Assignment 3.2 documentation

Péter Zaváczki
30433

December 11, 2018

0.1 Requirements

Design, implement and test a distributed system that uses web services to expose the server functionalities to its clients.

0.2 Functional requirements

Consider a distributed application called "Online Tracking System" that has a GUI which exposes the following functionalities to its users:

- The application has two types of users: administrators and clients.
- After the login, the user is redirected to its corresponding page.
- If the user does not have an account, it can register and become a simple user (client)
- The Administrator can:
  - Add/remove package. The package has the following characteristics:
    - Sender – Client
    - Receiver – Client
    - Name
    - Description
    - Sender City
    - Destination City
    - Tracking – Boolean – initially false
  - Register package for tracking
    - The package becomes tracked, and a route is associated to it. This route represents the path of the package to the destination, as pairs of (City, Time).
  - Package status updating
    - A new entry (City, Time) is introduced to the route
- Each time new information about a DVD is introduced in the system the application must create automatically a text file and write the information about the DVD in it
- The Client can:
  - List all its packages
  - Search packages
  - Package status checking

These functionalities will be exposed as 2 web services:

- WS1 – SOAP Web Service: Client Login and Register and Simple Client Operations
- WS2 – SOAP Web Service: Administrator Operations

0.3 Implementation technologies

- Develop one SOAP Web Service in .NET and the other one in JAVA
- The GUI can be either WEB or Desktop and can be developed in .NET or JAVA

0.4 Conceptual architecture

This application has a complex architecture. It has two Client-Server components. One of them is the webapp, in this case the Client being the user's web browser, to which the webpages are loaded and the server is the Tomcat server to which the WAR files are deployed. The other one is the SOAP based webservice deployed on a GlassFish server. The Simple Object Access Protocol (SOAP) uses a HTTP based protocol to model an RPC style communication. Using this is advantageous, since it allows programs that run on different OSes to communicate using the above mentioned HTTP and its XML based WSDL. At one endpoint we have an implementor of the webservice who publishes it, and at another endpoint, its client, who uses it by the stubs generated based on the published WSDL abstraction.

This can be seen on figure 1.

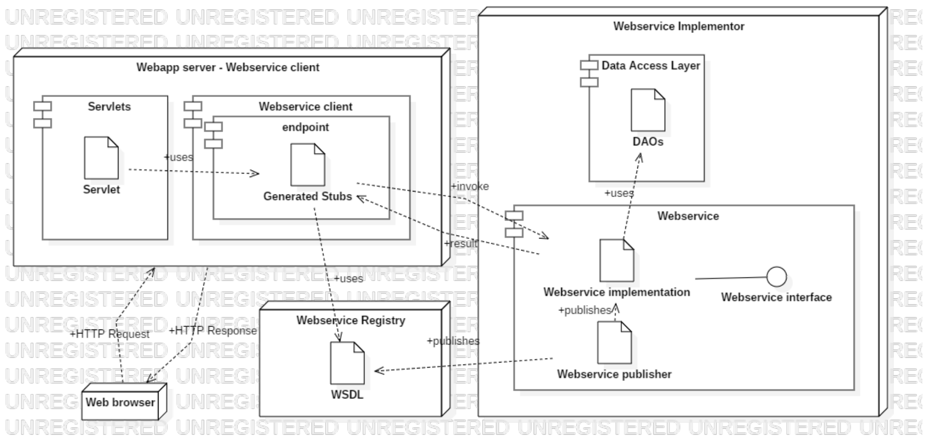

Figure 1: The conceptual architecture diagram of the project

0.5 Deployment

The application is deployed on multiple nodes: The webapp used for the UI is deployed on a Tomcat server, which has the Web Application Resource (WAR) files. The webservice, used for SOAP based communication is deployed on a Glassfish server, and it contains information about the service in form of a WSDL file.

This deployment can be seen on figure 2.

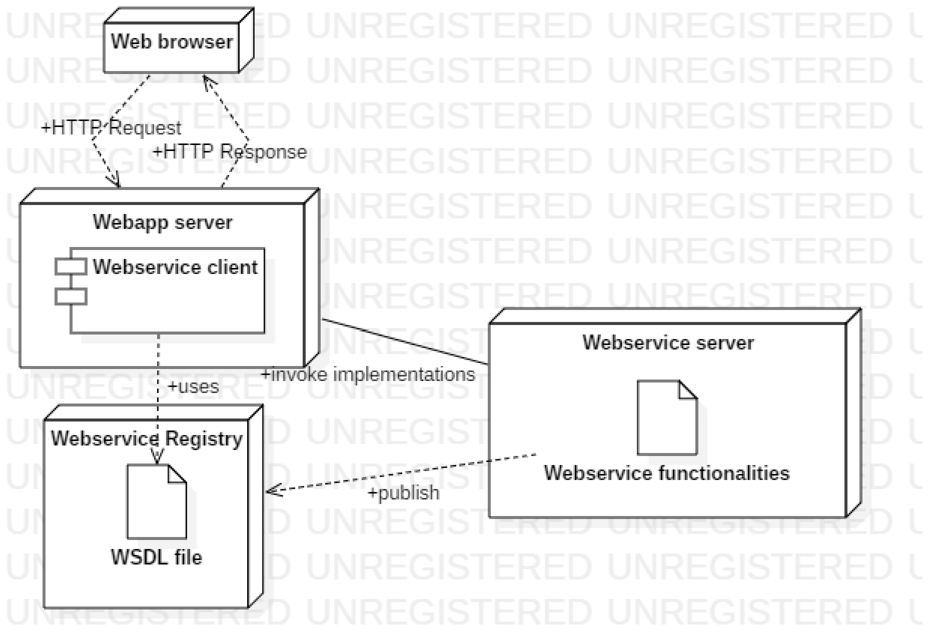

Figure 2: The deployment diagram of the project

0.6 Build considerations

- A web based UI was used for simplicity of development.