UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Architectural Requirements

Project: Figbook

Client: Figtory Animation

Team: Creativate

Armand Pieterse *u12167844*
Kgomotso Sito *u12243273*
Jimmy Peleha *u12230830*
Sphelele Malo *u12247040*
Ndivhuwo Nthambeleni *u10001183*

Department of Computer Science, University of Pretoria

Here's a link to GitHub.
`https://github.com/SpheMalo/COS-301-Main-Project.git`

July 22, 2015

Contents

List of Figures

1 Architecture Requirements

1.1 Architecture Scope

1.2 Quality Requirements

- **Security**
  - ⋆ The Mediawiki infrastructure provides secure access to the database and the opportunity to modify user interaction with certain pages on the system.
  - ⋆ Session management in the underlying has been accounted for. Additional session management is added to restrict access to unauthorized pages by users.
- **Autidability**
  - ⋆ The Figbook system uses Mediawiki APIs to track changes made my all users on all pages modified. These changes will not be visible to other users except for the administrator and the users working together on manuscripts.
- **Scalability**
  - ⋆ The system uses MySQL in conjuction with MediaWiki APIs. This makes it supportive of both small and large scale use.
  - ⋆ The use of objects to represent the database makes decreases the frequency of expensive calls to the database.
- **Integrability**
  - ⋆ Integrating the system with multiple platforms is not a problem as it is a web-based system.
  - ⋆ Figbook has been integrated with Mediawiki and the MySQL DBMS.
- **Usability**
  - ⋆ The Graphical User Interface is quite elegant, simple and user friendly. Underlying architecture and complicated functionality has been abstracted away from the user.
  - ⋆ A detailed user manual will be provided.
- **Reliability and Availability**
  - ⋆ The system will provide the user with all the functionality required to carry out the entire lifecycle of a book from the comfort of their own home.
  - ⋆ The registration of an ISBN number will be requested from within the system. Failure to obtain an ISBN number is dependent on entities outside the scope of a system.
- Maintainability
  - ⋆ The separation of concerns within the system makes it less tedious to rummage through the code for understanding. Simple calls to the API are used and commenting is to be added effectively.

1.3 Integration and Access channel requirements

Introduction

This section will cover the integration channels between the different systems that will eventually make up a working Figbook system. Access channels will also be discussed, these are the different channels to access the system together with the APIs used to provide this access.

1.3.1 Integration Channels

- Mediawiki - An open source wiki package to be used in Figbook
  - ⋆ The Mediawiki package will be used as an API within the Figbook project, it will provide collabrative writing previllages that are the core of Figbook.
  - ⋆ The Mediawiki database will be integrated with the Figbook database (both mysql) for uniform persistance.
  - ⋆ This package will also provide other services like communication, reporting and Authentication.
  - ⋆ Using Mediawiki as an API will mean that it will be contained within the system reducing dependencies on external systems.
- Bluemix Single Sign On - This service will be used as nice to have for the authentication module, to allow users to sign into the system using external LDAP directories.
- SOAP (Simple Access Project Protocol) allows programs that run on different operating system to communicate using HTTP and XML. It is platform and language independent, It is useful in handling asynchronous processing and it supports many protocols and technologies.

1.3.2 Access channels

- Figbook will be accessible via a web interface that is bootstraped to cater for all devices with browers.
- Figbook should be accessible in the future via a Desktop interface as well as a mobile application (not in scope of the project).
- Access to figbook is provided via the https protocol to ensure security.
- IPsec(Internet Protocol Security) will allow for a secure IP and to ensure no harmful data is ever transmitted to the servers of figbook.

1.3.3 API Specifications used for integration and access

- Web Service Definition Language(WSDL) - WSDL will be used to describe the functionality and the operations provided by the web-based service (Buzz system).
- Interactive Data Language(IDL) - This will be used for data handling purposes for the data source integration for Mediawiki and figbook and any other form of data required by the system.

- Php - Mysql APIs will be used to query the database.
- Mediawiki uses a RESTful web services API that allows users to access the services without knowledge of the inner system.

1.4 Architecture Constraints

Introduction

This section covers the different architectural factors that will place some constraints on the Figbook development. These constraints are divided into two categories as specified below.

1.4.1 Technical Constraints

- Programming languages - Figbook is a web-based application (fundamentally) and is being developed using the different html based languages mentioned in section 5(Technologies). The client however was requiring that there should be desktop and mobile interfaces, in which case the team will have to use programming languages outside the mentioned ones.These two additional interfaces will be made available only if time allows as they are nice to haves.
- Operating systems supported - As Figbook is web based it should be on all OSs that support web browsers. The hosting of this application is provided in the Linux OS and is configurable to run on a Windows OS as well.
- Platforms supported - These are the different browsers the application is required to run on.
  - ⋆ Internet Explorer - Versions 6 to current.
  - ⋆ Chrome - Versions 42.* to current
  - ⋆ Firefox - Versions 87.* to current
  - ⋆ Safari - Versions 5.1 to current
  - ⋆ Opera - Versions 12.1* to current
  - ⋆ iOS - Versions 6.1 to current
  - ⋆ Android - Versions 2.3, 4.0 to current
- Libraries and Frameworks - The client requested that we look into the Mediawiki library as it has collaborative writing in it. This meant that out of the different APIs available to build Figbook, mediawiki was a winner as suggested by the client.

1.4.2 Business Constraints

- Team composition - Figbook is being developed by just the 5 of us in the group, and the client provides advice on how we can complete the project, but all decisions are made by and within the team.
- Schedule - This project is to be completed within the dates specified by the University of Pretoria. The demo of the project will take place on the 23rd of October 2015, the project should be complete by that date.
- Software licensing - All APIs and/or software used to deliver the Figbook application should be and is open source. This will make it possible for the project to be placed under the GPL V3 license after completion.

2 Architectural patterns or styles

Introduction

This section will cover the architectural patterns and/or styles we will be using to solve certain challenges we might face, as well as provide us with a way to meet certain quality requirements.

2.1 Three-tier Architecture

- Reasons for using:
    - ⋆ Is to allow any of the three tiers to be upgraded or replaced independently, when changes in requirements or technology require such upgrades or replacements. Thus addressing **maintainability**.
    - ⋆ It is an applicable solution for web development. Where the three tiers are divided up into:
        - ∗ **Presentation-tier:** Which will be the front-end of the website.
        - ∗ **Application-tier:** This is the logical tier which provides the application's functionality.
        - ∗ **Data tier:** All the data persistence mechanisms which will be used.

3 Architectural tactics or strategies

4 Architectural Tactics/Strategies

4.1 Security

Authentication - Achieved via the Authentication module of the website. Users need to log in with a valid username and password before they can use any functionality of the website.
Encryption - Sensitive information like user passwords, content of the books etc is encrypted in the database instead of being stored in plain text.

4.2 Integrability

Project planning and system process management - by assigning roles and making use of an online scrum tool i.e ASANA.
Manual programming
Test specifications are also defined and documented to provide testing steps.
Canonicals Read and write operations (master/slave).
Caching - camelCase.
design patterns - e.g Edit Pages: Template Pattern

4.3 Scalability

Memcached - is used to cache image metadata, parser data, differences, users and sessions, and revision text. Metadata, such as article revision history, article relations (links, categories etc.), user accounts and settings are stored in the core databases.
Compression - Text is compressed and only revisions between articles are stored.
Scale by separating Read and write operations (master/slave).
Caching - Cached expensive results with the use of temporal locality of reference e.g Reference value(row in array) each iteration and cycling through loop repeatedly .
Avoided expensive algorithm and database queries

4.4 Usability

Naming Convention - Buttons and information on the website are labeled with functional names. Whatever a section/button/link does, is what it's called (avoided "Click Me" or "Click Here" type of labels).
Logical Flow of Page Elements - The webpages follow a human natural flow with its content (beginning of elements on the left, anything that follows from there goes in a right-down direction).

4.5 Maintainability

5 Use of reference architectures and frameworks

5.0.1 Channels

- REST: Representational State Transfer
  - ⋆ The REST architecture design is a good option to use as it is a simpler than SOAP and is a dynamic design.
  - ⋆ A RESTul system can integrate well with HTTP as RESTful systems are optimized for the web.
  - ⋆ Restful systems needs to follow a client-server model,so this means that there needs to be communication between the client and server which is vital in the buzz system.
  - ⋆ There is support for a lot of components to interact with each other and to be interchangeable.

5.0.2 Protocols

- Http(Hypertext Transfer Protocol) is the main protocol for all websites in the modern internet usage.It allows linking of nodes which allows the users to easily navigate through web pages.
- HTTPS is a more secure version of Http.HTTPS is a combination of HTTP and TSL and SSH.This protocol will ensure that data is safely transported.
- TCP/IP(Transfer Communication Protocol/Internet Protocal) Used to as communication over the internet.TCP is reliable and is able to check for errors in the transfer of the page over the IP.
- SMTP(Simple Mail Transfer Protocol) is used to send emails.This will be easier then mailing manually and is prominently used in the web space.
- IPv6(Internet Protocol version 6) This will redirected and to allow them to be redirect users or routed to the correct space on the internet.This provides access to buzz through the use of http and the TCP/IP protocols.
- IPsec(Internet Protocol Security) will allow for a secure IP and to ensure no harmful data is ever transmitted to the servers of buzz.

5.0.3 API Specifications

- Web Service Definition Language(WSDL) - WSDL will be used to describe the functionality and the operations provided by the web-based service (Buzz system).
- Common Object Request Broker Architecture(CORBA) - CORBA will mediate the communication between the diverse systems that will be integrated to the Buzz system to provide added functionality and data for the operation of the system.
- Interactive Data Language(IDL) - This will be used for data analysis purposes for the data passed from the data source to the Buzz system and any other form of data required by the system.

6 Technologies

- PHP 5
- MediaWiki 1.25.1
- MySQL
- Bootsrap
- Javascript
- HTML 5

7 Bibliography