Users
- Manage Jobs



4.4 Log Features

The system stores the logs to developers track events by their details. All the actions that occurs has been logged. These actions are :

- Authentication
- Creating job
- Creating orchestration
- Creating a rule
- Binding a rule to a job
- Approving or disapproving a rule by relatives
- Update a rule
- Shutdown or restart the server due to some reason
- Adding or dropping users by admin
- Promote or demote a user status by admin
- Drop a job from job basket by admin

inserted to database.

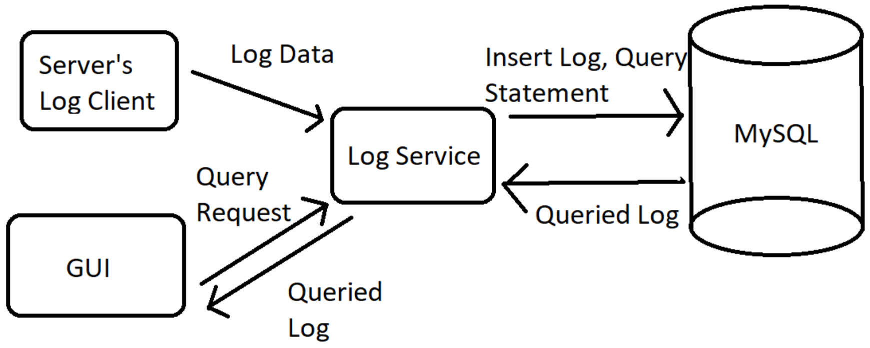


Figure 3: Log Structure

5 Other Nonfunctional Requirements

5.1 Performance Requirements

Recommended software and hardware requirements are described in sections 3.2 and 3.3.

5.2 Safety Requirements

All groups' database stores last state of processing data and database has back-ups. Every rule is checked by Business Rule Engine to be linguistically correct.

5.3 Security Requirements

This section provides information about the BizTalk security features, and how you can use them to help secure your data and environment.Authenticating the sender of a message. BizTalk can authenticate the sender of a message in order to validate the identity of the sender of the message Authorizing of the receiver of a message. After BizTalk receives the message, BizTalk can determine what processes and users have permissions to receive the message. BizTalk uses access control to ensure that BizTalk processes have appropriate limits and that access to business critical information is controlled. In other words,BizTalk Server ensures that users and accounts have the least user rights possible to enable them to do their tasks.

5.4 Software Quality Attributes

Firstly, this project is designed to satisfy the requirements stated in the story chart. A clear documentation is added for each section of this project to make the addition of new story charts more easier. All sections are also designed to handle invalid inputs. User interface is highly simple and easy to use.

5.5 Business Rules

Only Java code supported.

6 Glossary

- **BizTalk :** Is an ESB that enables data communication between different applications with different platforms.
- **Orchestration :** : Is the executable code to run a workflow supporting a business process, interacts with outside entities by using the ports. All these things together are a powerful tool to perform process automation.
- **Receive Port :** Listening messages.
- **Tool :** An item or implement used for a specific purpose. A tool can be a physical object such as mechanical tools including saws and hammers or a technical object such as a web authoring tool or software program.
- **Message Box :** Internal XML messages published and consumed by users.
- **Send Port :** Consumes internal XML messages, converts them to wire messages by the connected pipeline and assembler.
- **Job :** Is the desired job.
- **Rule :** Is the conditions for the realization of the work
- **Rule engine :** Checks whether the rules are correct.
- **Relatives :** : Users who approve jobs
- **Manager :** Creates orchestrations.
- **Admin :** Adds users and authorizes to them.