|  |
| --- |
| [[https://myetudes.org/etudes-melete-tool/images/printer.png](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385331) Send to Printer](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385331) | [Close Window](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385331) |
|  | |
| 20. Web Application Design Considerations  20.1. 3-Tier Architecture  *Copyright (c) 2014, Rula Khayrallah*  In module 15 we introduced a high level view of the web application as a client server architecture.  Web applications on the server side are usually structured in three tiers.  The **presentation tier** usually consists of an HTTP server which listens on an HTTP port and accepts requests from the outside world.  We have written our own server in node that accepts requests and serves static HTML documents.  In a more complex web application the HTTP server forwards requests to the **logic (or application) tier**.  The logic tier contains our actual application code.  It may generate dynamic content based on user requests or credentials.  It may also collect and update information from the user or it may even support financial transactions.  To access the application data, the logic tier relies on the persistence tier.  With complex web applications, we also need an application server.  The application server handles the details of HTTP requests for our logic tier and allows us to **define a high level routing to be associated with each request**.  For example, it can route incoming HTTP requests directly to the appropriate components in our application based on the request verb (GET requests vs POST requests).  Because an application server sits between the presentation tier and our actual application code, it is referred to as **middleware**.  Connect and Express are two middleware frameworks available for node.  The **persistence tier** is usually implemented using a database management system such as MySQL.  **The three tiers may actually reside on different computers. It’s even common for each tier to span several computers.**      It’s important to note that the lines between the three tiers are sometimes blurred.  For example we can implement a web application using Apache Tomcat in conjunction with Java servlets.  Tomcat is both a web server and servlet container:  it listens for http requests and as needed, creates and initializes a servlet to service a given request.  Whereas the java servlets belong to the logic tier, tomcat itself also belongs to the presentation tier.  20.2. The Web Stack  *Copyright (c) 2014, Rula Khayrallah*  When we develop a web application, we are faced with choices at all levels.  What web server do we use?  Some servers (such as the one we implemented in node) support event driven, asynchronous handling of requests.  Others, such as Apache web servers, create a new process for each request.  What framework/programming language is best suited for our application (on the server side)?  What about the persistence tier?  We commonly refer to a combination of choices as a web stack.  In making our choices, we need to take into account several factors:  security considerations, anticipated load, scalability, ease of development (based on our expertise), complexity of the application, maintainability and so on.   It's also helpful to use a well-tested solution.  **LAMP** is one of the earliest and most commonly used web stacks.  Its name refers to its original components:   * Linux for the operating system. * Apache for the HTTP server. * MySQL for the database management system. * PHP ( or Perl or Python) for the programming language used in the application code.   The four components of LAMP are free and open-source software.  The LAMP stack is used to power thousands of sites today:  it offers a well-tested powerful and reliable solution that is also easy to implement.    It is worth noting that as the industry moves towards JavaScript based backends, the following stack is gaining popularity:  Node.js + Express.js + Mongoose or MongoDb (for the persistence layer).  20.3. Scalability  *Copyright (c) 2014, Rula Khayrallah*  It’s important for a web application to be able to handle growth.  One approach for building scalability into web applications is to **allow each tier to span several computers.**  In this case we have several web servers handling http requests and routing them to one of several application servers.  We can then add computers to each tier to match the growing demand.  This is known as **horizontal scaling**.  Load balancers (which could be dedicated software or hardware) are responsible for distributing the work load evenly.  For this approach to work it is important that a component does not need to share anything with its counterpart on a different computer so there is no bottleneck involved around that shared resource.  This is referred to as the **shared-nothing architecture**.  The fact **that HTTP is a stateless protocol** where requests are **RESTful** makes that possible: each**request from any client contains all of the information necessary to service the request**.  As a result, any server in the presentation layer can handle any request.  Any application server in the logic tier can handle any request (even if the previous request from that client was assigned to a different server).  Scaling becomes more challenging when we get to the persistence tier.  The shared nothing approach does not apply to databases that receive updates.  Techniques such as database replication or sharding are used to alleviate bottlenecks in the persistence tier.  When the database is read much more frequently than it is written, several copies of the database are kept on different computers.  The copies are referred to as slaves.  The original is the master. Any slave can perform reads, only the master can perform writes, and the master updates the slaves as quickly as possible.  Another approach is sharding.  The database is split horizontally (by rows) and the smaller parts reside on different computers.  The split may be along geographical location or any other criteria. |  |