1. THEORETICAL BACKGROUND

1.1 Web applications

Static HTML Web sites are loosing their popularity, because users expect from modern Web sites more than just representing pictures and text. Generally, users willing to have a higly responsive application with different useful features, working in the Web. As a result Web applications are displacing Web sites on the market.

The difference of Web application from a Web site is the “ability of a user to affect the state of the business logic on the server"[7]. In other words a user or client makes a request to the server, the server performs some actions (calculate, fetch data from database or external Web-service) and sends the response back to the client, which is rendered in the browser see figure 1.1.

Client-server structure helps to distribute application tasks or workloads between the service providers called servers, and service requestors, called clients. Client-server model helps to separate client and server logic, as a result these parts can be independent and communicate via Application Programming Interface (API). Client server model grants several advantages:

- Client and server parts can be developed separately.
- The application may have several clients.
- A client, for example Web browser, can be already created.

Client server model is closely related to a multi-tier architecture - the concept where the parts of the system are divided into separate tiers. Web applications are often use three-tier architecture:

- Presentation tier is responsible for user interface generation and lightweight validation.

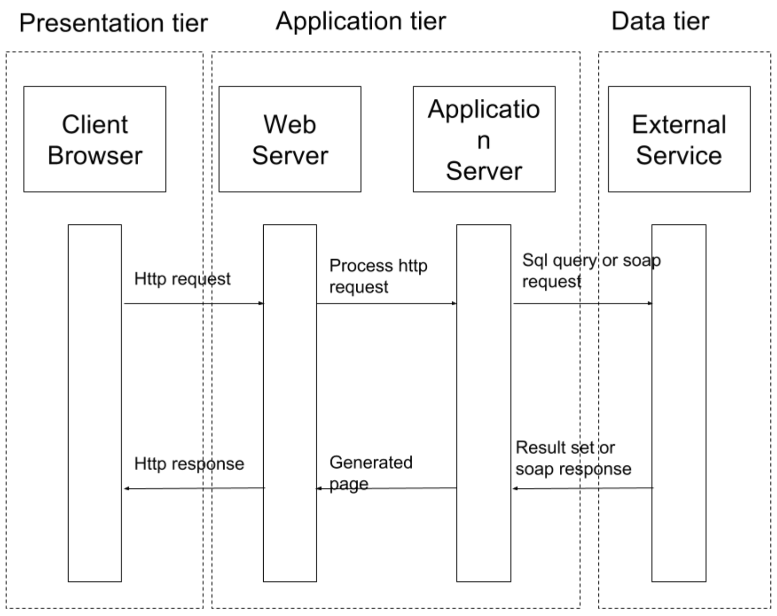

Figure 1.1 *Web application structure*

- Application tier (business logic tier) controls an application functionality, determines how data is created, displayed, stored and changed.
- Data tier - controls databases or other resources, provides access to the data.

Multi-tier architecture allows any of the three tiers to be changed or replaced independently, as a result developers have more freedom to use external libraries and frameworks.

All applications have a lot of common features and problems which were already solved by developers beforehand. It is a good practise to take an already existing solution, than try to implement your own new one. That is why many modern applications are based on one or several software frameworks. In a rapidly changing and highly competitive business environment, choosing a right toolset is one of the key factors of the success. Same implications are applied for testing frameworks.

Nowadays some companies still rely on manual testing or ignore this important part of software development at all. Such approach has several sorrowful consequences:

- The developers are afraid of changing already written code. Because they do not have a confidence that their changes would not break existing code. They stop cleaning their production code because they fear the changes would do more harm than good. “Their production code begin to rot" [1, p.123]

- The effort of finding errors and fixing them raises with the amount of code written, because the developers can not localize the place where the error is actual happening.
- Developing new features becomes harder, if they are based on the part of the system which have errors.
- Costs the whole system increases.

To test easily the huge amount of code an automated Web testing is needed. Web testing reduces the amount of work required to check Web applications as well as Web sites, amplify software value, enhance reusability of test cases and improve time-to-market.

IEEE defines software testing as the process of evaluating a software system to verify that it satisfies specified requirements [2]. A set of requirements for a Web application includes security, performance, presentation, etc. We will focus on several requirements for the Web application which differ from desktop application.

One of the key requirements which makes testing Web applications harder than testing desktop application is support of different browsers and operating systems and also different devices. A lot of desktop applications are developed to support some particular operating system or different versions of the product are developed and maintained for different operating systems.

Web applications on the contrary should support not only different operating systems, but also different browsers and devices. So, if developers team decides to support three operating systems (Windows, OSX, Android), three type of devices (phone, tablet, PC) and three browsers (Chrome, Firefox, Internet Explorer) the number of possible variations is already twenty seven. If you decide to support different version of browsers, which in some circumstances may vary a lot, the number of different configurations of tested machines will be close to one hundred. In this case manual testing is unexceptable, because it will lead to unwarranted expenses.

Another difference between Web and desktop applications is navigation on the Web page and between pages, the unexpected state change via the browser back button or direct Uniform Resource Locator(URL) entry in the browser. Some resources or parts of the application can be not accessible, due to connection problems or maintenance. Such unexpected behaviour may happen, and must be handled properly, not to crash the whole application.

Web testing includes the different type of testing like:

- functionality tests
- compatibility tests
- load tests
- performance tests
- integration tests

All these types of tests are equivalent important and picking a tool which will help to write these tests is not an easy task.

It is an advantage when the testing tool is using same principles and similar programming language with other tools in the project. We think that using same programming language to write both tests and code is much easier for the developer. This idea is related to Test-Driven Development(TDD), when tests are written before production code.

TDD is a very popular methodology of software development. The main idea is to write tests first and then code. The main benefits of such approach are:

- The developer is sure that his code works as intended, because all his code is tested.
- The errors are found at early stage of the development cycle, which reduces the cost of fixing problems.

Three laws of TDD [**?**, pp122][Book page 122]

1. You may not write production code until you have written a failing unit test.

2. You may not write more of a unit test than is sufficient to fail.
3. You may not write more production code than is sufficient to pass the currently failing test.

BIBLIOGRAPHY

[1] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition, 2008.

[2] Lei Xu and Baowen Xu. A framework for web applications testing. In *Proceedings of the 2004 International Conference on Cyberworlds*, CW '04, pages 300–305, Washington, DC, USA, 2004. IEEE Computer Society.