[Floss P1] LEGO-UI

Juri Schreib(https://github.com/Bujuhu)

Nikola Prvulovic(https://github.com/Dynamichost96)

Project Variant: 1, New Open Source Initiative

Title: lego-ui

Language: GO

Tracker: LEGO-UI on GitHub(https://github.com/Bujuhu/lego-ui)

Table of Contents

Project-Proposal

lego-ui is a graphical user interface written in go developed on top of the go-acme/lego library to make it simple for everyone to get DNS based LetsEncrypt Certificates.

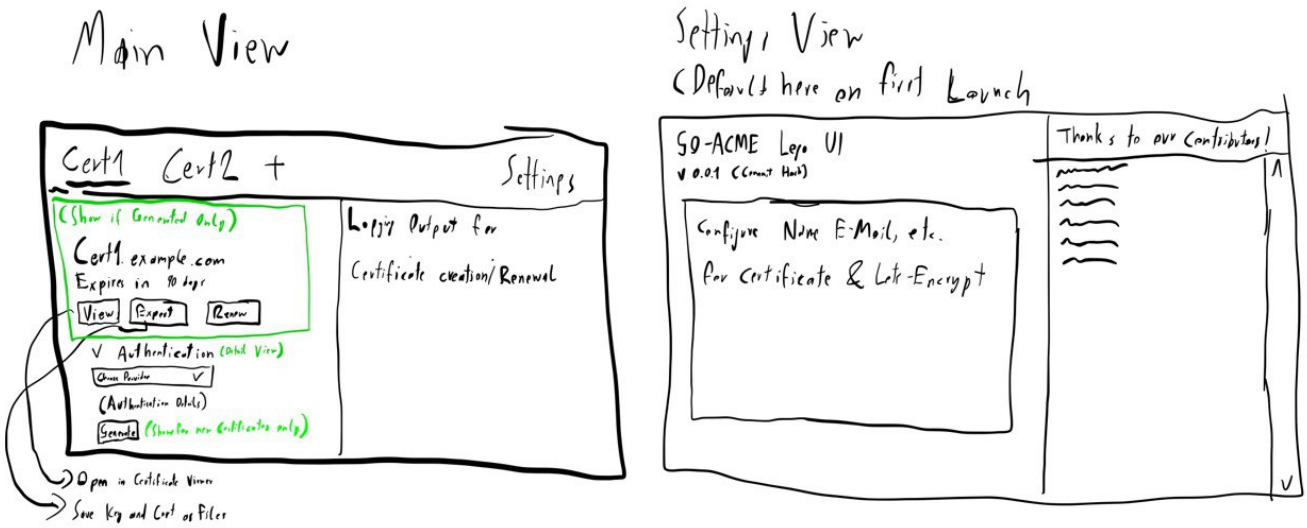
Vision

For novices, it can be pretty difficult to generate new TLS certificates for their devices. While there are some great solutions like from Synology, Cpanel, Nginx Proxy Manager or even GitHub itself, which are great to use but force the user to host their content in certain (sometimes proprietary) environments or using specific software.

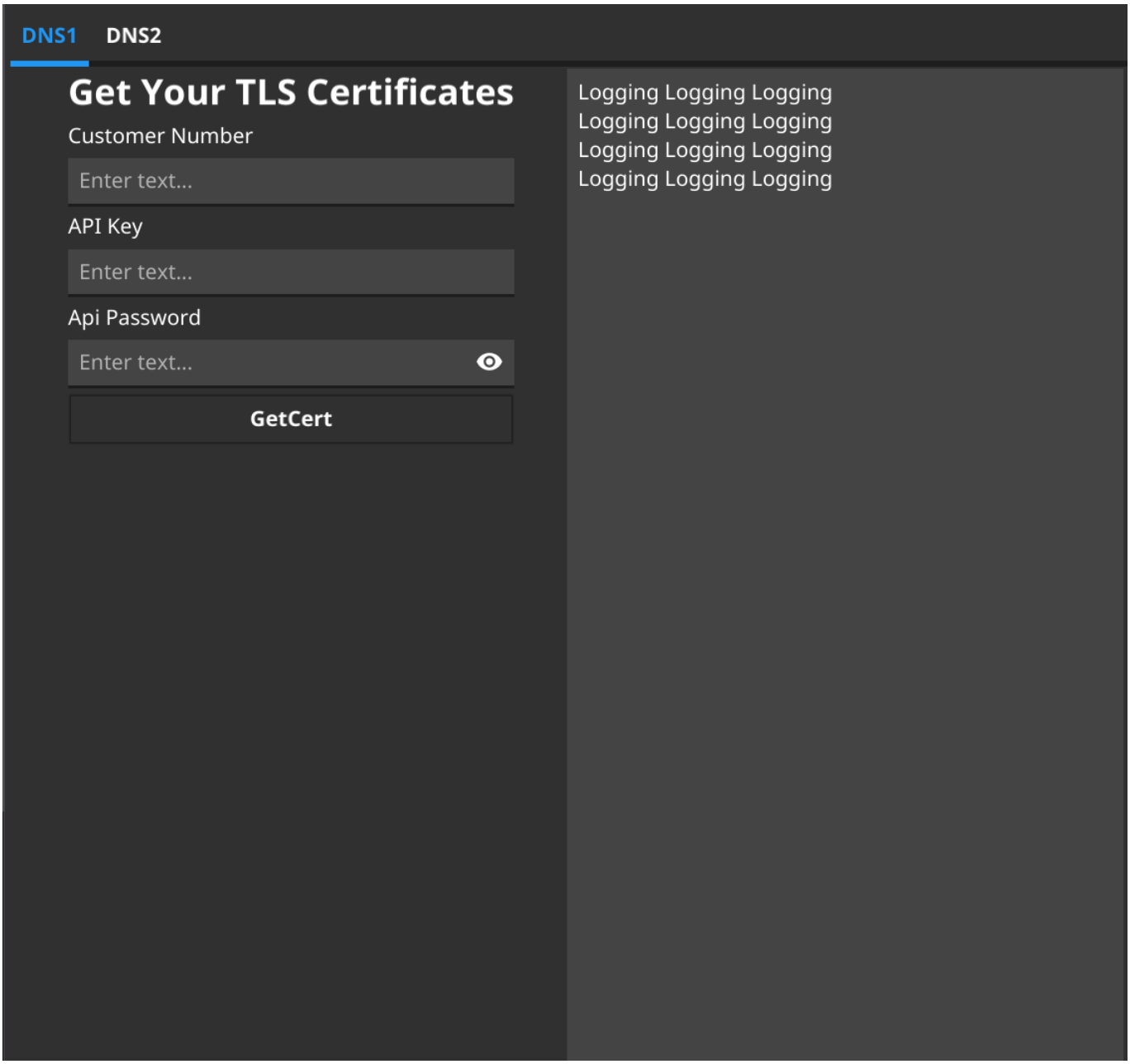
If users want to create TLS certificates, they need to configure their network, like opening a port (which some isps won't even allow), or configuring a reverse web proxy (which requires an additional provider or cloud server).

To improve this process, we want to create a User Interface, using the dns-acme challenge. This means, that only authentication data and personal information to generate the TLS Certificate are needed to generate a Certificate using Letsencrypt. The downside of this is, that the user needs a domain.

Mockup



Current Progress



Implementation

We want to implement the application using the go programming language, using the go-acme/lego library and fyne-io/fyne as our Hui library. Lego provides a go library to make DNS challenges out of the box to loads of different DNS providers. We decided on fyne as our GUI library, because it

- Runs natively on the machine (which means better integration into system features like keyring and the file system)
- Well-defined design template (reducing amount of work needed to design the UX)

Our architecture is based on the model view viewmodel pattern. This allows us to keep UI components modular, to ensure forms for additional DNS providers can be implemented quickly.

Features already implemented

- Generate a Certificate for Netcup Certificates
- Save Personal Information to use for certificates
- Save DNS-API credentials using the keyring (if keyring exists)

- Put the certificate into the local keyring or export it to the local file system if no keyring exists
- Renew Certificate
- Export Certificate

Open Features

- Certificate Expiration Warning

Changes

We've previously stated, that we want to make it possible to automatically configure the host file using elektra based on the domain name of a generated certificate. This however would require elevated privileges for automated setup, so we will this further back to explore, how/if we can implement this feature safely

Contributing

As the project is rather new, we are still developing contribution guidelines, but we warmly welcome everyone to contribute! If you have your own ideas, just create an issue in the GitHub repository we've linked to at the top of this document. You can also start working on any of the Tasks we've listed in "Open Features" or any other QoL improvements, like guides, CI/CD (including Reproducible Builds (Paper)) or anything else that comes to your mind. Feel free to ask any question, by opening an Issue on GitHub.

As we are handling very sensitive data as part of this project, a strict end extensive security architecture is also needed. We are utilizing this Paper to identify common fallacies and security issues, within open source projects.

Reducing Entry Barriers

- We are hosting our code and documentation on GitHub. This encourages collaboration through socail coding practices and makes our project more visible to a general audience. This Paper explains in great detail, how and to what extent GitHub is having a positive effect on open source projects. Additionally, we want to incorporate the findings of this paper to create a development process, that encurages collaboration
- We will have extensive documentation for both the source code in from of godoc (A javadoc derivative for the go programming language), and guides to get started, testing, compiling and contributing to the software.
- We will try to implement the project, using as much "vanilla" go as possible, and try to avoid the use and implementation of complex frameworks, which could make it more difficult for potential contributors to ease into the project.
- At a later point, we might produce videos or do live-streams while working on the project to reach a wider audience and allow developers to get an idea how the development process works.
- We want to establish quality measurements, using performance indicators, based on this paper

User Documentation

User Documentation can be found here

Decision

We made a descission on UX of the application, regarding the grouping of certificate per DNS provider or not. The decision can be found here

Testing for Release

Before we create a new release build of the application, the following criteria must be met:

The application must be capable of genreating TLS certificates for all implemented DNS Providers. Doing so as of now cannot be automated as either a mock of every DNS provider API is needed or a domain for every implemented DNS provider exists, which would be prohibitivly expensive.

Currently we rely on screenshots of contributors, to confirm, that when a change was done the process of creating and renewing certificates is sucessfull for every implemented DNS provider. However, due to privay concerns, we are not able to disclose theese screenshots publicly.

We are looking into ways to automate this process and make it more transparent. Your feedback is greatly appreciated!

For features that do not concern the creation or renewal of certifices, we rely on automated unit tests, based on the go and fyne testing frameworks. Before a new release can be built, all tests must be enabled and pass.

Maintanance

Maintainance is particularly important, as the API of underlying DNS providers could change anytime. To ensure we maintain a high degree of functionality with DNS providers, we must keep the go-acme/lego dependency up-to-date. This makes it transparent when underlying APIs change, and the input forms must be adapted. Before creating a new release, every DNS provider must either work or be disabled.

- Release builds will be posted on the GitHub Page of this project.
- As the project relies on linux specifc features, the binary will only be provided for linux
- Release builds must be created and posted by the owners of the Prject, currently @Bujuhu and @DynamicHost96
- libraries that do not facilitate IPC (e.g. keychain), should be bundled with the application

Directory Structure

- docs: Documentation, Rules, Tutorials, and other documents in relation to this project
    - desicions: Architectural and UX desicions about this project.
    - project-description: Project-management style reports and definitions
    - user documentation: User-facing help and guides to use the project
    - contributer documentation: Documentation to develop and contribute to lego-ui
- src: main directory for the source code of this project
    - services: models, based on a service pattern
    - dnsViewModels: bundled units of view and models to implement additional DNS providers
    - view: windows, views, partial views and widgets, that make up the graphical user interface

Contributors to current Work

- UX-Design, GUI Development, UI Tests: @DynamicHost96
- Services and Tests: @Bujuhu
- P3: @Bujuhu, @DynamicHost96