**Deployment diagram**

The deployment diagram highlights the physical deployment of artifacts on nodes. In our case we preferred to split the deployment diagram in two main views. One concerning the actual physical connections among nodes and the other concerning the logical end to end connection among artifacts inside nodes. Only the most relevant software artifacts have been inserted in the deployment diagrams.

As in every Client/Server architectural style there is the tier partitioning through clients and servers and for each tier there is a correspondent logical layer. The tier partitioning is visible in the physical view and the layer one in the logical view, instead.

**Deployment diagram – Physical View**

The physical view contains a fundamental element which is the Cisco Router. This device is connected to a modem for signals modulation and demodulation using the PPP (Point to Point Protocol). Indeed, the router is connected to a network backbone. The router acts as firewall for filtering both front-end network flow and intranet access. In addition, it has NAT (Network Address Translation) in order to address requests to the right server inside the intranet. Furthermore, the router has DHCP (Dynamic Host Configuration Protocol) for assigning IP addresses to servers inside the intranet dynamically.

As stated before the Proxy Server is reachable from web distributed clients directly due to his exposition on the Internet.

The presentation tier is composed by all clients, but only Operator Clients belong to the intranet. Car Clients and Special Parking Area Clients belong to the extranet (VPN). The router takes care of network partitioning exploiting subnet separation.

Clients may communicate to either the Web Server or the Application Server, directly. it being understood that the Web Server may exploit some of the Application Server functionalities communicating to it internally. All these connections are handled through the router.

The Application Tier is connected directly to the Data Tier because it is the only user of the data services. They are connected to plain Ethernet cable.

All communications rely on the TCP/IP protocols set.

**Deployment diagram – Applications end to end View**

Each node is characterized by its execution environment.

The Proxy Server runs a reverse proxy container. There are several reasons for installing reverse proxy servers:

* Encryption / SSL acceleration: a reverse proxy is equipped with SSL acceleration hardware.
* Load balancing: the reverse proxy distributes the load to several web servers, each Web Server or Application Serve.
* Serve/cache static content: a reverse proxy can offload the web servers and application servers by caching static content like pictures and other static graphical content.
* Compression: the proxy server can optimize and compress the content to speed up the load time.
* Security: the proxy server is an additional layer of defense and can protect against some Web Server or Application Server specific attacks.
* Extranet Publishing: a reverse proxy server facing the Internet can be used to communicate to a firewall server internal to an organization, providing extranet access to some functions while keeping the servers behind the firewalls.

The Application Server adopts GlassFish execution environment: this is the JEE engine that is going to run the application logic, which is going to be implemented in terms of Enterprise Java Beans. GlassFish is stable, well-known and open source. These are the main reasons for us to adopt it.

The Proxy Server and the Web Server adopt Apache TomCat: contrary to GlassFish, which provides a full implementation of the JEE framework, TomCat is specifically designed to run JSP (JavaServer Pages). For this reason, it can be used to run the server-side of applications, and it's a good idea to install it along with GlassFish.

The Database Server runs the well-known Oracle DBMS which is robust and reliable. In addition, Oracle DBMS supports and Entity/Relationship model schemas and therefore SQL which fits with our system needs.

A Web client runs on Computer Clients and consists of two parts: (1) dynamic web pages containing various types of markup language (HTML, XML, and so on), which are generated by web components running in the web tier, and (2) a web browser, which renders the pages received from the server. Web clients usually do not query databases, execute complex business rules. Such heavyweight operations are off-loaded to enterprise beans executing on the Java EE server, where they can leverage the security, speed, services, and reliability of Java EE server-side technologies.

An Application Client runs on Operator Clients, Smartphone Clients, Special Parking Area Clients and Car Clients. It provides a way for users to handle tasks that require a richer user interface than can be provided by a markup language. It typically has a graphical user interface (GUI). Application clients directly access enterprise beans running in the business tier. However, if application requirements warrant it, an application client can open an HTTP connection to establish communication with a JSP running in the web tier.

All these software containers are physical distributed and for this reason it is necessary to specify application communication protocols.

Communications between Application Clients and Application Servers occur via SOAP (Simple Object Access Protocol). SOAP can work with different operating systems and because of this, we can fulfill the portability nonfunctional requirement. Applications can run on Android, iOS, Windows Phone, Linux, Mac OS X and Windows. Basically, SOAP relies on HTTP and XML to exchange data through the web.

Communication between Proxy Server, Web Server and Application Server can be made using AJP (Apache JServ Protocol) which comes as a software module inside Apache TomCat execution environment.

JDBC (Java Database Connectivity) is an API (Application Programming Interface). It provides methods to query and update data in a database, and is oriented towards relational databases.