YEROTH–ERP–3.0 | GUIDE PRATIQUE

PR. XAVIER NOUMBISSI NOUNDOU

Ce document explique de manière simplistique le rôle de YEROTH–ERP–3.0, et dorénavant, sera le guide pragmatique de son utilisateur.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This introduction motivates why I created YEROTH–ERP–3.0, and why it uses the best software programming language of its time !

1.1 Motivation

YEROTH–ERP–3.0 is an **Enterprise Resource Planing (ERP)** software–system that aims 'effectiveness' and 'simplicity', compared to other high ranked ERP software–systems (e.g.: 'Sage Gescom i7', 'SAP Business One', etc.).

We chose to design and implement YEROTH–ERP–3.0 as a thick–client software–system because of the following reasons:

1. The implementation language C++ offers much flexibility:
   1. *MULTIPLE INHERITANCE:*

      It allows developers to abstract as much as possible business code upwards, away from downwards implementation classes. For instance, in YEROTH–ERP–3.0, GUI–Qt–windows inherits for instance *search filtering feature, and print capability from* 2 *different classes*.

      Print capability *couldn't be inherited from the same class where search filtering is abstracted and partially implemented (interface in* Java *for instance doesn't allow any method body code), because it works in its pure abstract class (*C++ *class with at least* 1 *empty method body), together with feature* ***database column filtering for viewing and printing.***

      The drawback of the multiple inheritance in C++ *is: it sometimes can be very difficult to build it using "*gcc (g++)*[GCC]" !*

   2. *MACROS:*

      They enable developers to create TEXT TEMPLATE in their code.

      For instance, I use macros in some parts of my code to increase execution time, and reduce stack activation for method or function calls in YEROTH–ERP–3.0 !

2. The availability of 'WHAT YOU SEE IS WHAT YOU GET' (WYSIWYG) tools for fast and useful user interface design (e.g.: Qt designer [Com20], miniStudio (vxWorks) [WEI20], etc.)
3. The low number of logical software–system architecture layer (i.e.: 2.) involved with the use of a thick–client software–system architecture, as opposed to a web–browser–based software–system (i.e.: 4, *client user interface*, *presentation layer*, *business logic*, and *data (DBMS)*).

Chapter 2

Conclusion

This conclusion explains why YEROTH–ERP–3.0 uses the BEST SOFTWARE TECHNOLOGY IN TERMS OF SOFTWARE–SYSTEM ARCHITECTURE !

	YEROTH–ERP–3.0	Odoo
libraries & programs	lxqt-sudo, etc.	python-lxml, etc.
user interface code TOOLS WYSIWYG	QT–DESIGNER	(CUSTOM BUILD) FRAMEWORKS
business code	C++	Python, JavaScript, XML
DBMS	MySQL	PostgreSQL
web–server		Werkzeug

Table 2.1: YEROTH–ERP–3.0 VS. Odoo.

YEROTH–ERP–3.0 has a thick–client software–system architecture because we found thick–client software–system architectures simpler than web–browser–based software–system architectures.

Thick–client software–system architectures is simpler because it requires less layers in its logical (or physical) software–system architecture, and is easier to develop and maintain as a software–system application.

Table **??** illustrates a thick–client software–system is SUPERIOR IN TERMS OF TOOLS FOR MAINTENANCE AND DEVELOPMENT than a web–browser–based software–system !

A web–browser–based software–system architecture has more drawbacks as follows:

1. it requires at least 2 other software–systems, *apart from the ones normally required by developed software–system itself, for instance libraries (e.g.:* Log4j*),* to fully operate (e.g.: web server, application server, etc.).

   Table 2.1 depicts this situation in the light of the open source ERP software–system Odoo.

   Accordingly, a thick–client software–system doesn't require any running and managing infrastructure such as for example an application server !

2. A web–browser–based software–system requires at least 4 layers in its logical system architecture (e.g.: client, presentation, logic, and data layers).

   Accordingly, a thick–client software–system only requires at least 2 layers !

3. A web–browser–based software–system potentially entails more software security vulnerabilities because its implementation requires the use of at least 2 different programming languages, and frameworks in combination.

   Accordingly, a thick–client software–system needs only the use of 1 homogeneous software programming language !

Chapter 3

Bibliography

[Com20] The Qt Company. Qt Designer Manual. http://doc.qt.io/qt-5/qtdesigner-manual.html, 2020. Last accessed on September 4, 2020 at 15:21.

[GCC] THE COMPILER SUITE GCC. THE GCC (G++) COMPILER SUITE. http://www..org. Last accessed on December 29, 2020 at 12:00.

[WEI20] Yongming WEI. miniStudio User's Guide. http://www.minigui.net/en/ministudio, 2020. Last accessed on September 4, 2020 at 15:21.

Index

Appendix A

Presentation Documents of open source software–system YEROTH–ERP–3.0