A KISS Ada GUI

Xavier Petit, Stéphane Rivière
Sowebio SARL, 15 rue du Temple, 17310 Saint-Pierre d'Oléron - France; +33 5 46 85 58 40; dev@soweb.io

Abstract

Presentation of a KISS Ada GUI included in the general-purpose framework LibreFrame.

Keywords: Ada, Sowebio, KISS, GUI, v22, LibreFrame.

1

In this presentation, we are going to talk about a five years journey, with an introduction to the work done so far, then the presentation of a new Ada GUI, and a conclusion on the work to come.

A.1 Background - 1:30 mn

2

Sowebio is a company with two complementary activities. On one side, a multimedia web and SAAS agency and, on the other, an IT engineering service called Sowneo, where we practice outsourcing, server clustering, virtualized instances, cloud escaping, industrial, management and embedded development and, at the end, electronic design including micro-controllers programming.

Our own high-performance servers cluster is used to host our web sites and SAAS, enabling us to achieve negligible operating costs for our hosting business.

3

Xavier and I run Sowneo. Our skills complement each other.

4

In terms of computer engineering, the only language we use is one of a kind. Whatever the projects described above, from clustering to embedded programming, we only use one language. But I'm not going to tell you which one ;)

A.2 Our definition of KISS - 2:30 mn

Before describing our path to LibreFrame, I'd like to talk about our definition of KISS.

5

A framework must be designed to be understood by a single person in a minimum of time, so one can quickly create self-contained, simple and productive applications.

The quest for simplicity consists in selecting the real needs, and limiting oneself to sufficient functionalities. This limitation is not pejorative, but a manifesto of lightness.

6

We should behave like craftsmen, not "geniuses".

In IT, you have to avoid "geniuses".

7

Working with "geniuses" is a pleasure, really.

8

Today, to build an application, a single person has to master an incredible house of cards full of languages, concepts, protocols, procedures, best practices, dialects, templates and tricks. This mastery is difficult to acquire, and will have to be renewed on a regular basis, even if the novelty is not justified by productivity gains.

9

There is too much of everything.

You need to create web SAAS with a development environment, then create Android app with another and finally create iPhone app with a third. And be modern. To be at the cutting edge, don't forget to migrate your applications written in Java to Kotlin or from Objective-C to Swift.

10

One day, we sat down by the side of the road, asked ourself the right questions, and chose this very secret language.

Could we do the same for a framework?

A.3 The v20 and v22 experiments - 5:30

11

We developed the v20 framework that served as the foundation for our servers cluster manager.

12

Then, we developed v22, a GP framework, with much broader ambitions, encompassing web development according to a philosophy that avoids the use of HTML/JS/CSS hell, by defining, straight in the code, blocks and sub-blocks of graphic elements, the latter automatically arranging themselves on the screen according to the terminal used.

13

We're going to take a little time describing v22, as LibreFrame takes over most of its features, with the notable exception of the GUI.

When developing an application using v22, we used to eat our own dog food[1], in order to continuously improve both the application and the framework.

14

To deploy applications, we use GNAT Remote, our deployment utility for Alire projects. It manages two DEV and PROD branches and restarts Linux services via systemd.

Here's an example of the actions performed for the vending machine telemetry server DEV branch.

At first, it backups sources.

15

Then remote copy and remote build.

16

And finally remote service restart to handle the new fresh binary.

17

The v22 framework handles all practical aspects of software development, making life easier for developers.

It supports post-mortem traces, multi-level logs, configuration files, file management, directories and searches.

The console interface handles text I/O, integrates cursor management, colors and various animation effects.

The Web interface, based on Gnoga, adds users management, integrates high-level CRUD functions as well as menus and common widgets, including responsive screens, to offer a consistent interface on all devices.

SQL handling includes key-value store for application level parameters, high-level CRUD functions, and managing database schema updates, making deployment of new versions a breeze.

The system package manages environment variables, system paths, return codes, CTRL-C handler, Debian and derived systems packages, memory observation, execution of external programs with retrieval of output codes and text response.

The remote access package, through SSH with key and password management, allowing all commons files and mounting operations. It also manages e-mail, SMS sending and communication with APIs.

Date management, CRC support, password generation and control, high level PDF generation and extensive text file management are also provided.

For UTF-8 handling, v22 uses UXStrings package from Pascal Pignard. Its speed and ease of use is a foundation for v22 KISS spirit and enable us to process big-data with files up to hundreds of GB.

18

The v22 reference manual is around three hundred pages long. It comes with an other hundred page user manual for the development environment, including Alire, with a few exclusive details to help you understand it.

A.4 Create with v22 - 1:30 mn

19

Sowneo has developed a vending machine management application, which consists of three Linux services:

- A telemetry server with hundreds devices in a VPN over LTE network;

- A web application for users and administrators;

- A background service sending SMS or email and making financial and inventory consolidation.

This application features over eight CRUD and more than fifty screens, lists and reports.

Without v22 dependencies:

- The application alone is coded in just over seven thousand lines in forty two files;

- With v22 framework, the total is just over fifteen thousand lines in eighty nine files.

The web application can also be embedded into generic smartphone apps to ease users.

A.5 Results of using v22 - 1:15 mn

We are now able to draw some conclusions.

The development of this seven-thousand-lines project validates our design choices well beyond our expectations.

Fewer lines of code means fewer bugs, less time coding and less stress.

20

The v22 framework requires no HTML/JS/CSS knowledge and the interface elements are self-placing with responsive management.

21

In this environment, productivity is outstanding, ensuring a remarkable competitive edge over the competition.

22

By freeing ourselves from Android and iPhone specific development, we can cut our sales proposals by a factor of three.

That's exactly why we won the vending machine contract.

A.6 From one framework to another - 1

23

1 https://en.wikipedia.org/wiki/Eating_your_own_dog_food

While v22 has exceeded our expectations, we can do even better.

After intensive use and dogfooding, we have identify several areas for improvements in terms of refactoring, internationalization, SQL extensions and graphical interface.

It's now time to introduce our new framework, which was called v25, before being renamed LibreFrame, since we believe we can build on it for the long term.

LibreFrame represents the last iteration before we pivot our developments towards higher-level softwares. It takes up the essentials of the v22 framework and incorporates the improvements mentioned above.

LibreFrame's innovation lies in its brand new graphical interface, coded from scratch, and I'm pleased to hand over to Xavier to present it to you.

B.1 A new GUI paradigm

24 25 26 27 28 29 30 31 32 33 34 35

See notes in sow-20250613-AEiC2025-slides.odp

C.1 En route for the first release - 1 mn

36

We want “everything in the code” while wishing few lines of code, traceable code and high performance code.

To achieve these goals, we are focusing on a reasonable level of abstraction, genericity and state machines, but limiting tasks and objects.

Our todo list, in no particular order.

Text and SDL display modes, customized themes, more general purposes widgets, industrial and geographic widgets, internationalization, localization, help system, web socket layer rewriting, SQL layer refactoring, make LibreFrame portable with all Linux, *BSD, MacOS and Windows systems.

Our todo list is opened. This Workshop is also an opportunity to gather feedback to improve LibreFrame.

C.2 Acknowledgements - 1 mn

37

Thanks to all Adacore and GCC people and the package designers David Botton, Dmitry Kazakov, Pascal Pignard and Gautier de Montmollin.

We also like to thank two Sowneo interns from ENSEIRB engineering school, Théodore Gigault, who cleared the way for Gnoga, and Arthur Le Floch, who came up with the idea of graphical blocks to build the interface.

We would also like to thank Jean-Pierre Rosen, Fabien Chouteau and Olivier Henley for their friendly support, as well as Fernando Oleo Blanco and his team for the organization of this second Ada Developers Workshop.

Finally, thank you all for listening. We hope to see you at the next FOSDEM for the first release of LibreFrame.

This presentation is dedicated to Simon Wright.

References

[1] https://www.sowebio.com/software-embedded-applications

[2] v22 framework on the web: https://v22.soweb.io

[3] On github: https://github.com/Blady-Com/gnoga