Woodgate Software: High End Architecture and Prototype

Ross Woodgate
CPIT
Madras Street
Christchurch 8011
+64 3 9408888
rww0065@student.cpit.ac.nz

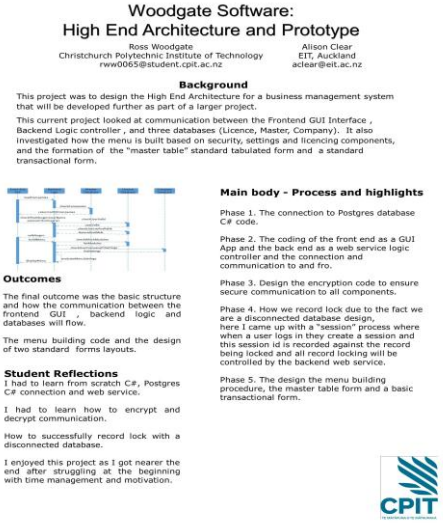
Alison Clear
EIT, Auckland
Airedale Street
Auckland 1010
+64 9 4444303
aclear@eit.ac.nz

ABSTRACT
One of the authors has been involved with Accounting and Business Management software for 15 years and has a vision for designing his own Business Management with a focus later on job costing systems. This project looks at the design of some of the initial high end architecture and building a prototype. In particular the login routine that needs to look at many databases and variables to determine a valid entry into the system. There is also a complicated hierarchy to the settings and security features that are required due to the system being built to suit many customers' requirements. The asynchronous messaging required to inform all users of what records are required by other users and to allow them to act on this information was developed.
The project also looks at a standard saving form for customers and also a transactional entry form. Three main modules were developed and fully tested.

Categories and Subject Descriptors
D.2 SOFTWARE ENGINEERING

General Terms
Management, Documentation, Design,

Keywords
Accounting software, database, software engineering


Woodgate Software:
High End Architecture and Prototype

Ross Woodgate
Christchurch Polytechnic Institute of Technology
rww0065@student.cpit.ac.nz

Alison Clear
EIT, Auckland
aclear@eit.ac.nz

Background

This project was to design the High End Architecture for a business management system that will be developed further as part of a larger project.

This current project looked at communication between the Frontend GUI Interface , Backend Logic controller , and three databases (Licence, Master, Company). It also investigated how the menu is built based on security, settings and licencing components, and the formation of the "master table" standard tabulated form and a standard transactional form.

Main body - Process and highlights

Phase 1. The connection to Postgres database C# code.

Phase 2. The coding of the front end as a GUI App and the back end as a web service logic controller and the connection and communication to and fro.

Phase 3. Design the encryption code to ensure secure communication to all components.

Phase 4. How we record lock due to the fact we are a disconnected database design, here I came up with a "session" process where when a user logs in they create a session and this session id is recorded against the record being locked and all record locking will be controlled by the backend web service.

Phase 5. The design the menu building procedure, the master table form and a basic transactional form.

Outcomes

The final outcome was the basic structure and how the communication between the frontend GUI , backend logic and databases will flow.

The menu building code and the design of two standard forms layouts.

Student Reflections

I had to learn from scratch C#, Postgres C# connection and web service.

I had to learn how to encrypt and decrypt communication.

How to successfully record lock with a disconnected database.

I enjoyed this project as I got nearer the end after struggling at the beginning with time management and motivation.

CPIT

This poster paper appeared at ITX 2014, incorporating the 5th annual conference of Computing and Information Technology Research and Education New Zealand (CITRENZ2014) and the 27th Annual Conference of the National Advisory Committee on Computing Qualifications, Auckland, New Zealand, October 8-10, 2014. Mike Lopez and Michael Verhaart, (Eds).

1. INTRODUCTION
This project focused on the set up the Front End GUI, Back End Web Service, and databases required for the development of the initial high end architecture and a prototype for the development of business accounting software.

This project built the logic for:

- How we connect to the Databases
- Communication between front end and back end applications
- Logging On
- Building the menu structure
- Standard "Master Table' Form
- Standard Transactional Form

2. DEVELOPMENT OF THE PROJECT

2.1 Structure of the Progam
The application was designed so any front end device will be able to be set up easily. The frontend can be any mobile device, a website or a PC App and can communicate with the web service successfully. The structure also enables the software owner to manage licensing requirements with one cloud License Database, enabling the implementation of temporary licensing in the future.

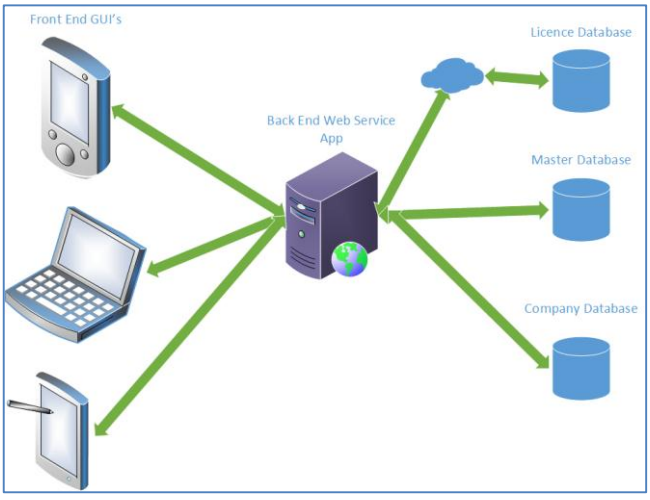


Fig 1 Diagram of Program Structure

2.2 PostgresSql and C#
For this project it was decided to program in C# and to use PostgreSql as the database. C# was chosen due to this being the most popular programming language for applications and the

ease of writing in C#. PostGreSQL was chosen primarily because it has no limits to size and is an open source free ware database, it is also a database that the programmer has used previously.

2.3 Logon procedure

When the login screen is loaded it looks at the Master Database for the list of companies available and loads the company dropdown box with the list of companies

It then checks in the licensing database

- The number of licences for the current logged on
- the Licence Database that the user logged in has access to the relevant company
- the number of licences currently in use is less than number of licences
- If licence available then we insert a new record into companyuseraccess

It then checks in the Company Data Table

- If user exists (note double check user should always be in both the Master and Licence Database)
- If password correct
- Then it creates a session and returns the sessionid back to the login form and loads the menu

2.4 Building the Menu

The menu was designed to be as wide as the screen and to have an area to show any messages sent to the particular session. This will allow for use of Excel or other packages to be used easily without the software taking any more real estate than necessary.

2.5 Record Locking

Due to the fact we were using a disconnected database record locking was required to be developed it was decided to use sessionID as the record locking method. When a user logs on they create a session record in the customer database. When we need to get exclusive access to an item eg editing a record then we update the sessionid field on that record with the current sessionid. If someone else requires access the method of looking at the lock will be as follows:

- If there is no sessionID against the record then we can use safely
- We then look at the Session record to see if the session has been logged off if it has we remove the sessionid from the record we want to edit and continue on.
- If there is a sessionID and the session is not logged off we see how long the session has been idle by looking at the session record dateLastTrans if this is older than a preset time then we will through Asynchronous messaging communicate with the session that has the record locked,
    - If no confirmation message is returned from the front end then we will assume the session has been disconnected and will log off the session remove the record lock we know about and continue on
    - If a confirmation record has come back we will advise user the record is a valid lock and a message has been sent to the session and we are awaiting further action

2.6 Nunit Testing

Nunit Testing was setup to check the routines and that they worked ok. This was not totally completed the framework for testing was developed and setup.

What this requires is setting assertion messages in the test regime and running the test when it returns an assertion that does not match the programed assertion message it returns an error to the Nunit testing platform.

3. PERSONAL REFLECTION

This project has taught me quite a bit to what is required to complete the software. I have struggled at many corners to understand the concepts required to complete this project.

Some of the many difficult concepts I have learnt from this project are:

- Web Service programing
- Asynchronous Messaging
- Decryption /Encryption
- Nunit Testing
- Passing variables and classes to front end using binary serialisation

4. CONCLUSION

The three main modules of the proposed Business and Accounting Management software were fully developed and tested. The Front End GUI, Back End Web Service, and databases required for the development of the initial high end architecture and a prototype for the development of business accounting software were completed.

5. REFERENCES

[1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst.* 15, 5 (Nov. 1993), 795-825. DOI= http://doi.acm.org/10.1145/161468.16147.