



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

ASSIGNMENT OF MASTER'S THESIS

Title: Redesign of the router administration web interface
Student: Bc. Bogdan Bodnar
Supervisor: Ing. Jiří Hunka
Study Programme: Informatics
Study Branch: Web and Software Engineering
Department: Department of Software Engineering
Validity: Until the end of winter semester 2020/21

Instructions

The goal of the theses is to redesign the Forris configuration interface of the Turris project developed by the CZ.NIC association. The final solution should support all hardware configurations of Turris routers and communicate with the configuration back end via a Forris-controller program.

Follow the steps outlined below:

1. Analyze the solution in place and competing solutions.
2. Choose proper technology and design a front end including a user interface focusing on maintainability, usability, and simple extensibility via plugins.
3. Implement the front end based on design. Implementation should be open-sourced, well documented and should contain a demo plugin with instructions to serve as an example for the development of future extensions.
4. Design, implement and evaluate appropriate tests.
5. Create a deployment package for the Turris OS and all supported devices.
6. Distribute the generated package to routers using test development branches and perform the user testing.

References

Will be provided by the supervisor.

Ing. Michal Valenta, Ph.D.
Head of Department

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Dean

Prague February 20, 2019



**FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE**

Master's thesis

Redesign of the router administration web interface

Bc. Bogdan Bodnar

Department of Software Engineering
Supervisor: Ing. Jiří Hunka

June 28, 2019

Acknowledgements

I would like to thank my supervisor Ing. Jiří Hunka, for his guidance. Also, I would love to express my gratitude to Turris team members, especially to Robin Obůrka and Štěpán Henek. I would also like to thank Alexander Shatrovsky for his corrections of grammar. Many thanks to my family who supported me during my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on June 28, 2019

.....

Czech Technical University in Prague

Faculty of Information Technology

© 2019 Bogdan Bodnar. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis

Bodnar, Bogdan. *Redesign of the router administration web interface*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

Tato diplomová práce se zabývá přepracováním administračního rozhraní pro Turrís routery vyvinuté sdružením CZ.NIC.

Hlavním úkolem práce je návrh, implementace a testování nového administračního rozhraní s důrazem na rozšiřitelnost a za pomoci moderních přístupů a technologií. Veškerá implementace je vyvinuta na základě znalostí získaných z analytické části práce.

Tato práce obsahuje analýzu stávajícího rozhraní Foris a konkurenčních řešení. Práce také popisuje návrh architektury a design uživatelského rozhraní zvoleného řešení. Výstupem této práce je samotný zdrojový kód administrátorského rozhraní, včetně automatických testů, a výsledky uživatelského testování.

Klíčová slova Turrís, Foris, administrační rozhraní pro směrovače, Python, Flask, React

Abstract

This master thesis deals with the redesign of the web administration interface of the Turris routers which are being developed by the CZ.NIC association.

The main goal of this thesis is to design, implement and test a new extensible router administration interface, using modern approaches and technologies, which will replace existing solution. Entire implementation is based on the knowledge acquired in the analysis part.

This thesis contains the analysis of the current Foris interface and competing products. It also describes the architecture, UX and UI design of the final solution. The output of this work also includes the new interface itself, tests for the code base and results of the usability testing.

Keywords Turris, Foris, router administration interface, Python, Flask, React

Contents

Listings	xv
Introduction	1
Background	1
Motivation	1
1 Analysis of similar solutions	3
1.1 Analysis strategy	3
1.2 Turris	4
1.3 COMPAL CH7465LG (UPC)	11
1.4 Summary	15
1.5 Conclusion	17
2 Analysis of the current solution	19
2.1 Hardware	19
2.2 Software architecture	21
2.3 Plugins	28
2.4 Conclusion	31
3 Design	33
3.1 Analysis of requirements	33
3.2 Technologies	39
3.3 Architecture	42
3.4 Plugins system	46
3.5 UI/UX Design	47
3.6 Conclusion	51
4 Implementation	53
4.1 Development environment	53
4.2 reForis project	56

4.3	Backend	56
4.4	Frontend	63
4.5	JavaScript bundler	63
4.6	Plugins system and demo plugin	71
4.7	Localization	72
4.8	Documentation	74
4.9	Deployment	74
5	Testing	79
5.1	Backend	79
5.2	Frontend	80
5.3	Integration tests	82
5.4	Usability testing	83
5.5	Use test with functional application	84
	Conclusion	89
	Conslusion	89
	Bibliography	91
A	Original Foris screenshots	99
B	Current Foris class diagrams	113
C	Wireframes	117
D	Hi-fi prototype screenshots	129
E	API endpoints	141
F	reForis web interface screenshots	147
G	Usability testing script and quizzes	163
	G.1 Quiz before test	163
	G.2 Quiz after test	163
	G.3 Script	164
	G.4 Moderator script	164
H	Acronyms	167
I	Contents of enclosed microSD card	169

List of Figures

1.1	<i>Screenshots of LuCI</i>	6
1.2	<i>Screenshot of the COMPAL CH7465LG web interface.</i>	12
2.1	<i>Simplified architecture schema of Foris.</i>	22
2.2	<i>Authentication and session management sequence diagram.</i>	29
3.1	<i>Use case diagram of UC-10: Reboot via notification</i>	38
3.2	<i>Use case diagram of UC-11: Reboot via notification</i>	38
3.3	<i>Architecture design of the future Foris application.</i>	43
3.4	<i>Authentication and session management sequence diagram.</i>	45
3.5	<i>Notifications activity diagram.</i>	50
3.6	<i>WAN settings activity diagram.</i>	50
4.1	<i>Screenshot of the PyCharm deployment settings.</i>	54
4.2	<i>reForis project directory structure.</i>	57
4.3	<i>reForis Flask application directory structure.</i>	57
4.4	<i>reForis React application directory structure.</i>	65
4.5	<i>Reboot handling sequence diagram.</i>	67
A.1	<i>Original Foris. Screenshot of the login page.</i>	99
A.2	<i>Original Foris. Screenshot of the notifications page.</i>	100
A.3	<i>Original Foris. Screenshot of the password settings page.</i>	101
A.4	<i>Original Foris. Screenshot of the network interfaces page.</i>	102
A.5	<i>Original Foris. Screenshot of the WAN configuration page.</i>	103
A.6	<i>Original Foris. Screenshot of the LAN configuration page.</i>	104
A.7	<i>Original Foris. Screenshot of the guest network configuration page.</i>	105
A.8	<i>Original Foris. Screenshot of the DNS configuration page.</i>	106
A.9	<i>Original Foris. Screenshot of the Wi-Fi configuration page.</i>	107
A.10	<i>Original Foris. Screenshot of the region and time settings page.</i>	108
A.11	<i>Original Foris. Screenshot of the administration page.</i>	109
A.12	<i>Original Foris. Screenshot of the updates settings page.</i>	110

A.13	<i>Original Foris. Screenshot of the “About” information page.</i>	111
B.1	<i>Config handler class diagram¹</i>	114
B.2	<i>Foris forms class diagram.</i>	115
B.3	<i>Foris form fields class diagram.</i>	116
C.1	<i>Wireframe of the login page.</i>	118
C.2	<i>Wireframe of the dashboard “Home” page.</i>	119
C.3	<i>Wireframe of the notifications dropdown menu.</i>	120
C.4	<i>Wireframe of the notifications page.</i>	121
C.5	<i>Wireframe of the WAN configuration page.</i>	122
C.6	<i>Wireframe of the LAN configuration page.</i>	123
C.7	<i>Wireframe of the Wi-Fi configuration page.</i>	124
C.8	<i>Wireframe of the administration page.</i>	125
C.9	<i>Wireframe of the updates settings page.</i>	126
C.10	<i>Wireframe of the backups creation and restoration page.</i>	127
C.11	<i>Wireframe of the “About” information page.</i>	128
D.1	<i>Hi-fi prototype. Screenshot of the login page.</i>	129
D.2	<i>Hi-fi prototype. Screenshot of the dashboard (home) page.</i>	130
D.3	<i>Hi-fi prototype. Screenshot of the notifications dropdown menu.</i>	131
D.4	<i>Hi-fi prototype. Screenshot of the notifications page.</i>	132
D.5	<i>Hi-fi prototype. Screenshot of the WAN configuration page.</i>	133
D.6	<i>Hi-fi prototype. Screenshot of the LAN configuration page.</i>	134
D.7	<i>Hi-fi prototype. Screenshot of the Wi-Fi configuration page.</i>	135
D.8	<i>Hi-fi prototype. Screenshot of the administration page.</i>	136
D.9	<i>Hi-fi prototype. Screenshot of the updates settings page.</i>	137
D.10	<i>Hi-fi prototype. Screenshot of the backups creation and restoration page.</i>	138
D.11	<i>Hi-fi prototype. Screenshot of the “About” information page.</i>	139
F.1	<i>reForis. Screenshot of the login page.</i>	147
F.2	<i>reForis. Screenshot of the overview page.</i>	148
F.3	<i>reForis. Screenshot of the notifications page.</i>	149
F.4	<i>reForis. Screenshot of the notifications page.</i>	150
F.5	<i>reForis. Screenshot of the WAN configuration page.</i>	151
F.6	<i>reForis. Screenshot of the LAN configuration page.</i>	152
F.7	<i>reForis. Screenshot of the Wi-Fi configuration page.</i>	153
F.8	<i>reForis. Screenshot of the network interfaces page.</i>	154
F.9	<i>reForis. Screenshot of the guest network configuration page.</i>	155
F.10	<i>reForis. Screenshot of the DNS configuration page.</i>	156
F.11	<i>reForis. Screenshot of the password settings page.</i>	157
F.12	<i>reForis. Screenshot of the region and time settings page.</i>	158
F.13	<i>reForis. Screenshot of the reboot device page.</i>	159
F.14	<i>reForis. Screenshot of the updates settings page.</i>	160

F.15 <i>reForis</i> . Screenshot of the packages settings page.	161
F.16 <i>reForis</i> . Screenshot of the "About" information page.	162

Listings

2.1	Example of <code>ubus</code> objects list.	26
2.2	Example of <code>ubus</code> objects methods.	27
2.3	Example of <code>foris-client</code> commands.	28
2.4	Example of a <code>foris-controller</code> module JSON Schema.	30
4.5	Example of <code>.gitlab-ci.yml</code> configuration file.	55
4.6	Application factory code example.	58
4.7	Wi-Fi view definition.	59
4.8	Login view code example.	60
4.9	Administration view code.	61
4.10	Loading additional <code>tzinfo</code> translations in a template.	61
4.11	Translation of the typical <code>foris-controller</code> setting module to the API endpoint.	62
4.12	The code example of getting and checking session stored in a filesystem.	63
4.13	<code>useAPIGet</code> hook code example.	66
4.14	Using <code>ForisForm</code> container with LAN form code example.	69
4.15	<code>LANForm</code> code example.	70
4.16	The plugin registration with <code>entry_points</code>	71
4.17	The plugin load via <code>entry_points</code>	72
4.18	The example of loading the translations.	73
4.19	OpenWrt Makefile sample.	76
5.20	Coverage report of the Python code testing.	80
5.21	Coverage report of the JS code testing.	82

List of Tables

1.1	The hardware comparison table.	16
1.2	The interfaces functionality comparison table.	16
1.3	The interfaces usability heuristic comparison table.	17
2.1	Hardware details of the Turrus routers version 1.0 and 1.1	20
2.2	Hardware details of the Turrus Omnia router	20
3.1	The traceability matrix of requirements coverage.	40

Introduction

This master thesis deals with the redesign of the Foris configuration interface of the Turris project developed by the CZ.NIC association.

Background

Turris routers run on TurrisOS which is an operating system for routers based on OpenWRT [1]. Despite OpenWRT already having LuCI [2] configuration interface which is available out-of-the-box, Turris team has decided to develop its own web administration interface due to complexity of the existing LuCI interface for basic users.

Thus, came into being Foris, which is a simplified interface for administration requirements of Turris routers. It has been created in order to provide users with a simple and extensible administration and network configuration tool.

Motivation

The history of Foris begins in October, 2013 when Jan Čermák posted the first commit ² to the Foris repository. Turris project had started as a startup and gave birth to Foris which has grown very rapidly. During the process, a lot of changes and new features were added to the existing code base without much planing for future maintenance.

Thus, modern-day Foris has a lot of legacy code and technologies used. Foris is an open-source project and it has an extensible architecture in order to allow third-party developers and enthusiasts to create plugins to meet their requirements. Unfortunately, developers are not very interested in contributing to the project with legacy code and unpopular technologies used in it.

² A curious reader can find the first commit message on the CZ.NIC Labs GitLab [3].

INTRODUCTION

The new redesigned Foris interface is made using modern technology and a simple plugin system. Thus, it intends to rectify the shortcomings described above in order to attract third-party developers.

Analysis of similar solutions

Analysis of existing solutions is essential for this work because it helps to get an overview of the modern state of the domain and also alleviate possible issues in subsequent design and implementation.

This chapter analyzes and compares current competing solutions to identify common mistakes in making a router web interface. The analysis is also focused on UI and UX design, trying to fix issues of the current interface and taking into account both problems and benefits of the reviewed interfaces.

1.1 Analysis strategy

This section describes the principles of the following analysis of the routers and their web configuration interfaces. The aspects of the solutions, which this analysis is focused on, are described in the following subsections.

1.1.1 Hardware

Specification

Allows to find out if router has any extra features (such as additional ports or swappable storage etc.).

Expansibility

the evaluation of expansibility of a device based on its specification.

1.1.2 Software

Operating system

Specification of a router OS or a software version. The number of features provided by a router is directly dependent on the OS used.

Configuration web interface

Functionality

Configuration features of a web interface entirely depend on the router functionality.

Usability

The testing of usability of an interface based on heuristic evaluation consisting of ten individual heuristics described by Jakob Nielsen [4]:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose, and recover from errors
- Help and documentation

It can help to identify problematic areas in the domain and prevent mistakes in future design.

Extensibility

This part of the analysis evaluates the possibility to extend the interface with plugins. Also, this part is trying to assess how large is the stack of technologies a developer must be familiar with in order to create a plugin. It will help us to understand whether it is a common approach or something more specific to this solution.

The result of the comparative part of the analysis is summarized in a few tables in the last section of this chapter on page 15.

The issue with the examination of these routers and their interfaces is that the author of this paper does not have them at his disposal. Fortunately, some of these solutions are open-source, and the author is allowed to analyze them closely. It holds true especially for current implementations of Turrís routers which are analyzed in the next section.

1.2 Turrís

Even though the redesign of the web configuration interface of Turrís routers is described in detail in the next chapter on page 19, it is also analyzed in this one with references to another chapter where it is compared with other solutions on the market.

Turris project is developed by the CZ.NIC association and described on the official Turris web page [5] as a service that utilizes a specialized router to help its users protect their home networks from intruders.

1.2.1 Hardware

Presently Turris offers two different models of its router with one more being in the development stage. Hardware specification of these models is described in the chapter 2 on page 19.

1.2.1.1 Expansibility and upgrades of the hardware configuration

All models of Turris routers can more-or-less be upgraded by a user. Replacement of Wi-Fi modules and mounting of additional storage in Turris and Turris Omnia models is possible thanks to mini PCI slots. Moreover, Turris Omnia offers to add an LTE modem [6]. MOX can be assembled from seven different modules, this creates a massive diversity of the configurations.

In light of the above, design and realization of this work should take into account different hardware configurations. It is discussed in more detail in the “Design” chapter on page 33.

1.2.2 Software

1.2.2.1 Operating system

All represented models of routers use Turris OS as an operating system. The source code of the Turris OS is available in the CZ.NIC labs git repository [7] under GNU GPLv2 license. Turris OS is the Linux distribution based on OpenWrt [1]. Unlike OpenWrt, it comes with features like automatic updates and offers a simpler web interface. Simply said, Turris OS attempts to be more user-friendly than OpenWrt. Fortunately, all these operating systems are open-source, so it’s possible to study this software directly.

1.2.2.2 Web configuration interfaces

Turris OS has two configuration interfaces. It comes with LuCI configuration interface [2] because it’s based on OpenWrt. Moreover, Foris configuration interface is also present in the Turris routers as part of the Turris OS. Both of these interfaces are studied in the following sections.

1.2.2.3 LuCI

LuCI is open-source software which is available on the OpenWrt organization GitHub [2] under Apache License 2.0.

LuCI uses UCI (Unified Configuration Interface) [8] which is meant to centralize the configuration of OpenWrt. UCI configuration files can be modified

1. ANALYSIS OF SIMILAR SOLUTIONS

through various programming APIs (like Shell, Lua and C), which are used by LuCI to make changes to the UCI files [9].

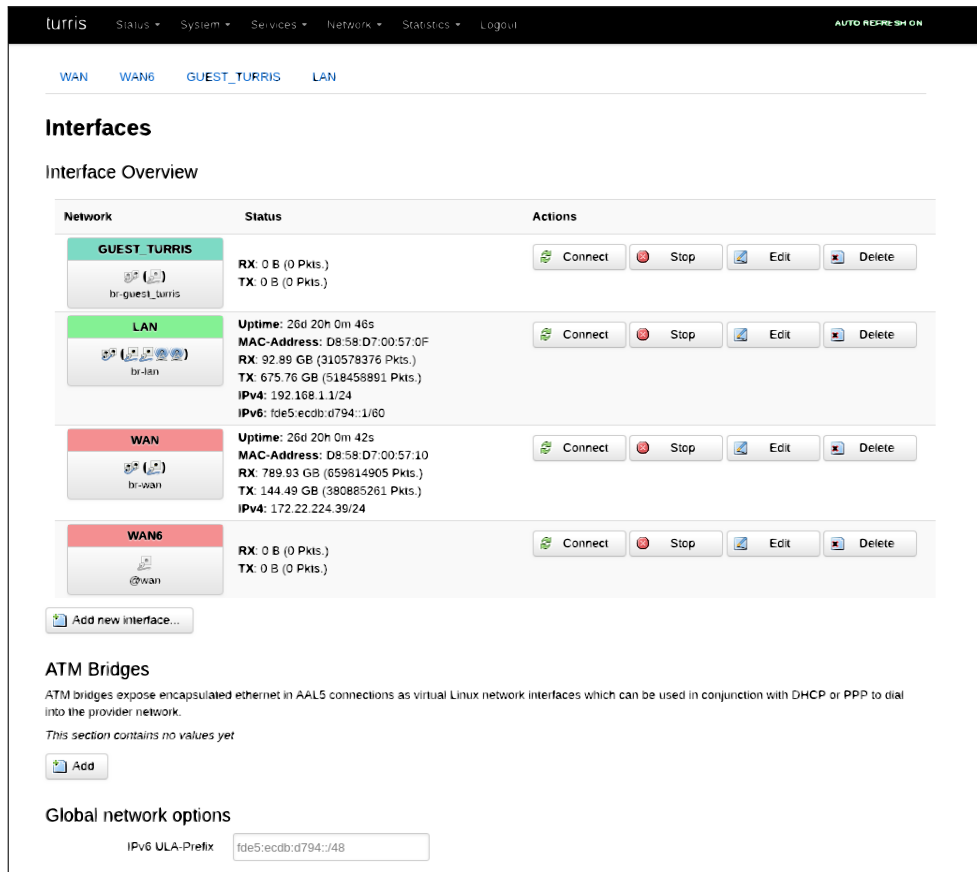


Figure 1.1: Screenshots of LuCI

Functionality

LuCI is a powerful and exhaustive configuration interface. This interface can be considered to be a step forward from setting up a router via configuration files through the command line to doing it via a web interface. Due to this fact, it attempts to cover all configuration possibilities which can be enabled using config files ³.

Usability

Visibility of system status An issue was found with some pages containing a long list of items that can be added or modified. After a user adds

³The whole list of supported configuration files is listed on the OpenWRT wiki page [10].

an item, he is led to believe that configuration was applied to the system. In reality it was not, because there is a button at the bottom of the page labeled **Save & Apply** which should be clicked to apply the setting. If the list of items is too long, then the button is hidden under the scroll bar. It confused the author a few times during the process of adding of firewall rules.

Match between system and the real world The terms match real world terms as closely as possible in the case of routers configuration interface. The configuration terms are typical for networks domain, even though they may not be known to non-advanced users.

User control and freedom Changes made by a user are not saved until the user clicks the button **Save&Apply**.

Consistency and standards Issues were not found.

Error prevention Bizarre behavior was detected. The forms provide validation of IP addresses and highlight a violating field in red if an IP address doesn't match the correct format. But despite this, a user can still submit a form and changes may not be applied. A user doesn't get any error message in this case.

Some fields don't have any validation at all, so it's possible to fill in some meaningless value and it will be saved (e.g., "Start" in DHCP Server settings may be some really big number overflowing the range of IP addresses).

Recognition rather than recall This partially overlaps with the first heuristic. A user should remember that he made some changes and should not forget to scroll to the bottom of the page and apply them. Other issues weren't found.

Flexibility and efficiency of use The interface is neither flexible nor adaptable for both beginners and advanced users. During the study there could not be found a way to adapt the interface in any way. The interface is more suitable for advanced users, and it doesn't have any wizards to provide a beginner user with a fast way to configure routers.

Aesthetic and minimalist design Menu items or tabs split most of the pages but there is still a lot of information on single column pages.

Long lists preceding short forms make it even worse. On some pages, a user should scroll through a long list in its entirety to find the information he needs. It's better to show only a few list items and expand it if needed.

Help users recognize, diagnose, and recover from errors The same issue as described in “Error prevention”.

Help and documentation Help texts exist, but they are very few and sparse. They are not sufficient for inexperienced users, because even some abbreviations are not explained (e.g., DHCP, DNS, MTU, etc.).

Conclusion Authors of the LuCI consider LuCI to be a clean interface [2]. This is true when compared to configuration file changes being done via a console. But LuCI is overloaded with a lot of configuration elements and has an outdated design. Simply said, it’s better suited for advanced users.

Such a solution poses a challenge to average users who seek an answer to the problem of a well-working home network. It provides an exhaustive amount of configuration options in a way of a direct modification of the configuration files but it’s not used by all users. It also doesn’t have any kind of an initial configuration wizard which would allow a user to make only a few necessary adjustments to get a fully-working router.

Extensibility LuCI is extensible via so-called modules. Process of module creation is well-documented on LuCI GitHub wiki pages [11].

Modules are written in Lua programming language [12] using luci API. LuCI has a simple template processor which parses HTML files into Lua functions and allows to store precompiled template files [13]. LuCI API also provides “CBI models” which allow to create forms based on a user interface and save their contents to a specific UCI config file using just the definition of the models’ structure.

Each module is distributed as an OpenWrt package and should follow the strictly predefined source code structure.

LuCI has a really thorough and well-documented way to extend it. Unfortunately, it forces developers to study a set of obscure technology. This can be an issue with attracting new developers to the community.

1.2.2.4 Foris

Foris is used by Turris routers as the current solution to provide users with a basic configuration interface. Redesign of current Foris interface is the main goal of this paper, so this solution is analyzed in detail in the chapter 2.

A set of original Foris web interface screenshots is available in the appendix section on 99 page.

Functionality Foris allows users to perform basic router configuration. This web interface includes all the settings needed to cover all of the most popular scenarios used by an average user, including network settings, update management, and maintenance.

It's possible to install plugins (e.g., storage management, VPN server, etc.), which can extend functionality to meet custom demands of an individual user. The plugin system of Foris is described in more detail in the paragraph on the extensibility on page 10.

On top of that, Foris provides first configuration wizard, which guides the user through the necessary steps to have a fully-working router.

Usability

Visibility of system status It has an issue similar to the previous interface.

A user should press the "Save" button to apply the settings change, but in some long forms the button is hidden under the scroll bar, and users may not notice that they should submit changes. Moreover, changed fields are not marked as such - a user should remember the changes he did. Even though forms in Foris are shorter than in LuCI, their length may still cause issues.

Match between system and the real world The terms match real world terms as close as possible in the case of routers configuration interface.

User control and freedom A user can simply quit the page if he makes an error in some of the fields. Also, there is a "Discard changes" button on every single form. Despite the button's main purpose being to reset the page, additionally it provides one more way to perform the "undo" action.

Consistency and standards Some inappropriate location of the settings was found in the first evaluation which is described in the next part of the usability analysis (e.g., updates settings and notifications settings on the maintenance page).

Error prevention The interface has a rich validation of fields where it's needed. Also, user input is limited by choices where it's possible. No outstanding issues were found.

Recognition rather than recall A user should remember that he made some changes and that they need to be applied.

Flexibility and efficiency of use The interface provides first configuration wizard which allows inexperienced users to get a working solutions with just a few easy steps. There is also a way to skip it for advanced users (user may want to just restore configurations from a backup).

Aesthetic and minimalist design The pages contains only required information except for a few cases defined under "Consistency and standards".

Help users recognize, diagnose, and recover from errors The interface provides WAN and DNS connection tests, which allow the user to perform configuration diagnostic.

Help and documentation Every page has a rich help text. Some fields which may be unclear have toggled help text. The author's subjective view is that this is enough to satisfy both beginners' and advanced users' needs.

MI-NUR The first analysis of the actual Foris interface was performed as part of the final project for the MI-NUR course ⁴. Here is the summary of this evaluation.

- Advantages
 - Design of the interface is simple and clear.
 - It has a wizard with simple first configuration.
 - Language switch is intuitively located on each page.
 - The web interface has a responsive and modern design.
 - The interface provides real-time notifications without the need to refresh the page.
- Disadvantages
 - In the first configuration guide, a user should click on the next highlighted menu item and doesn't get transferred automatically.
 - "Automatic restarts after software updates" is located under the "Maintenance" section and not under the "Updater" section which is implied by the setting's name.
 - "Maintenance" also has "Enable notification" setting. It can confuse a user because there is already a "Notification" menu item present.
 - User can't quickly check notifications from a random page. They are hidden under a special menu item.

Extensibility Foris architecture supports functionality extension with plugins. This extension system of Foris is described in detail in the chapter 2 on page 28 and is summarized here.

The problem of this solution is that the technologies which developer should use to implement a frontend part of the plugin are neither widespread

⁴ The evaluation was made by the following team: Bodnar, B., Karola, A., Kryvosheienko, M., Laskov, B. and Samigullina, G.

nor standardized. Particularly, using custom forms (Bottle framework doesn't have those) and HTML elements instead of using existing, well-known and widespread technologies like some other framework with built-in forms or existing toolkits that work with HTML, JS, and CSS.

All these issues discourage the community of developers from writing new Foris plugins. Author of this paper is trying to solve this issue in the chapter 3.

1.3 COMPAL CH7465LG (UPC)

UPC is one of the most popular internet providers in Europe. So UPC modems represent a wide-spread solution for homes and small offices.

Even though CH7465LG is a modem it has router functionality as well. So this model can be analyzed and compared with another product in this section [14].

1.3.1 Hardware

1.3.1.1 Specification

The following summarized specification was described in detail in the security evaluation performed by Search-LAB [15].

1.3.1.2 Components

- Intel XScale CE26XX processor
- All-Flash TSOP-48 1Gbit parallel NAND flash chip
- PHISON eMMC flash chip
- HIGH PERFORMANCE 2Gbit DDR3-1600 SDRAM 8 BANKS X 16Mbit X 16
- 4+1 ethernet ports
- dual band 3x3 802.11ac single chip with 1300Mbps PHY rate support and
- SMD QFN56 GP 802.11B/G/N Wi-Fi chip

1.3.1.3 Interfaces

- 3 control leds (behind a plastic cover)
- WPS button
- 2 RJ11 telephone connectors

1. ANALYSIS OF SIMILAR SOLUTIONS

- 4 RJ45 Ethernet connectors
- Reset button
- RF cable input
- DC input
- Power on switch

1.3.2 Expansibility

This model of the modem doesn't allow to extend it with additional hardware or change hardware configuration.

1.3.3 Software

Software version of the sample is CH7465LG-NCIP-6.12.18.24-5p4-NOSH.

1.3.3.1 Web configuration interfaces

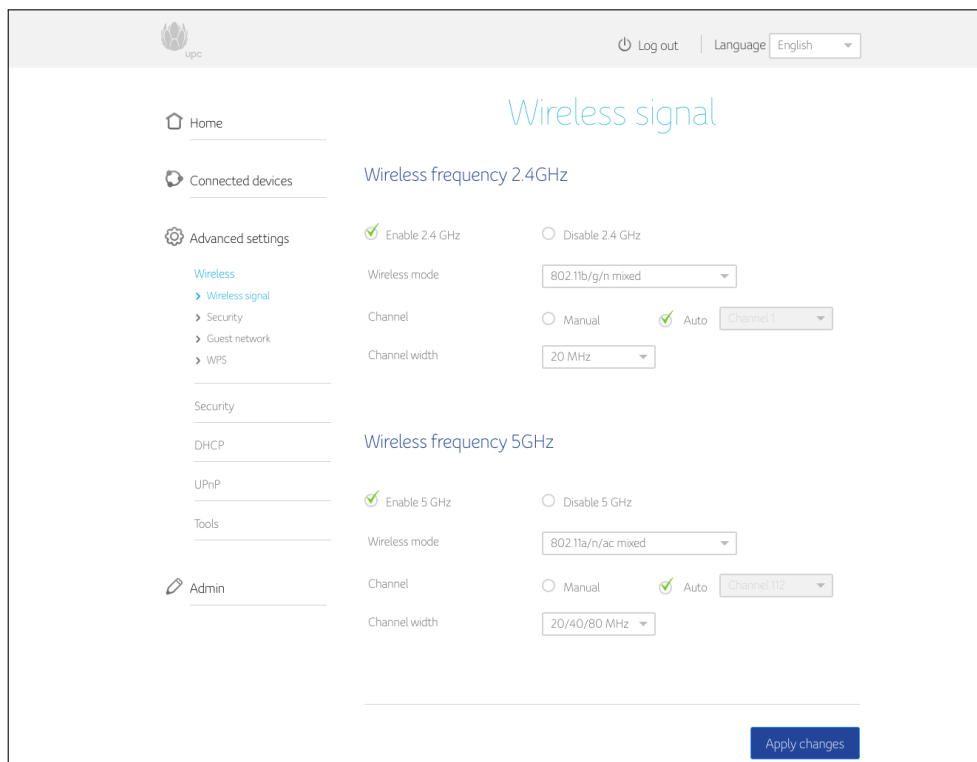


Figure 1.2: Screenshot of the COMPAL CH7465LG web interface.

Functionality Following list of web interface functions was discovered during analysis. The functions are grouped by configuration pages.

- Home
 - Quick Set-Up Wizards
 - * Configuration of Wi-Fi network
 - * Network diagnostics tools
 - * Guest network configuration
 - Wireless Gateway status overview
 - Wireless connected devices list
 - Ethernet connected devices list
- Connected devices
 - List of all connected devices with names, “connected to” interfaces, MAC addressees, IP addressees and connection speeds.
- Advanced settings
 - Wireless
 - * Wireless signal
 - * Security
 - * Guest network
 - * WPS
 - Security settings
 - * Firewall
 - * MAC filtering
 - * IP and Port filtering
 - DHCP
 - UPnP
 - Tools
 - * Network status
 - * Ping
 - * Traceroute
 - * MTU size
- Admin
 - Change password
 - Reload and Reboot

1. ANALYSIS OF SIMILAR SOLUTIONS

- Remote access settings
- Device info

This model of router and its web configuration interface is intended mostly for average users, and it covers all basic home needs. The interface has 3 wizards for a quick router setup which may be useful. It also has a well-designed dashboard with list of connected devices.

As a result of the analysis, no LAN port settings were detected. The web interface doesn't allow to set up the IP address of the router in a LAN network.

Usability

Visibility of system status The locations of the settings are hidden by JS functionality and are not visible in URL. Thus it's not possible to save a page with a location of a setting as a link, and also it can be problematic for users who are used to quickly navigating a website using the URL path.

During network diagnostics "Telephone Service" is marked with a green check mark even when it's not being used. Changes in the settings are not highlighted, but a user is warned before he quits the page without saving changes first.

Match between system and the real world Issues weren't found.

User control and freedom Undo button was not found in the interface. If a user does not want to apply settings, then he should not press the "Apply changes" button.

Consistency and standards Some settings are not properly located and may confuse a user (e.g., "Backup" setting is located in the "Reload and Reboot" section) The "Wi-Fi" term is used alongside "Wireless".

Error prevention Wi-Fi password fields validation is performed after clicking "Apply changes" button.

Some fields demand the data to conform to a certain format upon saving (e.g., MAC address) and some (e.g. IPv6) are validated in real time.

Recognition rather than recall A user has to remember that he made changes in any of the interface forms and that these changes should be applied. Issues were not found in other cases.

Flexibility and efficiency of use The interface covers both beginner and advanced user groups' needs. But all network configurations are hidden under the "Advanced configuration" menu item, that may confuse more advanced users.

Aesthetic and minimalist design The interface has a minimalistic design. The configuration pages are split properly, and they are not overflowing with useless information.

Help users recognize, diagnose, and recover from errors The interface comes with a connection troubleshooting tool on the front page. So, a user can run diagnostics of the network connections and apply fixes if necessary.

Help and documentation In every router configuration interface there are a lot of domain-specific terms which should be accompanied by description or a help page. In the investigated web interface there are a lot of help texts, but they are still lacking in some places (e.g., wireless modes, firewall options, UPnP, MTU)

Other issues The password input fields are visible and not rendered as a password. It may cause security issues, as someone behind your shoulder can read this password. Also, these fields are not recognized by browsers as a password input thus not taking advantage of the autofill feature.

Lags and freezes after moving between configuration sections were found during heuristic evaluations.

Conclusion The web interface has a modern and pleasant minimalistic design. It is focused on satisfying users with not very advanced, but rather basic needs. However, fixing discovered issues would make this interface even better.

1.4 Summary

1.4.1 Hardware features

The following table Table 1.1 considers only hardware features which may impact a web interface.

From this table, the reader can see that the analysis doesn't concern itself with any router with hardware extension possibilities. Very few routers provided on the market allow a user to change their hardware configuration due to not being popular among average users. Thus, the author hasn't got any for the analysis.

1.4.2 Software

1.4.2.1 Functionality

The definition of the advanced configuration doesn't have certain criteria. It makes difficult to categorize interface by suitability for advanced users or be-

HW feature	Router	Turris 1.0	Turris 1.1	Turris Omnia	COMPAL CH7465JLG
USB	✓	✓	✓	✓	✗
miniPCI	✓	✓	✓	✓	✗
Exchangeable Wi-Fi module	✓	✓	✓	✓	✗
SIM slot	✗	✓	✓	✓	✗
Extensibility	✓	✓	✓	✓	✗

Table 1.1: The hardware comparison table.

SW feature	Interface	LuCI	Foris	UPC
Advanced configuration	✓	✗	✗	✗
Initial configuration wizard	✗	✓	✓	✓
Extensibility (plugins)	✓	✓	✓	✗

Table 1.2: The interfaces functionality comparison table.

gainers. The author offers the following simple categorization: If the interface affords to perform almost all theoretically possible network configurations provided by router then it's suitable for advanced users. If, on the other hand, the interface covers a smaller subset of router configuration parameters in order to be more clean and understandable for beginners, then we consider the interface to provide only basic settings.

The functionality comparison is summarized in the table Table 1.2.

1.4.2.2 Usability

The usability comparison is based on the previous heuristic evaluations and it's summarized in the table Table 1.3.

The most critical and common usability issues found during heuristic evaluations of all the considered interfaces are summarized in the following list:

- No visible status of configuration changes. This also causes the next issue.
- Changes are not preserved when a user accidentally leaves the page.

Heuristic	Interface		
	LuCI	Foris	UPC
Visibility of system status	●	●	●
Match between system and the real world	●	●	●
User control and freedom	●	●	●
Consistency and standards	●	●	●
Error prevention	●	●	●
Recognition rather than recall	●	●	●
Flexibility and efficiency of use	●	●	●
Aesthetic and minimalist design	●	●	●
Help users recognize, diagnose, and recover from errors	●	●	●
Help and documentation	●	●	●

Legend:

- Excellent solution – ●
- Without issues – ●
- Minor issues – ●
- Significant issues – ●

Table 1.3: The interfaces usability heuristic comparison table.

- Confusing categorization of the settings.
- Neither beginner nor expert users' needs are fully met.

1.5 Conclusion

The analysis helped to reveal the most common and major issues and limitations of the current solutions, especially in regards of UI and UX design. Moreover, during the analysis stage the author has done a deep dive into the domain of router configuration interfaces. This has provided him with an opportunity to approach the problem at hand from different angles: both from developer's and user's point of view.

All of the exposed issues will be taken into account during the subsequent redesign and implementation parts of the project. The author will afford to keep all the advantages of the existing solution and to fix existed limitations.

Analysis of the current solution

The main goal of the analysis is to familiarize ourselves with the technologies used and the architecture of the Foris interface. Another goal is an investigation for weak architecture points and avoiding them in the subsequent design and implementation. Also, the research has to focus on plugin development which is essential to design and implementation of a better pluggable architecture.

2.1 Hardware

Turrís team has developed two models of the routers, and one model is currently in the development stage.

- Project Turrís (1.0 and 1.1)
- Turrís Omnia
- Turrís MOX (development stage)

Turrís routers have open-source hardware design. Hardware schemas and documentation are located on a documentation web page of the Turrís project [16].

2.1.1 Project Turrís

Turrís, the first model of Turrís routers. It was rented to selected users for a symbolic price of one Czech crown to aid research.

The hardware configuration of these routers is described in the Table 2.1. Information in the following table is taken from the official Turrís web page [17], and additional information about the Turrís router is available there.

2. ANALYSIS OF THE CURRENT SOLUTION

	Turriss 1.0	Turriss 1.1
Processor	Processor Freescale P2020, 1200 MHz	
RAM	2 GB DDR3 RAM in a SO-DIMM slot	
Storage	16 MB NOR and 256 MB NAND flash memory	
Ethernet	Dedicated gigabit WAN and 5 gigabit LAN ports	
Wi-Fi	802.11a/b/g/n with 3x3 MIMO and removable external antennas	
USB 2.0	2x on the back side	
USB 3.0	no	1x on front side + 2x internally
miniPCIe	1 free slot, 1 slot occupied by Wi-Fi card	
Other interfaces	UART, SPI and I2C connected to a pin-header for easy customization, debug console over internal microUSB port	
SIM slot	no	yes
Power source	7,5 V	12 V
Power consumption	9.5 W without load, 12.5 W with CPU load and 14 W with maximum wired and Wi-Fi network load. Measured power consumption includes the supplied power adapter.	

Table 2.1: Hardware details of the Turriss routers version 1.0 and 1.1

CPU	1.6 GHz dual-core ARM
RAM	1 GB DDR3 (optionally 2 GB)
Storage	8 GB flash
LAN	5× Gbit port
WAN	1× Gbit port
SFP	yes
USB	2× USB 3.0
Mini PCI Express	2×
mSATA / mini PCI Express	1×
Wi-Fi (mini PCIe)	3×3 MIMO 802.11ac 2×2 MIMO 802.11b/g/n

Table 2.2: Hardware details of the Turriss Omnia router

2.1.2 Turriss Omnia

Turriss Omnia is the second generation of Turriss router. Changes from the previous version are not so significant but noticeable, they affect the design of the device as well as hardware. The whole list of hardware feature is available on the Turriss Omnia webpage [18] and also in the following Table 2.2.

2.1.3 Mox

The latest generation of the Turrís routers at the moment and it's in the development stage as of April 2019. MOX is the first modular router in the world which allows users to build their own set of modules which are appropriate to their use-cases. Seven different modules will be available to build a custom router. The whole list of the modules and their configurations is available on the MOX modules specification page [19].

This paper isn't focused on the MOX model but it should consider it in the final design of the interface, for it to be flexible towards this configuration diversity. Another thing which is significant in the scope of this work is a future possibility to make the so-called mesh Wi-Fi routers system. It will offer users the ability to configure the whole network using a single interface, although it can entail connectivity issues.

2.2 Software architecture

Simplified architecture schema of Foris is shown in the Figure 2.1.

2.2.1 Backend

A backend is a set of services, uci configurations, scripts, and notifications. All these resources are covered by `foris-controller` as a facade. So frontend has just one communication point with the backend.

`foris-controller` is open-source software, which is available on the CZ.NIC gitlab [20] under GNU GPLv3 licence.

`foris-controller` does the following:

- Gets requests on the bus.
- Processes requests via backends.
- Sends response to the bus.
- Sends notifications to the bus.
- Validates messages against a JSON-schema.

Unfortunately, the `foris-controller` interface is not well-documented. But it validates inputs against JSON-schema which is self-documented. Each module of the `foris-controller` uses a relevant JSON-schema.

2.2.2 Frontend

Web application and WebSockets server represent a frontend.

2. ANALYSIS OF THE CURRENT SOLUTION

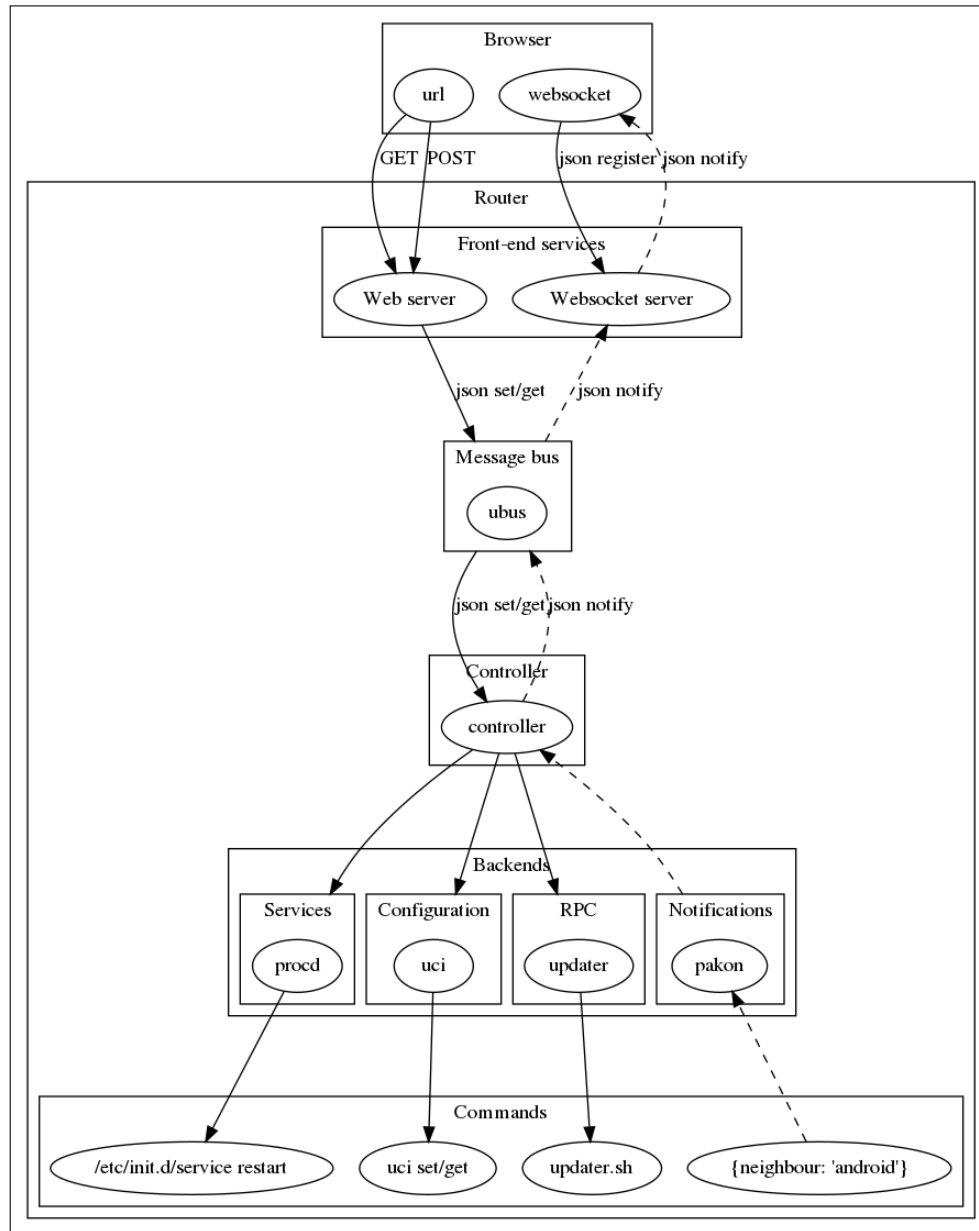


Figure 2.1: *Simplified architecture schema of Foris.*

2.2.2.1 Web server

The web application is written in Python [21] language using `bottle` [22] framework. And it runs with `lighttpd` server[23].

Foris is open-source and all the code described below is available in the official Turris repository [24] with GNU GPLv3 licence.

Config handlers The code is split by modules into so-called `config_handlers`. Every `config_handler` is the Python class which inherits `BaseConfigHandler` and defines `get_form()` method. The method has the following responsibilities:

- Getting configuration data from the backend (`foris-controller`) and preparing it to be used in a form.
- Preparing data to have a relevant structure and sending data to the backend.
- Creation of a form with fields and validation.

List of the config handlers:

- `DNSHandler`
- `GuestHandler`
- `GuideFinishedHandler`
- `LanHandler`
- `MaintenanceHandler`
- `NetworksHandler`
- `NotificationsHandler`
- `PasswordHandler`
- `ProfileHandler`
- `RemoteHandler`
- `UnifiedTimeHandler`
- `UpdaterHandler`
- `WanHandler`
- `WifiHandler`

The UML diagram of the some of the config handlers is available in the appendix on page 114.

2. ANALYSIS OF THE CURRENT SOLUTION

Forms Foris forms don't use any existing libraries and are written from scratch. The form represents an object of the `fapi.ForisForm` class. This class encapsulate data preparation, validation, caching, callbacks calling, error collecting (and rendering).

Templates HTML code is dynamically generated via Jinja2 template language [25].

Styles None of the toolkits for developing HTML documents with styles are used. The styles are written from scratch via SASS, and it describes custom elements styles.

Nowadays there exist a lot of toolkits which make developing arrangement and styling of components easier and more maintainable but unfortunately none of them are used.

Client-side code The client-side code is represented by dynamically generated (via Jinja2) JS code and static code.

The dynamically generated JavaScript code is used to make translated messages which are used in dynamic components such as messages (e.g., pop-ups and validation errors). This solution has one consequence. Every translated text should be generated in one place and can't be defined in the place where it's used.

The static JavaScript code is represented by one file, and it's in charge of AJAX calls, WebSockets, rendering errors and warning messages (pop-ups). Also, in this solution `vex` [26] a library is used that provides interactive pop-up dialogs.

This approach also has consequences, there is a lot of logic contained in the one long file which makes it difficult to maintain and add new features. Also, this solution doesn't use minification [27], which can optimize code and decrease the amount of data transferred to the client.

Conclusion The set of technology has a few design issues which has prompted this work to be started. `bottle` framework is a good choice to make some simple web interface like Foris used to be at the beginning. But after new functionality was added, a lack of features provided by the framework was discovered (forms support, sessions management etc.). It resulted in the need to implement such features on one's own. This decision led to an increase in the lines of code and obstructions emerging in the way of maintaining and adding new features.

Also, the things which were written from scratch (e.g., forms and client-side code) don't split the logic in the best way and should be refactored or redesigned.

2.2.2.2 WebSockets

Another part of the frontend is WebSockets server.

WebSockets is a protocol that allows a persistent TCP connection between server and client so they can exchange data at any time [28]. The main advantage of using WebSockets is that server can stream data without client requests avoiding excess AJAX pulling.

WebSockets server is responsible for the following:

- Gets notifications from the bus.
- Resends received notifications through WebSockets to the client browser.

In other words, WebSockets transfer messages from the bus to the client browser. It allows to update configuration page dynamically and notify browser about configuration changes. It's useful in cases when a user should be notified immediately (e.g. router restart is needed, configuration changes are being applied etc.)

2.2.3 Communication buses

Backend and frontend communicate with each other via communication bus. Now it supports three communication buses:

- `ubus`
- `MQTT`
- `unix-socket`⁵

2.2.3.1 `ubus`

Currently Foris uses `ubus` [29] to accommodate communication between frontend and backend. `Ubus` is the communication interface which allows relaying messaging between processes and services, which is widely used as the main communication channel in OpenWrt.

It's simple to get a list of the all objects provided by `ubus` as shown in Listing 2.1.

⁵`unix-socket` is used only for testing and it is not described in this paper.

2. ANALYSIS OF THE CURRENT SOLUTION

```
root@turris:~# ubus list
dhcp
foris-controller-about
foris-controller-diagnostics
foris-controller-dns
foris-controller-guest
foris-controller-lan
foris-controller-maintain
foris-controller-networks
...
service
session
system
uci
websocket-listen
```

Listing 2.1: Example of `ubus` objects list.

Also, the command line interface of the `ubus` allows listing methods on a certain object as we can see in the Listing 2.2. Notice, that the list of objects methods is similar to JSON file structure.

```
root@turris:~# ubus -v list foris-controller-wifi
'foris-controller-wifi' @30bcb750
"get_settings": {
  "request_id": "String",
  "final": "Boolean",
  "multipart": "Boolean",
  "payload": "Table"
}
"reset": {
  "request_id": "String",
  "final": "Boolean",
  "multipart": "Boolean",
  "payload": "Table"
}
"update_settings": {
  "request_id": "String",
  "final": "Boolean",
  "multipart": "Boolean",
  "payload": "Table"
}
```

Listing 2.2: Example of ubus objects methods.

2.2.3.2 MQTT

There is a plan to switch from `ubus` to MQTT in the future. Currently MQTT is used in the testing branches of Turris OS. MQTT, the M2M/IoT connectivity protocol [30]. So, it allows connecting more devices via a uniform communication channel. It is planned to use in the future to provide a possibility to configure many routers from single administration interface.

MQTT support is implements with Mosquitto [31]. Mosquitto, the open source MQTT broker.

2.2.3.3 foris-client

The support of these buses is provided by `foris-client` library [32] which is used in frontend as abstraction layer on the buses.

```
root@turris:~# foris-client-wrapper -m router_notifications\  
  -a create \  
  -I '{"severity":"news","immediate":false,"msg":"Example."}'  
>>> {"result": true}  
root@turris:~# foris-client-wrapper -m router_notifications\  
  -a list -I '{"lang":"en"}'  
>>> {"notifications": [{  
  "id": "1553543653-7854",  
  "displayed": false,  
  "severity": "news",  
  "created_at": "2019-03-25T20:54:13",  
  "msg": "Example.",  
  "lang": "en"  
}]}
```

Listing 2.3: Example of foris-client commands.

2.2.4 Authentication

The password to the Foris interface is saved in the UCI configs in the form of a hash using 48bit pseudo-random salt internally generated by pbkdf2[33].

The `foris-controller` provides a handler to check if the hash of the received password and the hash stored in the UCI match.

So the Foris web application performs authentication using `foris-client` library which will request `foris-controller` to compare password hashes. Thus, authentication is complete.

2.2.5 Session management

Both the web server and WebSockets server use server-side sessions via `ubus` sessions management. After the client logs in, a new session is created using `ubus` session management. The two session ids created via `ubus` are set as a cookie with `foris.session` and `foris.ws.session` names in the client browser. Thus, in subsequent requests, the web server and WebSockets server can determine if the user has already been authenticated.

Both authentication and session management processes are illustrated via sequence diagram on the Figure 2.2.

2.3 Plugins

Since Foris is composed by frontend and backend, the plugin should contain modules for both these parts. That means that plugins consists of a module

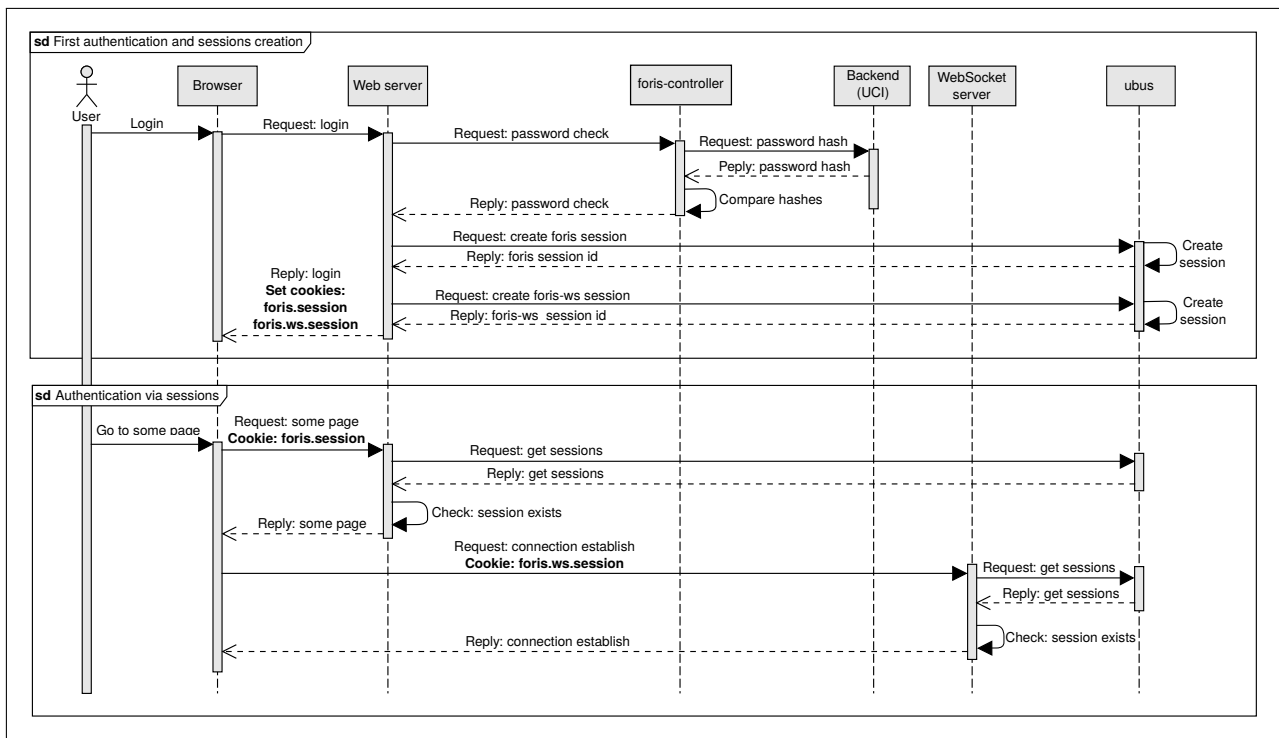


Figure 2.2: Authentication and session management sequence diagram.

for foris-controller and foris (frontend).

2.3.1 Backend part (foris-controller)

Module for foris-controller contains:

Python code gets requests, processes it and responds with results using a bus. It may also send notifications.

JSON schema is a vocabulary that allows you to annotate and validate JSON documents [34]. All requests are described and validated against JSON schema. Example of JSON Schema for foris-controller is shown in the Listing 2.4. More information about JSON Schema validation can be found in the foris-schema library [35].

Tests it is always better to test your code...

```
{
  "oneOf": [{
    "description": "Request to Get number of slices",
    "properties": {
      "module": {"enum": ["sample"]},
      "kind": {"enum": ["request"]},
      "action": {"enum": ["get_slices"]}
    },
    "additionalProperties": false
  },]
}
```

Listing 2.4: Example of a `foris-controller` module JSON Schema.

Python code should contain the routing and handlers:

Routing gets incoming messages and calls appropriate handler.

Handlers get messages from routing and performs any necessary operations (e.g. read and update of UCI configuration, start/restart/stop services, run commands).

It's possible to use prepared wrappers to call backend functions:

cmdline is a wrapper around command line commands.

uci this wrapper allows to read and update UCI configuration.

files wrapper around filesystem.

services wrapper around service management.

2.3.2 Frontend plugin (`foris`)

Foris code contains plugin example. The example isn't documented as well as the development process, but it has a lot of comments in the code. The following text is a result of the demonstration plugin's analysis.

Foris plugins contains following parts:

Python code is responsible for the following:

- Forms definition
- Template selection
- Positioning items in the menu
- Requests processing (AJAX as well)

- Calls to the backend
- Assets import

Templates (Jinja2) are used to generate dynamic HTML and JS.

Static/generated files (SASS,CSS,JS) are mostly styles and third-part libraries.

It's needed to define three Python classes to create a new Foris plugin. These classes must inherit the following classes from the `foris` python module.

ForisPlugin defines and registers a plugin. It defines a plugin name and the paths of static files.

ConfigPageMixin, **SamplePluginConfigHandler** represents a plugin page. Defines templates, slug name and order in menu.

BaseConfigHandler represents a main form handler. It's similar to config handlers described on page 23.

The templates, static and generated files are very similar to described in the analysis of Frontend above on page 24.

2.3.3 Summary

The plugin creation process involves creating two parts of the plugin for backend (`foris-controller`) and frontend (`foris`) separately. The plugins aren't well documented, but they have demos which are self-documented.

The main issue is that third-party developers have to study technologies and API which aren't widespread and standardized (e.g., forms, custom styles, etc.).

2.4 Conclusion

In the course of the analysis, the author became acquainted with the architecture of the application and attempted to identify weaknesses both in the architecture itself and in the way plugins were developed.

The following design is made with an attempt to avoid these problems, and it's described in the design chapter on page 33.

Design

The design part is essential for any project and its goal is to define the structure of the software solution and create bases for the development process. It makes a comprehensive view on the project and allows avoid common architectural, implementation and usability issues.

Moreover, it allows to split and structure development process into particular stages for better control and project management.

This chapter describes the design stage of the new Foris web interface. The chapter is split into a few parts. The first part describes an analysis of functional and non-functional requirements. The design of the plugin system is presented in the second section. The last part contains information about user interface and user experience design.

3.1 Analysis of requirements

In the first part of this chapter the author describes functional and non-functional requirements which were defined after discussed with Turris teams. Analysis of current solution is described in the previous chapter on page 19.

In the second part of this chapter use cases are described which were defined after the analysis of functional requirements.

3.1.1 Functional requirements

Functional requirements define application functionality (it can be said that it defines requirements from the user's point of view). In other words, in this section the author lists functions which the future application should be able to perform.

The following requirements are collected during analysis of the current solutions, backend functionality and from discussions with Turris team members.

The requirements are split into two sections according to network configuration or administration of the router.

3.1.1.1 Network

REQ-1 Wi-Fi Showing and changing the configuration of WiFi modules and networks.

REQ-2 WAN Showing and changing the configuration of the WAN port.

REQ-3 WAN connection test Performing a connection test and presenting results.

REQ-4 LAN Showing and changing configuration of the local network.

REQ-5 DNS Showing and changing configuration of the DNS behavior.

REQ-6 DNS test Performing a DNS connection test and showing results.

REQ-7 Guest network Enabling and configuring of a guest network.

3.1.1.2 Administration

REQ-8 Password Foris and LuCI password changing and setting.

REQ-9 Notifications Router notifications showing and dismissing.

REQ-10 Email notifications The configuring router notification via email.

REQ-11 Backup Creating and restoring a configuration backup.

REQ-12 Reboot Performing a device reboot.

REQ-13 Updates Setting of updates behavior.

REQ-14 Region and time Setting up local region and time.

3.1.1.3 Other use cases

REQ-15 Reference to advanced configuration (LuCI) Possibility of migrating to LuCI configuration interface.

REQ-16 About (information) List of device and operation system version information.

3.1.2 Non-functional requirements

Non-functional requirements are specified by so-called non-behavioral attributes of the application. It mostly specifies how an application should work rather than what functionality it should provide.

Hardware The application should support all developed Turrus routers (Project Turrus (1.0 and 1.1) and Turrus Omnia).

Integration with foris-controller The application should use existed `foris-controller` program as backend.

Notification via WebSockets The application must to communicate with WebSocket server to get and use “real-time” notifications.

Plugins The web interface should be extensible via plugins.

Localization The user interface should be translatable to many languages. All text content has to be extractable in order to be uploaded to a Weblate localization system [36] for subsequent translation.

3.1.3 Use cases

The system doesn't have any deeply nested actions with complex logic. It makes the use cases simple and straightforward, so they just follow the requirements. Because of it, most of the use cases aren't needed to be supplemented with diagrams. But a few of the use cases are accompanied by a diagram to avoid misunderstanding.

The router configuration interface has only one user role which is “user”. So the following use cases are described with only one person. In some of the use cases, the router may be considered as a subject which does action, and it is presented on the diagrams as well.

Every single use case is started with login if the user isn't logged in yet. Thus the login action is skipped in the description of each use case.

UC-1 Change Wi-Fi module configuration

1. User navigates to Wi-Fi module configuration.
2. User enables Wi-Fi module which he wants to configure if it is not enabled yet.
3. User changes Wi-Fi configuration.
4. User is notified whether Wi-Fi configuration changes succeeded or not.

UC-2 Change WAN configuration with connection test

1. User navigates to WAN configuration.
2. User changes WAN configuration.
3. User is notified whether WAN configuration changes succeeded or not.
4. User performs WAN connection test.

UC-3 Change LAN configuration

1. User navigates to LAN configuration.
2. User changes LAN configuration.
3. User is notified whether LAN configuration changes succeeded or not.

UC-4 Change DNS configuration with DNS connection test

1. User navigates to DNS configuration.
2. User changes DNS configuration.
3. User is notified whether DNS configuration changes succeeded or not.
4. User performs DNS connection test.

UC-5 Change guest network settings

1. User navigates to guest network configuration.
2. User enables guest network.
3. User sets guest network.
4. User is notified whether guest network settings changes succeeded or not.

UC-6 Change Foris password and set LuCI password.

1. User navigates to password settings.
2. User changes password.
3. User is notified whether password changes succeeded or not.
4. User sets LuCI password.
5. User is notified whether LuCI password setting succeeded or not.

UC-7 Set sending notifications to email.

1. User navigates to notifications center.
2. User sets his email address in the notifications settings to get notification by email in the future.
3. User is notified whether notification email setting succeeded or not.

UC-8 Backup saving

1. User navigates to backup page.
2. User performs backup collection and downloads backup as file.

UC-9 Backup restoration

1. User navigates to backup page.
2. User uploads backup file to the configuration system.
3. User performs restoration of a backup.
4. User is notified whether backup restoration succeeded or not.

UC-10 Reboot via notification

1. User is notified whether reboot of a device is needed.
2. User performs reboot of the device. User doesn't have to navigate somewhere to perform reboot. He has a possibility to perform reboot using a received notification.
3. User is notified whether reboot was done successfully.

UC-11 Updates setting.

1. The use case begins with automatic updates disabled.
2. User navigates to update settings page.
3. User enables automatic updates.
4. User is notified whether updates changes succeeded or not.
5. User is notified whether some updates are going to be applied.
6. User is notified whether updates were applied successfully.

3. DESIGN

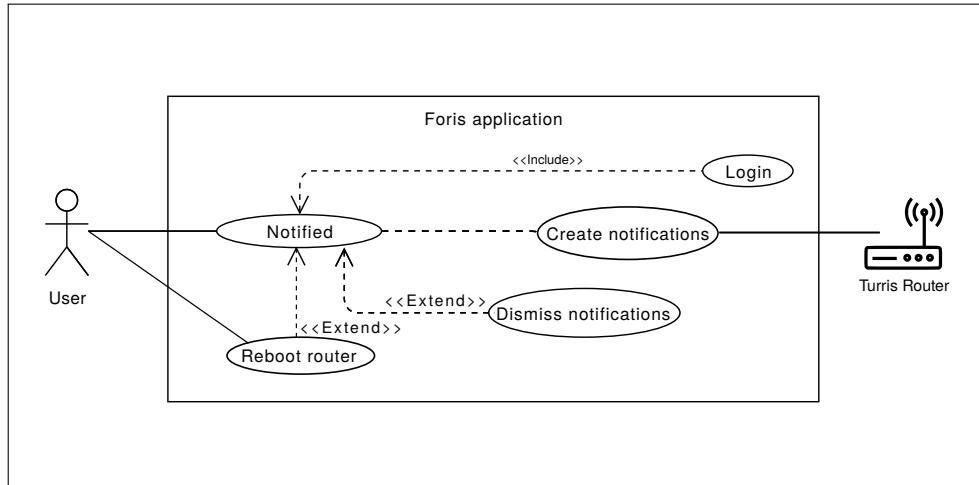


Figure 3.1: Use case diagram of UC-10: Reboot via notification

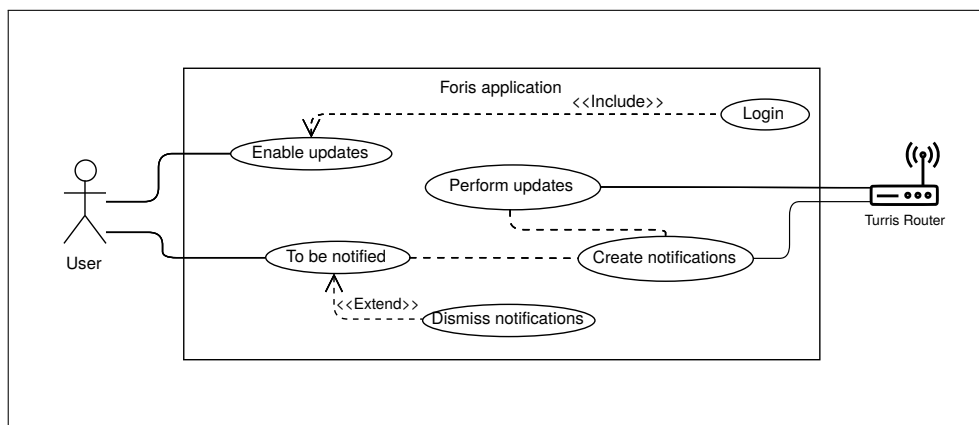


Figure 3.2: Use case diagram of UC-11: Reboot via notification

UC-12 Region date and time.

1. User navigates to administration page.
2. User can see actual date and time settings.
3. User performs region, data and time setting changes.
4. User is notified whether changes succeeded or not.

UC-13 Go to advanced configuration.

1. User goes to LuCI by reference.

UC-14 Get information about device.

1. User goes to About page.
2. User gets required information about device and OS version.

3.1.3.1 Requirements traceability matrix

When all requirements and use cases are defined, we can simply check if all requirements are covered with at least one use case by using the following traceability matrix.

3.2 Technologies

The following stack of technologies was chosen based on the previous analysis.

- Backend
 - Python 3.7
 - Flask 1.0.2
- Frontend
 - JavaScript (ECMAScript 2015+)
 - Bootstrap 4.3
 - Babel 7.2.2
 - React 16.8.6

		Functional requirements															
		REQ-1	REQ-2	REQ-3	REQ-4	REQ-5	REQ-6	REQ-7	REQ-8	REQ-9	REQ-10	REQ-11	REQ-13	REQ-14	REQ-15	REQ-16	
Use cases	UC-1	✓															
	UC-2		✓	✓													
	UC-3				✓												
	UC-4					✓	✓										
	UC-5							✓									
	UC-6								✓								
	UC-7									✓	✓						
	UC-8											✓					
	UC-9											✓					
	UC-10									✓			✓				
	UC-11									✓				✓			
	UC-12														✓		
	UC-13															✓	
	UC-14																

Table 3.1: The traceability matrix of requirements coverage.

3.2.1 Backend

Python Python programming language [21] was chosen just like in the current solution. This is justified by the fact that Python is well known in Turris team which should make maintenance of the Foris simple in the future.

Another reason to prefer Python is the existing set of tools and libraries to facilitate working with `foris-controller` in an abstract way. Those include but are not limited to `foris-client` which is a Python library.

Also, there exists a proven and tested way to deploy Python projects to routers using OpenWRT as OS.

Flask During previous analysis lack of some commonly used features was found in the chosen `bottle` framework ⁶. A lot of open-source web frameworks [37] exist nowadays. After research and discussion with Turris team members it was decided to use Flask [38]. The Flask has several advantages which are going to help with future development of the Foris configuration interface:

- HTTP request handling
- Session management

⁶ See analysis of the current solution on page 24.

- Integrated support for unit tests
- Built-in support for templates (Jinja2)
- Possibility to use extensive set of Flask extensions [39]

Moreover, the Flask framework allows splitting an application into modules (so-called Blueprints [40]), which will be an excellent base to build a pluggable application.

Also, the Turrís team members are familiar with the Flask web framework, which also had an impact on the final choice.

3.2.2 Frontend

React At the beginning of the project redesign it was decided to only use Jinja2 templates. Dynamic interactions and AJAX calls were planned to be made using Vanilla JS [41], which stands for native JavaScript relying only on standard libraries. Although, after the initial analysis it was found that using some of the existing tools can improve code structure and readability.

The Turrís team members don't have experience using any JavaScript library or framework, so choosing the proper technology was completely up to the author.

From a large array of currently existing frameworks and JavaScript libraries React [42] was chosen. There are several reasons for this choice that are described below.

The React is not a framework, it is a library that allows you to use only the necessary parts and to avoid over-engineering. This approach is different from other frameworks that are made with an MVC pattern and require developers' effort to keep specific code structure. It's a proper way to build large web application interfaces with complicated logic which Foris isn't.

The pros of React is that it uses virtual DOM. It allows to split (isolate) code into reusable components (or libraries) and keep logic separately (e.g., validation, AJAX API calls, communication with WebSockets, etc.). It enables to make better code structure by composing components, unlike solutions which work with real DOM and don't have such a feature.

Also, React is well-documented and currently it's one of the most used solutions having a vast and rapidly growing community [43]. Moreover, it's supported by Facebook, Inc. All above ensures future support and stability.

Babel The toolchain that is mainly used to convert ECMAScript 2015+ code into a backward compatible version of JavaScript in current and older browsers or environment [44]. There are a few reasons to use the Babel in our future web application described below.

As was mentioned, Babel allows using features from newer versions of ECMAScript without carrying for deprecated browsers. It makes the development

process much more straightforward. So a developer can use any construction and syntax sugar from modern ECMAScript standards without headache.

Since the React library uses JSX [45] which isn't a part of the JavaScript language it should be compiled to compatible JavaScript code. Babel can also do it using special presets.

Moreover, Babel provides code optimization and minification during server-side compilation. It can optimize client-side code and make it more efficient. Minification allows to decrease the amount of code sent to the client which positively affects latency.

Babel will serve to build React application and minification. Building and minification of the JS client code will be done in the Turris build system on the deploy stage. So the router will get the prepared client-side code without sources. This approach decreases router CPU usage and communication load.

Bootstrap The current solution of the Foris web interface uses custom written styles⁷. If we take into account insufficient documentation, support and implementation of new functionality becomes a burden from a developer's standpoint.

One of the main objectives of this work is to encourage third-party developers to create modules (plugins) for the Foris interface. In light of the above, some standardized techniques and tools should be used.

Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. It's the world's most popular framework for building responsive, mobile-first sites [46].

The most frequently used elements in every router configuration interface are the forms. Foris configuration interface almost entirely consist of forms. Bootstrap allows for creation of nicely styled HTML forms without involving a developer to make layouts from scratch. It has a lot of premade style components such as buttons, form fields, popups, alerts and navigation elements.

Bootstrap has a large community and thanks to it a lot of styling themes were developed. It's easy to switch from one theme to another just by replacing Bootstrap source code.

3.3 Architecure

The architecture of the future application is going to be similar to the current Foris application. It's due to the use of buses and services which are described in the non-functional requirements of the application which are defined on page 35.

The design of the architecture is visualized on the following architecture design diagram on the Figure 3.3.

⁷See an analysis of the client-side code of the current solution on page 24.

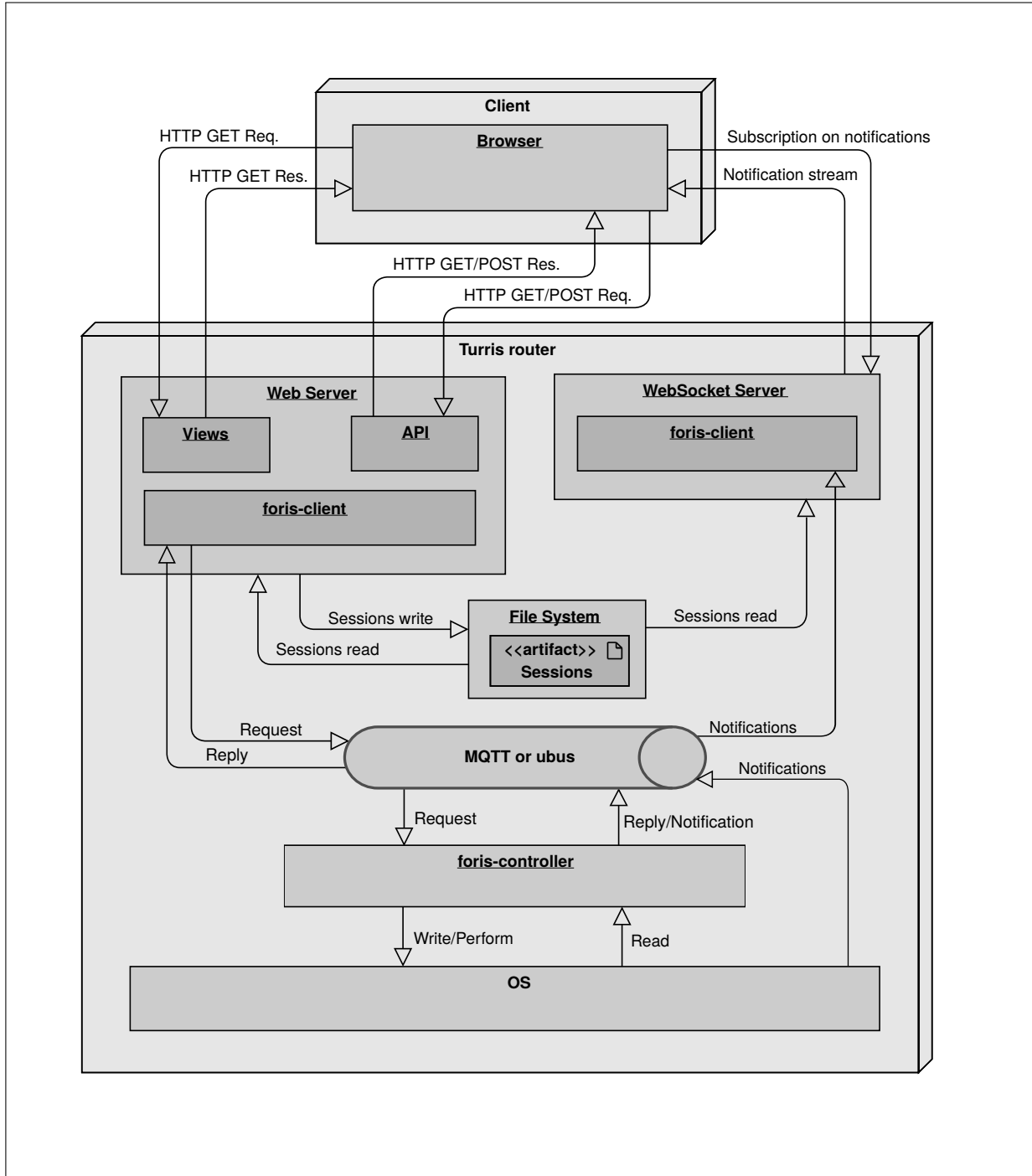


Figure 3.3: Architecture design of the future Foris application.

The detailed description of the architecture nodes and instances is described below in the following sections.

3.3.1 Web server

3.3.1.1 Flask application

The main part of the application is going to be a Python application created via Flask [38] framework.

The main task of the Flask application is to serve HTTP requests and response. The Flask application will have two main components (in Flask terminology they are called blueprints [40]).

The first blueprint is the main application. It defines the main views. These views are single pages of Foris application, or in other words, they are individual tabs. The list of the views is shown in the “Views” section below.

The second subpart of the Flask application is API. It’s going to be used for serving AJAX calls which will be made by React application. The list of the API endpoints is also described below in the “API” section.

3.3.1.2 `foris-client`

Due to non-functional requirements listed on page 35 the new Foris application will use the same Python library called `foris` [32] to perform backend actions. The `foris-controller` is described on page 27 in the “Analysis of the current solution” chapter.

3.3.1.3 `lighttpd`

Flask has its own built-in web server, although this server is intended only for development purposes. The built-in Flask web server is pretty slow, and it’s not production-ready. Because of it, it was decided to use `lighttpd` [23] server to serve Foris web application.

The `lighttpd` web server was chosen due to its efficiency in serving HTTP requests using only a small memory footprint. This is extremely relevant to the router’s domain.

3.3.2 WebSockets server

The same WebSockets server will be used in the production release. Although, there are changes planned in the session management that should be implemented in the actual `foris-ws` WebSockets server. These changes are described below.

3.3.3 Authentication and sessions management

3.3.3.1 Authentication

The authentication is done in the same way as it is in the current solution and is described in the the chapter 2 chapter on page 28.

3.3.3.2 Sessions management

The session management will have changes compared to the current Foris application. In the new application the session management will be moved to Flask application (versus using `ubus` session management).

It will be done using `Flask-Session` [47] extension. The `Flask-Session` allows using server-side sessions and storing session in many ways, e.g., using a database, Memcached or filesystem. The sessions should be shared between Flask application and the WebSockets server. Due to this it was decided to store the session on the filesystem and allow the WebSockets server to read sessions. This approach will cause changes in the `foris-ws` WebSockets server which will be described in the “Implementation” chapter.

Both authentication and session management processes are illustrated via sequence diagram on the Figure 3.4. A curious reader can compare this diagram with Figure 2.2 on page 29 to observe the changes.

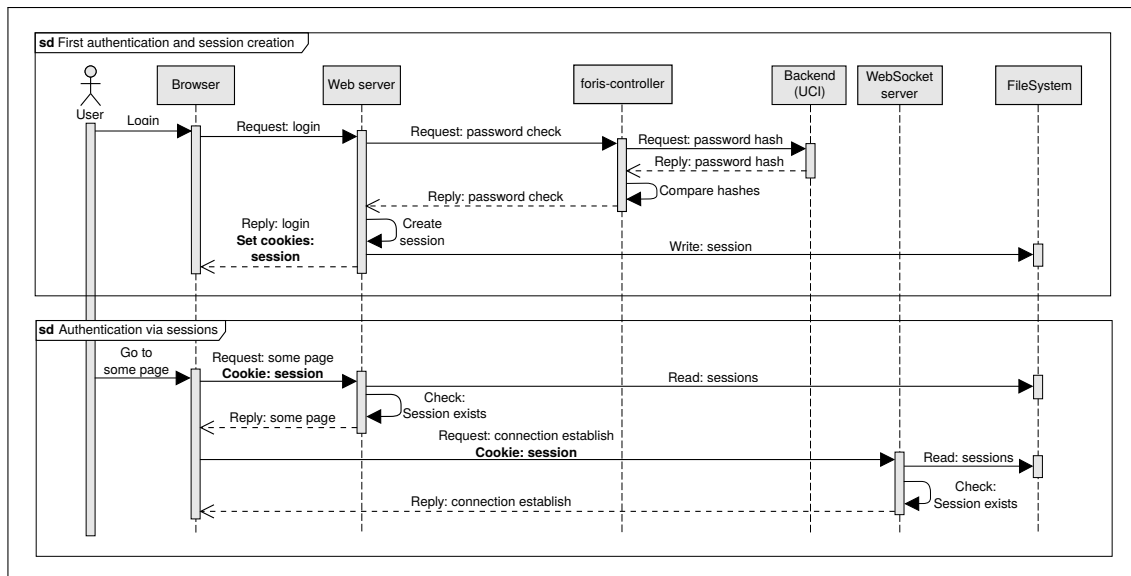


Figure 3.4: *Authentication and session management sequence diagram.*

3.3.4 React application

Since one of the main task for the reimplementaion of the Foris was to make a better plug-ins system, it was decided not to use a single React application. Otherwise, it could complicate the creation of plug-ins for developers unfamiliar with the React library.

So the React client-side code will be split into many applications. Each application will represent a single tab in the interface (e.g., notifications, Wi-Fi, WAN, etc.).

This approach allows avoiding using React library for following plugins implementation. Moreover, this approach shift routing from client-side code to the Flask application.

3.4 Plugins system

3.4.1 Flask Blueprints

The plugins system will be implemented with Flask Blueprints [40]. Flask blueprints system gives third-side developers full freedom in making plugins for our Foris interface. A developer can define URLs, templates and all the logic as he wishes without any limitations. Moreover, the developer is not restricted in JS libraries or frameworks to develop plugins.

3.4.2 Python module

A resulting plugin is encapsulated with all dependencies and can be provided as a Python module with installation via `setup.py` script. This approach allows to define modules which can be automatically registered and integrated into Foris application just by using a particular name.

3.4.3 Registration

The author suspects that Flask Blueprints system may not be enough, because plugins should specify the name of navigation item and position in the navigation list. But this falls under implementation details, and it will be investigated and described in the next chapter.

3.4.4 `foris-controller`

The integration plugin for the `foris-controller` stays the same because non-functional requirements force author to use the existing `foris-controller` program. Creation plugin for this part of the Foris application is described in the “Analysis of the current solution chapter” on page 29.

3.5 UI/UX Design

The large part of the UI and UX design was developed under the MI-NUR course under the direction of the supervisor of this paper Ing. Jiří Hunka. The analysis, wireframes creation and development of the non-functional prototype was made by members of the following team: Bodnar, B., Karola, A., Kryvosheienko, M., Laskov, B. and Samigullina, G. The following section of this chapter is a restructured and rewritten result of that work.

3.5.1 Plan

The plan of the following user interface design is based on the MI-NUR course lectures.

- Product statement
- Use cases
- Task list definition and analysis
- Prototyping
- Heuristic evaluation
- Redesign

3.5.2 Product statement

The Foris web application is a configuration interface for Turris routers which provides access to network settings and device maintenance. The Turris routers are more powerful and advanced than a regular home router. In other words, it is intended for more experienced users (“geeks”). So the web interface should provide access to everything needed by beginners and advanced users router settings and maintenance tasks.

3.5.3 Use cases

The first part of the user interface design of the future Foris application is use case definition and analysis was made in the “Analysis of requirements” chapter and described on page 35.

3.5.4 Task list

A task list helps with the initial design and describes the application from a user’s point of view. The task list is based on the use cases.

The list of task is split into groups by obvious affiliation to a specific category. It’s made to improve the readability of the long task list.

- Notifications
 - Show notifications.
 - Dismiss notification.
 - Dismiss all notifications.
 - Set email notifications setting.
- Administration
 - Change Foris password.
 - Change root password.
 - Change time settings.
 - Change region (timezone) settings.
 - Change interface language.
 - Reboot device.
- Network configuration
 - WAN
 - * Set IPv4 configuration type (DHCP, Static, PPPoE).
 - * Set IPv6 configuration type (DHCPv6, Static, 6to4, 6in4).
 - * Set custom MAC address.
 - * Perform WAN connection test.
 - LAN
 - * Set and configure Router LAN mode.
 - * Set and configure Computer LAN mode.
 - Wi-Fi (for both modules)
 - * Enable module.
 - * Configure Wi-Fi module settings.
 - * Configure Wi-Fi network settings.
 - * Configure Wi-Fi Guest network.
 - DNS
 - * Set DNS forwarding.
 - * Enable/Disable DNSSEC [48].
 - * Perform DNS connection test.
 - Interfaces
 - * Assign an interface to a certain network.
 - * Disable an interface.

- Guest network
 - * Enable/Disable guest network.
 - * Set guest network settings.
- Updates
 - Enable/disable automatic updates.
 - Set update approvals.
 - Select packages to install and update.
 - Set restart after update.
- Misc
 - Log in.
 - Log out.
 - Go to LuCI (Advanced administration).
 - Display information about device and operating system (such as serial number and OS version).
 - Start the first configuration guide.

3.5.5 Activity diagrams

Activity diagrams help to better understand user interface behavior.

Most of tasks are straightforward and don't need to be illustrated with activity diagrams. In other cases, tasks with more complex logic are accompanied by activity diagrams.

3.5.5.1 Notifications

Notifications system allows the user to get real-time notifications about router activity. Moreover, if the router reboot is needed, the user gets notification about it and can perform device reboot in the same place.

This process is illustrated via the activity diagram on the Figure 3.5.

3.5.5.2 WAN settings

All network configuration changing flows are very similar. Thus the description of WAN settings flow covers them.

DNS and WAN network configurations flows are extended via configuration tests. It's illustrated on the following activity diagram on the Figure 3.6.

The rest of the network configurations are the same except for the missing connection test.

3. DESIGN

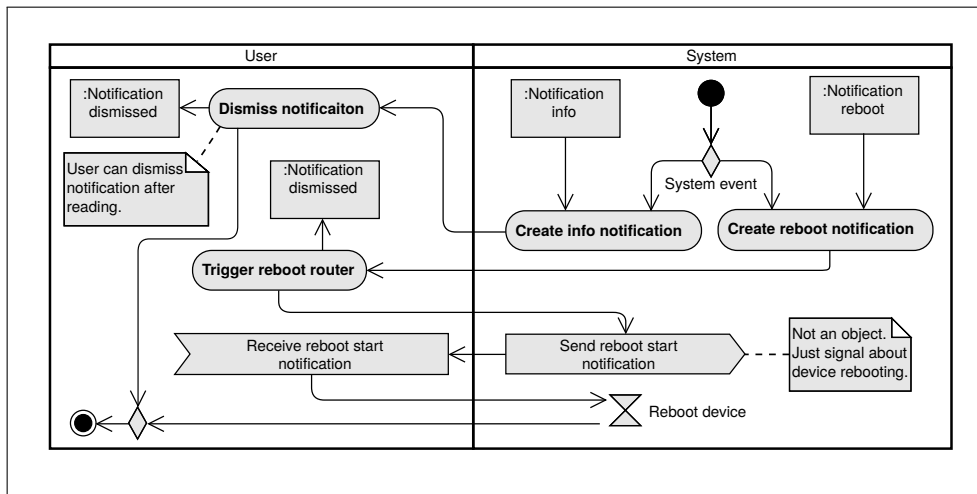


Figure 3.5: Notifications activity diagram.

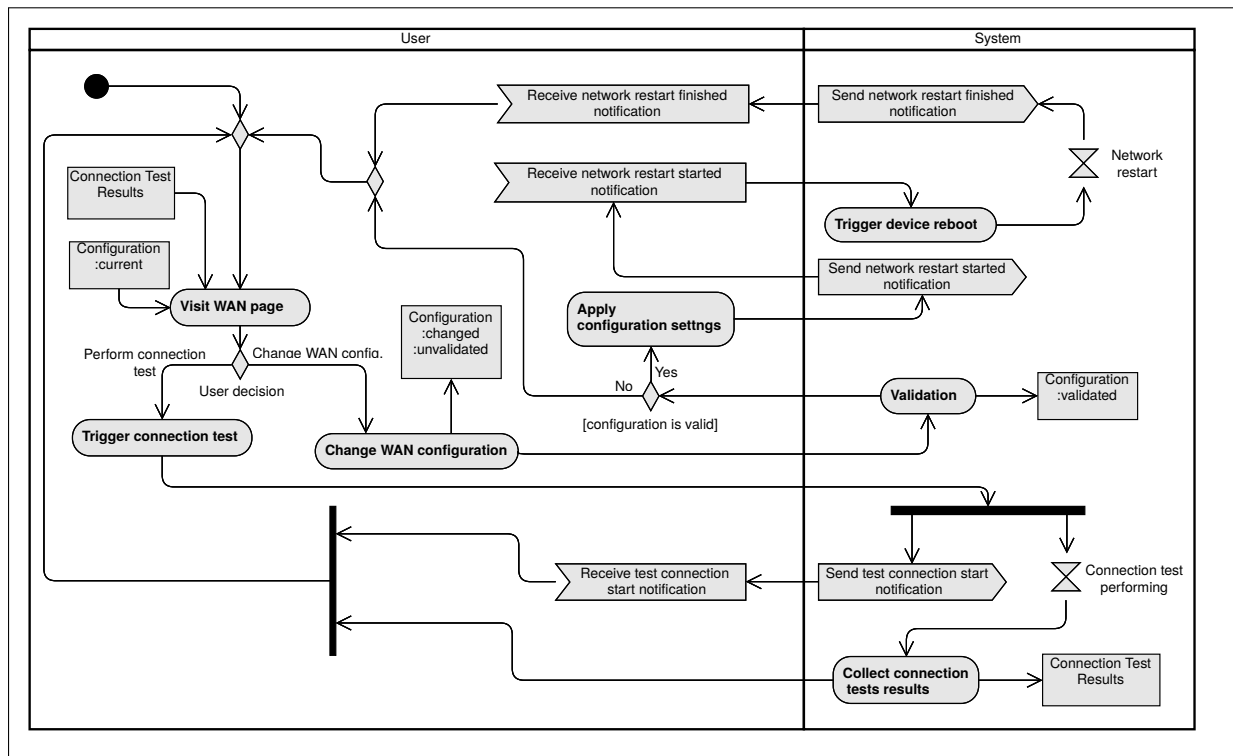


Figure 3.6: WAN settings activity diagram.

3.5.6 Prototyping

The prototyping helps to structure the user interface design process. The prototyping is split into two parts. The first part is wireframes (lo-fi prototype) creation and the second part is interactive (hi-fi prototype) prototype development. Both parts are described in the following sections.

3.5.6.1 Wireframes

Wireframes are a fast solution to design the structure of the future user interface. Moreover, wireframes are material for future hi-fi prototype.

Wireframes are made with Axure prototyping application [49].

All wireframes are provided in the appendixes part of this paper on page 117.

3.5.6.2 Hi-fi prototype

Hi-fi prototype, the web application for the target platform that originates from the lo-fi prototypes (wireframes) and research notes. The prototype is implemented in the target platform, so it's web application developed in Python programming language using a Flask framework. The frontend part of the prototype is made via Bootstrap 4 framework using Flexbox Layout.

All screenshots of the hi-fi prototype are listed in the appendixes part of this paper on page 129.

The first iteration of usability testing is done using this hi-fi prototype. The result of this usability test is described on page 83.

3.6 Conclusion

All of the materials developed in the design stage of this work and described in this chapter serve as a base for the following implementation part.

Implementation

The emphasis of this chapter is put on the implementation stage details. The implementation is based on the design stage and it leverages results of the previous analysis in an effort to develop a better web interface for a router's configuration.

The project got a working title reForis, which stands for redesigned Foris. It was given a new name to avoid confusion in the names of the projects because the current Foris web interface is still supported. The project's name may change to Foris v2.0 after it passes the testing stage.

All following described source code is available on the CZ.NIC labs Git-Lab [50] under GNU General Public License v3.0 license.

The screenshots of the resulting reForis web interface are available in the appendix section of this paper on 147 page.

4.1 Development environment

4.1.1 Hardware

Hardware required for implementation in the form of Turris Omnia router has been kindly provided to the author by the members of the Turris team.

4.1.2 Development build deployment

The software development of the router has a specific aspect. The source code of the Python application should be transferred to the router for testing purposes. It's better if the code changes are automatically synchronized and a developer doesn't spend time transferring code manually.

There were many ways to approach code deployment e.g., mounting a disk via `sshfs` [51]. Author has tried this particular method at the start of the project but after some time he has discovered a few issues with this solution.

4. IMPLEMENTATION

Firstly, the author didn't want to open ports used by ssh on the development computer for security and paranoia reasons. Eventually it led the author to open port 22 on the router. A directory on the router was mounted to development machine.

There is yet another solution to this problem. For example, one could store all the code directly on the router. The issue with this approach is that if a developer forgets to synchronize code between his machine and a router, access to the router will be lost as soon as one of the devices leaves network. The solution is to use another synchronization program such as `rsync` [52].

However, the author has discovered a way to improve upon the last solution. It is built-in in the PyCharm IDE [53]. PyCharm allows a developer to set the remote machine for code deployment. Moreover, it monitors code changes and only deploys recently modified files. It also comes with an option to push selected source code files using the IDE interface.

The screenshot of the deployment settings is shown on the Figure 4.1.

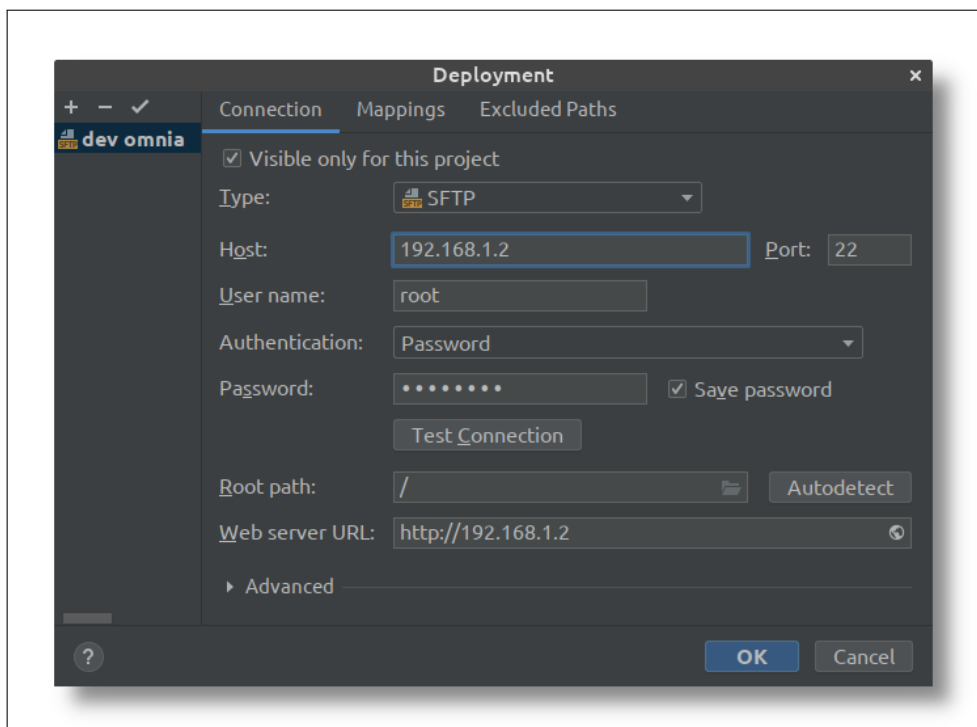


Figure 4.1: Screenshot of the PyCharm deployment settings.

Another issue lies within eMMC [54]. It can only sustain 3–10K rewrite cycles before it starts to cause bit errors [55]. In this regard, it's better to send the code to the RAM. This means that a developer needs to set source code mapping to `/tmp` which is then mapped to the RAM instead of eMMC flash disk. This can be achieved with `tmpsfs` [56].

4.1.3 Version control system

Well-known Git [57] is used as a version control system for this project. Another service that was used in the project is the CZ.NIC GitLab [58] company server. All project source code is open-sourced and can be found in the Turris project in the reForis repository [50].

GitLab provides many tools [59] as GitLab CI/CD [60] which is also used in this project and described below.

4.1.4 Continuous integration

CI is the set of approaches and practices with the main goal of frequent source code testing to prevent integration issues [61].

GitLab comes with its own CI and CD pipeline tooling [60]. This tool was used in the project to perform regular tests and lint checking.

GitLab CI/CD allows performing tests in the docker container [62] with a very simple configuration.

```
image: debian:9

stages:
  - test

before_script:
  - apt-get update && apt-get -y install sudo make

test:
  stage: test
  script:
    - make prepare-dev
    - make test

lint:
  stage: test
  script:
    - make prepare-dev
    - make lint
```

Listing 4.5: Example of .gitlab-ci.yml configuration file.

It allows to run tests upon each code push. In case tests aren't passing the development branch can't be merged into the master branch.

4.1.4.1 Tests

The Python application as well as React application is tested via CI tools. Code testing tools and approaches are described in detail in the testing chapter of this paper on page 72.

4.1.4.2 Linters

Linters are tools which allow to perform generally static source code analysis to prevent bugs and stick to a certain code style. It's especially useful when the project is developed by a team of many developers.

The Python code is analyzed by two linters `pylint` [63] and `pycodestyle` [64]. `Pylint` is a customizable linter allowing to check type errors, unused imports and much more. `Pycodestyle` checks if code corresponds to PEP8 [65] style guide.

The JS application code is checked by `ESLint` [66]. It's used to find problematic patterns in JavaScript code base. `ESLint` is extensible which allows using specific plugins for frameworks and libraries such as React. These plugins are extremely helpful in avoiding common bugs when using the framework. In the project author used the following list of lint plugins for React library:

- `eslint-config-react-app`
- `eslint-plugin-jsx-a11y`
- `eslint-plugin-react`
- `eslint-plugin-react-hooks`

4.2 reForis project

The new Foris application project got a development name of reForis (which means a redesign of Foris).

ReForis project has the following directory structure illustrated on the Figure 4.2.

4.3 Backend

4.3.1 Flask application

As was described in the "Design" chapter on page 44 the backend part of the reForis is Flask application. The application directory structure is listed on the Figure 4.3.

```

reforis
├── reforis ..... the Flask application directory
├── reforis_static.....the Python module with static files
├── tests..... the tests set for Flask application
├── js..... the React application directory
├── setup.py.....the definition of the python Package installation
├── MANIFEST.in..... defines non Python sources
├── babel.cfg..... the Babel configuration
├── Makefile..... the utility contains set of the development helpers
├── scripts..... the directory contains script for timezone generation
├── pycodestyle..... the definition of the Python code style
├── pylintrc..... the pylint configuration file
├── README.md
└── LICENSE

```

Figure 4.2: *reForis* project directory structure.

```

reforis
├── __init__.py....the python module and application factory definition
├── config.....the Flask application configurations directory
├── api.py.....the API Blueprint with endpoints implementation
├── views.py..... the Views Blueprint with views implementation
├── plugins.py.....the helper to make plugins registration
├── locale.py.....the translations helper to generate JS translations
├── auth.py..... the login and sessions implementation
├── backend.py.....the foris-controller communication helper
├── static.....the static dir, contains images, CSS and JS files
├── templates.....the templates directory
└── translations.....the locale directory, contains .po and .mo files

```

Figure 4.3: *reForis Flask application directory structure*.

4.3.2 Application factory

During implementation it was decided to use Flask application factory function instead of application object definition. This approach allows to create more Flask applications with a different configuration for more thorough testing. It also encourages to create a better application structure by defining all of the application dependencies in one place [67].

The application factory function code is shown on the Listing 4.6.

```
def create_app(config):
    from flask import Flask
    app = Flask(__name__)
    # Config file load.
    app.config.from_pyfile(f'config/{config}.py')

    # Session management registration using Flask-Sessions.
    from flask_session import Session
    Session(app)

    # Protected/unprotected views definition.
    from reforis.auth import register_login_required
    register_login_required(app)

    # foris-client helper registration.
    set_backend(app)

    # locale registration (using Flask-Babel extension).
    set_locale(app)

    # Blueprints registration.
    from .views import base
    from .api import api
    app.register_blueprint(base)
    app.register_blueprint(api)

    # Plugins load and~registration.
    load_plugins(app)

    return app
```

Listing 4.6: Application factory code example.

The application is split into two main subparts called Blueprints in the Flask terminology. These parts are described below.

4.3.3 Views

The “Views” blueprint defines very simple views which render single pages with a template. These views don’t have any logic in them because all further

communication between client and server goes via AJAX calls using reForis API which is described in the next section.

An example of such view is shown in Listing 4.7 code sample.

```
from flask import render_template

@base.route('/wifi')
def wifi():
    return render_template('wifi.html')
```

Listing 4.7: Wi-Fi view definition.

But the simplicity of views has a few exceptions which are described in the following sections.

4.3.3.1 Auth views

The login and logout logic isn't moved to API because of the simplicity of using pure HTML form without AJAX calls and built-in in Flask redirect functions.

The login view is illustrated in the following code examples.

```
from flask import redirect, render_template, request,\
session, url_for
from reforis.auth import login

@base.route('/login', methods=['GET', 'POST'])
def login():
    error_message = None
    if session.get('logged', False):
        return redirect(url_for('Foris.index'))

    if request.method == 'POST':
        password = request.form['password']
        if login(password):
            return redirect(url_for('Foris.index'))
        error_message = _('Wrong password.')
    return render_template(
        'login.html',
        error_message=error_message
    )
```

Listing 4.8: Login view code example.

4.3.3.2 Administration view

Administration view has region and time settings which need additional region, country and city translations. So in this way, we need to pass additional timezone translations `tzinfo` into the template and load it into Babel JS library.

This process is illustrated by code examples in the Listing 4.9 and Listing 4.10. The translations technique is described in detail in the “Localization” section of this chapter on page 72.


```

from flask import current_app, render_template
from reforis import TranslationsHelper

@base.route('/administration')
def administration():
    babel = current_app.extensions['babel']
    translations = TranslationsHelper.load(
        babel.translation_directories,
        [get_locale()],
        'tzipno'
    )
    return render_template(
        'administration.html',
        babel_tzipno_catalog=translations.json_catalog
    )

```

Listing 4.9: Administration view code.

```

...
<script type="text/javascript">
    ForisTranslations.load({{ babel_tzipno_catalog | safe }});
</script>
...

```

Listing 4.10: Loading additional tzipno translations in a template.

4.3.4 API

The React application communicates with the Flask server via AJAX calls. The design of the API is imposed by `foris-controller`, and communication tasks between client-side application and backend.

All API endpoints are implemented as views and belong to a Blueprint having the same name – API.

In this case, the API Blueprint serves as a layer between the `foris-controller` and a client application. It passes required endpoints with some adjustments from Foris controller to the client via HTTP.

Each of the `foris-controller` endpoints which represent network, router system operations and Foris application configurations has two actions: `get_settings` and `set_settings`. These endpoints are “translated” to par-

ticular HTTP endpoint with GET and POST methods by appropriate actions. This aspect is shown on the Listing 4.11.

```
from flask import current_app, request, jsonify

@api.route('/wifi', methods=['GET', 'POST'])
def wifi():
    return _foris_controller_settings_call('wifi')

def _foris_controller_settings_call(module):
    try:
        res = ''
        if request.method == 'GET':
            res = current_app.backend.perform(
                module, 'get_settings'
            )
        elif request.method == 'POST':
            data = request.json
            res = current_app.backend.perform(
                module, 'update_settings', data
            )
        return jsonify(res)
    except ExceptionInBackend as e:
        _process_backend_error(e)
```

Listing 4.11: Translation of the typical foris-controller setting module to the API endpoint.

The full list of implemented API endpoints is available in the appendix part of this paper on page 141.

4.3.5 Static files

One specific aspect of the router configuration interfaces is that while setting up a router a user may not be connected to the Internet. This leads to the fact that CDNs servers can't be used to deliver static JS and CSS libraries. In this way, all static files should be stored on the router and be directly delivered to the user's browser.

4.3.6 Authentication and sessions management

The authentication process and session management are described and illustrated via sequence diagram in the "Design" chapter on page 28.

4.3.6.1 WebSockets server

As was described in the previous chapter, moving sessions storage to the filesystem triggers changes in the WebSockets server. This changes affect the authentication process.

The `FileSystemCache` object from `werkzeug` library [68] is used for getting sessions stored in the file system, just in the same way as in `Flask-Session` library. This process is shown on Listing 4.12.

```
# ... get session_id from cookies

from werkzeug.contrib.cache import FileSystemCache

fs_cache = FileSystemCache(SESSIONS_DIR)
data = fs_cache.get('session:' + session_id)

if data is None:
    # Process "Session not found" error

if not data.get('logged', None):
    # Process "Session is found but not logged"

#Process successful authentication
```

Listing 4.12: The code example of getting and checking session stored in a filesystem.

For more information about authentication via WebSockets, please check `foris-ws` repository [69].

4.4 Frontend

As was mentioned in the “Design” chapter on page 46, the frontend part of the `reForis` project is developed with `React` library.

4.5 JavaScript bundler

As was described in the design chapter on page 41: `Babel` is used for `ECMAScript 2015+` and `JSX` code compiling. But this solution needs another tool for `JavaScript` modules bundling. `Webpack` [70] is used in this project as a module bundler. Moreover, it has a built-in minification module which allows to reduce the size of resulting `JS` code.

4.5.1 React application file structure

React doesn't enforce any specific file structure. Based on the React file structure tutorial [71] the author has chosen the "Grouping by features" approach to structure React components.

The React application structure is shown on the Figure 4.4 ⁸.

4.5.2 Hooks

Hooks is a new React approach for creating components [72] which is widely used in this project. React Hooks came with React 16.8 and brought a noticeable change in the component development.

There was an issue with sharing reusable state logic between components before hooks had been released. One of the approaches was to create so-called HOCs (Higher-Order Components). It's a wrapper component which contains state logic and passes it to children components. The HOCs are hard to maintain and debug because if you have a lot of HOCs with different state logic and build many layers from it, then you get into so-called wrapper hell. Wrapper-hell is when state's data passes between many HOCs layers, and you aren't able to debug and maintain it.

The author faced this issue at the beginning of this project. As a result, all components were rewritten using the hooks technique.

Hooks allow you to split state logic into small parts and attach them independently where it's needed. Moreover, you can compose the state logic of many hooks into one and attach it to a particular component.

There is a convention that all hook names should start with `use` word.

In the next sections the most important and interesting hooks of the reForis application are described.

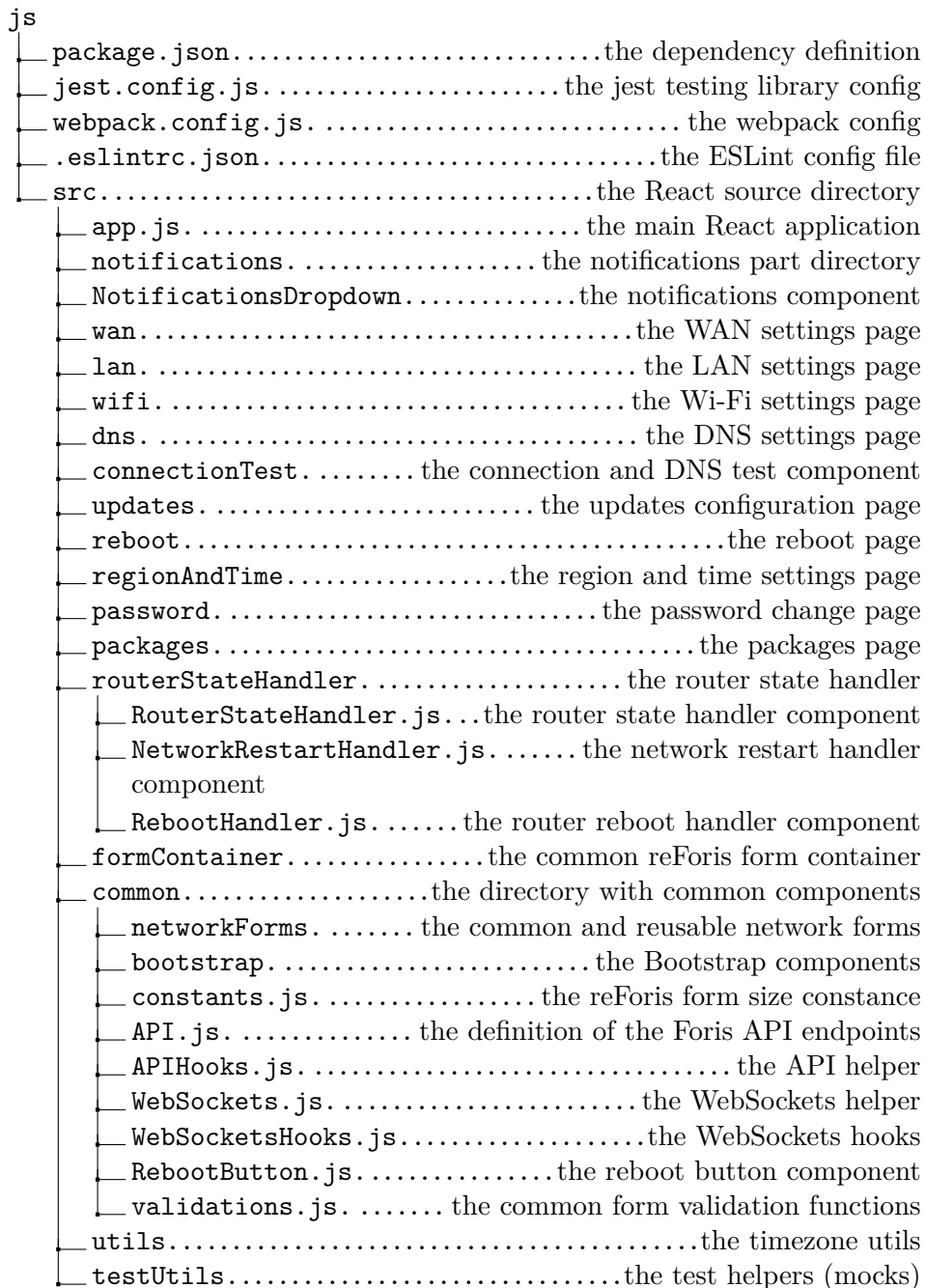
4.5.2.1 useForm

The `useForm` hook encapsulates all form logic: form fields data, error validation and form state. This hook provides a few functions to change hooks' states.

The hook gets a validator as an additional function parameter. `useForm` validates form data when it changes and outputs errors in case of invalid input values.

The main purpose of this hook is to split the specific Foris form logic and common form processing logic. This hook is used in the `ForisForm` component.

⁸ The React file structure is not full, the full structure is available in the project repository.

Figure 4.4: *reForis React application directory structure.*

4.5.2.2 API hooks

The application has two API helper hooks: `useAPIGet` and `useAPIPost`. These hooks encapsulate API calls logic states of response and request. `useAPIGet` hook is shown on the following Listing 4.13.

```
import {useCallback, useReducer} from 'react';

export function useAPIGet(url) {
  const [state, dispatch] = useReducer(
    APIGetReducer, {
      isLoading: false,
      isError: false,
      data: null,
    }
  );

  const get = useCallback(async () => {
    dispatch({type: API_ACTIONS.INIT});
    try {
      const result = await axios.get(
        url, {timeout: TIMEOUT}
      );
      dispatch({
        type: API_ACTIONS.SUCCESS,
        payload: result.data
      });
    } catch (error) {
      dispatch({
        type: API_ACTIONS.FAILURE,
        payload: error.response.data
      });
    }
  }, [url]);

  return [state, get];
}
```

Listing 4.13: `useAPIGet` hook code example.

4.5.2.3 Router state handler hooks

The hooks which handle device reboot and network restart. They have very similar logic and implementation. The difference is only in subscribing on the different `foris-controller` modules for WebSockets notifications and redirecting after these processes are done.

These hooks provide the following logic: When reboot or restart notification is received from the WebSockets server (with a list of the possible future IP addresses of the router) then WS connection is closed and the user is notified that reboot or network restart has been triggered. Then it waits until the server is down via health check pooling and notifies a user with a reconnecting message. Then it performs health check pooling to each IP address from the list received from WebSockets notification. When one of the servers respond then redirect a user's browser to the login page (in case of a reboot) or same page (in case of network restart).

The user should log in again after reboot because sessions are stored in the RAM and are erased after device reboot.

This process is illustrated in the reboot handling sequence diagram on the Figure 4.5.

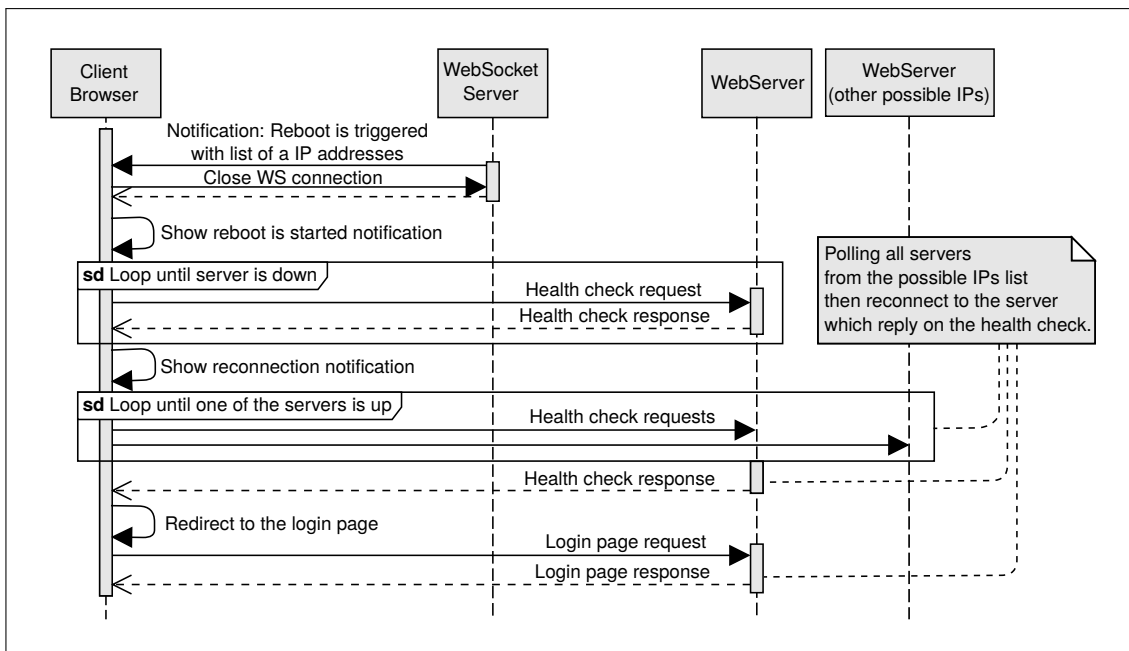


Figure 4.5: *Reboot handling sequence diagram.*

4.5.3 JSX

React library comes with so-called JSX language extension (dialect) of the JavaScript. It combines JS and HTML syntax and makes it possible to compose HTML elements and React components directly inside JS code [45]. The reader may see the JSX usage in the code examples in the next section.

4.5.4 React components

React components are the building blocks of the application. Most React applications have main components (called containers) which include other components.

This section describes the main and most interesting components and containers of the reForis application. But before we start let's take a look at a few React helper libraries used in this project.

4.5.4.1 PropTypes

JavaScript is a dynamically typed language. Using wrong types may cause bugs in the code. The `prop-types` library was made to prevent this annoyance at least in the React components [73]. This library allows to specify components of property types and to check them during runtime. In case a wrongly typed property gets passed to a component, an exception is raised.

This approach makes debugging and error prevention much easier and it's used for all components in this project.

4.5.4.2 ForisForm

The main Foris form container. This container is used for each settings form in the application.

It uses `useForm` hook and provides all needed functionality to the children form components. Using this container allows keeping form components implementation simple. The children form components specifies the fields and form data changes and doesn't have a responsibility of the API calling and WebSockets communication.

`ForisForm` provides `formData` and `formErrors` objects which contain corresponding data. These objects mostly have other nested objects inside of them because of the nested forms usage.

Also, `ForisForm` provides `setFormValue` function which gets updated rule and returns `onChange` event handler. It allows to specify which particular form data should be changed when some change has happened in the form.

Example of using a Foris form is shown in Listing 4.14.


```

import React from 'react'; // Is needed to use JSX
import ForisForm from '../formContainer/ForisForm';
import LANForm from './LANForm';
import API_URLs from '../common/API';

export default function LAN({ws}) {
return <ForisForm
  ws={ws}
  // Definition of the API endpoint and WS module
  forisConfig={{
    endpoint: API_URLs.lan,
    wsModule: 'lan',
  }}
  prepDataToSubmit={prepDataToSubmit}
  validator={validator}
  >
  <LANForm/>
  <LAN_DHCP_ClientsList/>
  </ForisForm>
}
// prepDataToSubmit and validator functions definitions...

```

Listing 4.14: Using ForisForm container with LAN form code example.

4.5.5 Form example

Let's consider using ForisForm container with the LAN component. The LAN component has two nested forms (subforms) which are switched when a user chooses a proper LAN mode from two options (**managed** and **unmanaged**).

You can see the usage of this object in the following LANForm code example on Listing 4.15. In this example the `formData.mode` string is passed as a value to the `Select` component. And `formData.mode_managed` object is passed to the `LANManagedForm` and process with the same approach as in the `LANForm` component.

This approach allows making arbitrarily nested forms which are using similar logic. It improves readability and maintenance of the forms implementation.

```
import React from 'react';
import Select from '../common/bootstrap/Select';
import LANManagedForm from './LANManagedForm';
import LANUnmanagedForm from './LANUnmanagedForm';

export default function LANForm({
  formData,
  formErrors,
  setFormValue,
  ...props
}) {
  const lanMode = formData.mode;
  return <>
    <h3>{_('LAN Settings')}</h3>
    <Select
      label={_('LAN mode')}
      value={formData.mode}
      choices={LAN_MOD_CHOICES}
      onChange={
        setFormValue(value => ({mode: {$set: value}}))
      }
      {...props}
    />
    {lanMode === LAN_MODES.managed ?
      <LANManagedForm
        formData={formData.mode_managed}
        formErrors={formErrors.mode_managed}
        setFormValue={setFormValue}
        {...props}
      />
      : lanMode === LAN_MODES.unmanaged ?
        <LANUnmanagedForm
          // ...
        />
        : null}
    </>
  }
}
```

Listing 4.15: LANForm code example.

4.6 Plugins system and demo plugin

Implementation of the plugin system including demonstration plugin is described in this section. This implementation is based on the plugin system design mentioned on page 46.

Diagnostics plugin was chosen as a demo plugin due to its simplicity and visibility.

JavaScript code is not used in this plugin. It is done in order to demonstrate third-party developers that it is not mandatory to use any JS framework and that they have freedom in choosing proper tools.

The source code of `reforis-diagnostics` plugin is available on the CZ.NIC GitLab [74].

4.6.1 Registration

As was mentioned in the design section, the plugin registration is done using `setup.py` setup script [75]. Each reForis plugin must to define so-called `entry_points` [76] section in their `setup.py` file. The `entry_points` section should contain an entry point with a name defined in `foris.plugins` section and a path to a Blueprint definition. This is illustrated in the subsequent Listing 4.16.

```
setuptools.setup(  
    name='reforis_diagnostics',  
    # ...  
    entry_points={  
        'foris.plugins':  
            'diagnostics = reforis_diagnostics:diagnostics'  
    },  
    # ...  
)
```

Listing 4.16: The plugin registration with `entry_points`.

4.6.2 Loading

When all installed plugins have an `entry_point` defined in `foris.plugins` section, then collecting all plugin Blueprints is very simple. Using `pkg_resources` [77] module it's possible to iterate through all `entry_points` having a certain name and load them. `iter_entry_points` function provides iterator over all globally installed modules' `entry_points`. The code example is shown on Listing 4.17.

```
import pkg_resources

def get_plugins():
    return [
        entry_point.load()
        for entry_point in pkg_resources.iter_entry_points(
            'foris.plugins'
        )
    ]
```

Listing 4.17: The plugin load via `entry_points`.

4.6.3 Static files

Static files (such as images, `.css` and `.js`) are files that are not dynamically generated and are sent to the client browser as is. All these files are stored into a separate Python module called `reforis_static`. It's done to provide single static root to the `lighttpd` server which is used as a production web server in this project.

4.7 Localization

The localization system was created in order to fulfill the non-functional requirement which is described on 35 page. The localization tools and techniques are described in this section.

4.7.1 Tooling

4.7.1.1 Flask-Babel

Flask-Babel is an extension to Flask that adds `i18n` support to any Flask application with the help of babel. It has built-in support for timezone specific date formatting. Also, it provides a simple and friendly interface to `gettext` system [78].

4.7.1.2 `gettext`

`gettext` is a set of well integrated tools that provide a framework within which other packages may produce multilingual messages. These tools include a runtime library for retrieval of translated messages and a library supporting parsing and creation of files that contain translated messages [79].

4.7.2 Messages creation

`gettext` produces `.pot` [80] files as a result of an extraction of translation messages from source files. These files become templates for `.po` files which contain translations for a particular language.

The advantage of the `.po` files is that they have a human-readable format. Moreover, Weblate service [36] has support of `.po` file format. Thus, it provides a simple web interface which allows an enthusiast to make translations of application text content.

Jinja2 templates, python code and JS code can contain text which should be translated in the reForis project. Babel supports all these code syntaxes. It can be configured to collect all translation strings into a single `.pot` file given regex-defined paths to files and their corresponding file types.

4.7.3 Messages processing

When `.po` files containing prepared translations are ready, they have to be compiled into `.mo` (Machine Object) files. `.mo` files are processed binary data. Thus, they can be quickly processed and used by a computer in order to swap original message with the one in native user language.

Flask-Babel and `gettext` process all Jinja2 templates and Python code. Getting translations to work in JS source code was a tiresome process. Flask-Babel can't process JS codes in real-time because the entire JS code runs on the client machine in a browser. A solution is using a small JavaScript library called `babel.js`. It's a simple library that provides a `gettext`-like translation interface [81].

The translations catalog is generated using `TranslationsHelper` object. It's a helper which is inherited from `babel.support.Translations` object in order to generate JSON translations dictionary with `babel.js` suitable format. Then it is uploaded into JS code with Jinja2 template system. This process is illustrated in the Listing 4.18.

```
babel.Translations
    .load({{ babel_catalog | safe }})
    .install();
```

Listing 4.18: The example of loading the translations.

4.8 Documentation

4.8.1 Flask application

Documentation of the reForis Flask application is done using Sphinx [?]. Sphinx is a set of tools for extracting, structuring and building documentation. It allows generating documentation in PDF, HTML, ePub and other formats.

Entire documentation of Python code is done as Python docstrings and it has been a part of the Python source code.

The Flask application documentation also contains plugin system documentation with explanation of plugin development and links to the demonstration plugin.

4.8.1.1 API endpoints documentation

Endpoints of API and Views Blueprints are also automatically extracted via Sphinx and the following plugins:

- `sphinxcontrib.httpdomain`
- `sphinxcontrib.autohttp.flask`
- `sphinxcontrib.autohttp.flaskqref`

The endpoints documentation is illustrated by request and response examples.

4.8.2 React

The React components are documented via React Styleguidist documentation system [82]. The great feature of this tool is a possibility to illustrate component documentation with code examples and rendered HTML code. Moreover, it provides an interactive playground which allows to quickly try a component. Also, it can automatically create documentation based on components `propTypes`.

It supports only interactive HTML format. But it's the best solution for React components documentation in author's opinion.

4.9 Deployment

4.9.1 TurrisOS and OpenWrt packages

TurrisOS is set of OpenWrt packages maintained by Turris team and contains Turris-specific packages and backports from upstream [7]. Thus, the new reForis package should be added to TurrisOS by adding OpenWrt-specific installation script. OpenWrt uses the makefile which has been transformed

into an object-oriented template [83]. An example of such makefile for reForis project is shown in the Listing 4.19.

4. IMPLEMENTATION

```
include $(TOPDIR)/rules.mk

PKG_NAME:=reforis
PKG_VERSION:=0.2.1
PKG_SOURCE_URL:=https://gitlab.labs.nic.cz/turris/reforis
# Another package definitions...

define Package/reforis
    URL:=https://gitlab.labs.nic.cz/turris/reforis
    TITLE:=reforis
    DEPENDS:= #... Dependency list
    MAINTAINER:=CZ.NIC <packaging@turris.cz>
endef

define Py3Package/reforis/install
    $(INSTALL_DIR) $(1)/usr/bin
    $(CP) $(PKG_INSTALL_DIR)/usr/bin/* $(1)/usr/bin/

    # Installation of a lighttpd configuration
    $(INSTALL_DIR) $(1)/etc/lighttpd/conf.d/
    $(INSTALL_DATA) ./files/reforis-config.lighttpd-conf\
        $(1)/etc/lighttpd/conf.d/reforis-config.conf

    $(INSTALL_DIR) $(1)/usr/share/foris
    $(INSTALL_BIN) ./files/lighttpd-dynamic-conf\
        $(1)/usr/share/reforis/lighttpd-dynamic-conf
    #...
endef

define Package/reforis/postinst
[ -n "$$IPKG_INSTROOT" ] || {
    /etc/init.d/lighttpd enable
    /usr/bin/maintain-lighttpd-restart
}
endef

define Package/reforis/description
Web administration interface for turris router.
endef

# Run prepared build script for Python 3
$(eval $(call Py3Package,reforis))
$(eval $(call BuildPackage,reforis))
$(eval $(call BuildPackage,reforis-src))
```


4.9.2 lighttpd configuration

Since lighttpd server has been chosen as a production server, lighttpd configuration is needed. The server configuration should be set using a couple of environment variables (such as `$DEBUG`, `$BUS`, `$CONTROLLER_ID`, etc.). Moreover, it's better to pass exact URL routes to the configuration.gversion of OS.

The author doesn't like this solution because of bad readability and high complexity of this bash script and considers to use Python script with Jinja2 template in the future.

4.9.3 Deployment to a testing branch

When installation scripts of an OpenWrt package are added to the `master` branch of the `turris-os-packages` [84] GitLab repository, the package is built automatically by Jenkins [85] automation server. If the build is successful, it is automatically deployed to devices running on any versions of TurrisOS. It is done with the updater system.

The reForis project has been deployed as an additional package. In the current Foris interface it can be installed in the packages configuration section. Also, the reForis package is available in the TurrisOS packages repository [86]. It can be downloaded and installed on any TurrisOS device as an OpenWRT package.

Testing

The testing stage is essential for any project, especially for a commercial project which is going to be used by real people. Tests allow to reduce the number of bugs and usability issues. As a result, it increases the users' satisfaction with the product.

Entire described test implementation is available in the reForis repository on CZ.NIC GitLab [50] or on the disk which is attached to this paper.

5.1 Backend

5.1.1 Router backend mocking

As was described in the previous parts of this thesis, the reForis project uses `foris-client` library. It is a helper for communication between `foris-controller` and the reForis application. This library is not present in the test environment because of lack of `foris-controller`. Thus, this library has to be mocked. The mocking is done by using `unittest.mock` [87] library module.

There was an issue of mocking Python module which is not available. The author found a solution using `surrogate.py` [88] library which can create mocks on the Python module level.

5.1.2 Views and API

Since Flask application is used as a layer between `foris-controller` and React client application, tests of the Python code are really simple and only check that all required views are exposed. Also, tests check if correct `foris-controller` module is called with an API request.

5.1.3 Authentication

The only part which is completely implemented in reForis Flask application is authentication. A low-quality implementation of authentication can potentially cause security issues. Thus, authentication logic should be covered by appropriate tests. The author has implemented a set of authentication tests in order to test the session attributes of a logged and unlogged user and test redirects in case an unlogged user tries to get access to any of the protected pages.

5.1.4 Coverage report

Name	Stmts	Miss	Cover
reforis/__init__.py	68	9	87%
reforis/__main__.py	14	14	0%
reforis/api.py	158	32	80%
reforis/auth.py	31	2	94%
reforis/backend.py	60	29	52%
reforis/cli.py	19	19	0%
reforis/config/__init__.py	0	0	100%
reforis/config/dev.py	6	0	100%
reforis/config/prod.py	8	0	100%
reforis/config/test.py	2	0	100%
reforis/guide.py	10	5	50%
reforis/locale.py	22	5	77%
reforis/plugins.py	3	0	100%
reforis/utils.py	6	0	100%
reforis/views.py	55	5	91%
TOTAL	462	120	74%

Listing 5.20: Coverage report of the Python code testing.

5.2 Frontend

5.2.1 Jest

Jest is a testing framework aimed at helping writing, running and maintaining JS code [89]. It allows to run tests in parallel using an isolated environment. Furthermore, it provides a simple API to write assertions with a possibility to

make snapshots of HTML code and compare them during tests. Jest is used in the project as a testing framework.

5.2.2 React hooks testing

The author, in the first place, tried to implement unit tests for React hooks. But then he figured out that it brings some complications because hooks are used only inside of components. Thus, mocked component environment is needed to get it done. Then author considered to use `(react-hooks-testing-library)` [90]. But after consulting the documentation, the author realized that hooks can be tested via components if they are created only for use inside certain components. Thanks to this brilliant idea and contributors to this library, it is possible to use fewer workarounds. Thus, `(react-hooks-testing-library)` wasn't used at all.

In this way `useForm` hook is tested inside `Form` wrapper component. Entire form logic is tested on the mocked component passed to the `Form`.

5.2.3 React components testing

Configuration forms are the main part of the react application. These components have validation and data processing logic inside that has to be tested. All these components are covered by tests with snapshots. It means that the rendered HTML code is compared against prerecorded snapshots of the same components. It's a simple way to assert the testing expectations. Moreover, it allows to simulate user behavior and compare a snapshot of a changed component after user manipulation. Also, all these components are tested for correctness and validity of the data they send to the server.. It is done using `jest-mock-axios` [91] library.

The configuration form components contain form element components, also known as pure components (such as inputs and buttons). It means that they do not have any logic inside and their resulting render completely depends on the arguments passed to them. All these pure components are also tested with snapshots.

Simulation of a user behavior and snapshots rendering is done using React Testing Library [92].

5.2.4 Coverage report

File	%Stmts	%Branch	%Funcs	%Lines
All files	71.85	70.3	63.58	71.91
common	71.21	73.68	64.71	70.77
connectionTest	77.78	55.56	77.78	77.78
dns	61.11	71.43	52.94	64.71
formContainer	79.17	71.43	74.07	79.17
guestNetwork	77.14	78.57	66.67	77.14
interfaces	75.86	62.16	74.19	75.29
lan	91.94	84	72.22	91.94
languagesDropdown	0	0	0	0
notifications	84.31	60	93.75	83.67
notifications/Notifications	87.5	75	87.5	87.5
notificationsSettings	76.92	65.22	50	76.56
overview	0	0	0	0
packages	70	100	61.11	70
password	54.39	46.88	68.42	54.39
reboot	0	100	0	0
regionAndTime	68	41.67	58.33	68
testUtils	100	100	100	100
updates	66.67	62.5	100	66.67
utils	100	100	100	100
wan	84.52	77.88	61.11	84.52
wifi	85.33	88.1	75	84.72

Listing 5.21: Coverage report of the JS code testing.

5.3 Integration tests

There is a possibility to write integration tests. Unfortunately, it forces a developer to create a special testing environment which simulates hardware used in Turrus routers (e.g. it can be a Docker container). Turrus team already has one, but it is no longer used and is not supported. Author considers creating such an environment in the future, but this project is going to be time-consuming and it has a low priority currently.

5.4 Usability testing

Usability testing of a graphical user interface is an approach to test how easily can a user control a GUI. The test is performed with real users in order to find issues with usability and efficiency. Results of the analysis of the usability test are used to fix usability issues in the next development iterations of the project.

5.4.1 Prototype stage

The first stage of Usability testing was done under the MI-NUR course under the direction of the supervisor of this paper Ing. Jiří Hunka. The first use test was done with a non-functional prototype which was described on page 47. The prototype was developed by members of the following team: Bodnar, B., Karola, A., Kryvosheienko, M., Laskov, B. and Samigullina, G.

The subsequent list is a restructured and rewritten summary of that work.

5.4.1.1 Changing a password is confusing

Description Foris and LuCI have two different passwords. When a user wants to change a password his is prompted to type in his current password. However, it is not clear, which password, Foris or LuCI, should that be, because both passwords are changed on the same form.

Possible solution To split the form into advanced configuration (LuCI) and Foris password forms or rename the “Current password” field.

5.4.1.2 Pressing the Enter key to submit the form

Possible solution Correct usage of HTML forms.

5.4.1.3 Password input value is not visible

Possible solution Add a “Show/Hide” password trigger button.

5.4.1.4 Packages are not alphabetically sorted

Possible solution To sort them. It can be better to provide a user with the text search.

5.4.1.5 Confusing color of input text

Description The color of typed text in the form input is gray (it is generally used for disabled form elements).

Possible solution Changing color of form inputs to black.

5.4.1.6 “Show all notifications” button name is confusing

Description “Show all notifications” buttons redirects a user to the page with notification settings and an incomplete list of notifications.

Possible solution Renaming “Show all notifications” to something else. Or extracting notification settings into a separate page.

5.4.1.7 Reboot button is hard to find

Description It is caused by the small size of the reboot section located at the bottom of the page.

Possible solution To put it on the top to be more visible.

Remark In the final solution, the “Reboot” button is moved to a separated page and it can be accessed from the main navigation menu.

5.4.2 Conclusion

Some of the above-described issues were fixed in the prototype stage. The author took them into account during the implementation of the final application in order to avoid replicating them.

5.5 Use test with functional application

5.5.1 Preparation

As Jakob Nielsen, Ph.D. mentions in one of his papers [93]: testing with only five users covers most of usability issues. It means that it is not essential to perform usability testing with a big group of users and it is better to select test subjects from the target audience.

After consulting with Turris team members it was discovered that Turris router audience mostly consists of people under 30 years of age. The Turris routers are more powerful than classic home routers, and users of Turris routers mostly have some experience with network device configuration. But also there is a group of people who are beginner enthusiasts.

5.5.2 Personas

As a result of the information above the author split target users into two following groups of personas.

5.5.2.1 Expert

Age: 20–30

Network configuration experience: expert.

Description: It may be a student or a Computer Science graduate.

5.5.2.2 Beginner

Age: 20–30

Network configuration experience: beginner.

Description: It may be a Computer Science freshman or an IT enthusiast. The main prerequisite is that the user has not had experience in the network device configuration.

Seven volunteers were selected (4 experts and 3 beginners).

5.5.3 Script

The usability test script was created based on the issues which were found during the preceding usability testing and functionality walkthrough. The script is available in the appendix of this paper on page 163. This script is accompanied by two quizzes in order to better understand user emotional state before and after tests and to get more information about the experience of the user or some improvement suggestions.

5.5.4 Analysis

Video recordings and screencasts of user behavior during the usability test are used in the analysis and are available on the disk which is attached to the paper.

During usability testing analysis subsequent issues were found.

5.5.4.1 The Turris logo in the header is not clickable

Description It is used by users who are used to that logo linking to the home page.

Possible solution To add a link to the logo.

5.5.4.2 Some countries are placed in the wrong region in the timezone selection

Description Spain is located under the African region.

Possible solution To use another library which has the correct country and region placement.

5.5.4.3 Countries and cities are unordered

Possible solution Add alphabetical order to countries and cities.

Remark It had been fixed but another issue was found. The JS string comparison works incorrectly with some languages. Thus, it should have some multilingual solution.

5.5.4.4 “Notifications” title in the dropdown notifications menu is not clickable

Description When a user wants to show all notifications there is only one way to do it by clicking on a some notification. It’s better to provide some way to do it when there are no notifications.

Possible solution To add a link to the notifications page to the title in the dropdown menu.

Remark It has been fixed on the spot.

5.5.4.5 “Notifications” title in dropdown notifications menu is not clickable

Description When a user wants to show all notifications there is only one way to do it with clicking on the certain notification. It’s better to provide some approach to do it when there are no notifications.

Possible solution To add a link to the notifications page on the title in dropdown menu.

Remark It have been fixed on the spot.

5.5.4.6 DNS forwarder option does not have help text

Description Some of the beginner users may not know what DNS forwarder means. It is better to add a remark to this option.

Possible solution Add help text to DNS forwarder option.

5.5.4.7 Link to advanced administration is confusing

Description It's not clear to users that "Advanced administration" menu item is linked to LuCI configuration interface.

Possible solution Renaming of the "Advanced administration" button. It can be named "Advanced administration (LuCI)".

5.5.4.8 Packages vs plugins

Description The difference between packages and plugins may be confusing.

Possible solution Better naming for these menu items. It may be named "Packages" and "Package management".

5.5.4.9 Guest Wi-Fi setting is hardly visible.

Description The configuration of the guest Wi-Fi network is located at the bottom of the Wi-Fi module configuration and it is not highlighted.

Possible solution Put guest Wi-Fi settings under their own title to make them more noticeable.

Some of these issues were fixed on the spot. Other issues should be discussed with Turrís team members in order to find the best solutions and will be fixed in the next development iterations.

5.5.5 Improvement suggestions

Despite the high praise of UI experience users had a few subsequent improvement suggestions.

- Dashboard with network load, CPU and memory usage on the main page.
- List of connected devices (via Wi-Fi) on the main page with the possibility to disconnect a selected device.
- Possibility to select UTC timezone.

All these suggestions will be passed to the Turrís team members and may be added in the future.

Conclusion

Conclusion

The goal of this project was to redesign and implement the new Foris router administration web interface in order to create better architecture and user experience based on the analysis and requirements. The project is done using modern and simplified plugin system in the interest of motivating third-party developers to contribute.

The resulting product code has been tested and added the test TurrisOS release. Appropriate usability testing was performed and evaluated. Moreover, some of the discovered issues were fixed on the spot. Thus, Foris is ready for the next development iteration.

In view of the above, I believe that I have fulfilled all goals.

This is my first production grade project which I was entirely responsible for. It was really challenging, although positive feedback from Turris team and users makes all the challenges worth it. I will be glad to follow through this project and see it in the production stage.

Bibliography

- [1] OpenWrt Project: Welcome to the OpenWrt Project . Feb 2019, [Online; accessed 13. Feb. 2019]. Available from: <https://openwrt.org>
- [2] OpenWRT / LuCI . Feb 2019, [Online; accessed 13. Feb. 2019]. Available from: <https://github.com/openwrt/luci/wiki>
- [3] first commit of somewhat-stable code (see comment below) (c42a021a) · Commits · Turrís / foris. Jun 2019, [Online; accessed 27. Jun. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/foris/commit/c42a021a7f6249ed31fe1bc87d7e19aeb1ed8479>
- [4] Nielsen, J.; Molich, R. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, 1990, pp. 249–256.
- [5] Turrís | Turrís . Mar 2019, [Online; accessed 18. Mar. 2019]. Available from: <https://www.turrís.cz/en>
- [6] Installation of LTE modem into Turrís Omnia router [Project: Turrís] . Feb 2019, [Online; accessed 13. Feb. 2019]. Available from: https://doc.turrís.cz/doc/en/howto/lte_modem_install
- [7] Turrís OS . Feb 2019, [Online; accessed 13. Feb. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/openwrt>
- [8] UCI (Unified Configuration Interface) – Technical Reference [Old OpenWrt Wiki] . Feb 2019, [Online; accessed 15. Feb. 2019]. Available from: <https://oldwiki.archive.openwrt.org/doc/techref/uci>
- [9] The UCI System [Old OpenWrt Wiki] . Feb 2019, [Online; accessed 15. Feb. 2019]. Available from: <https://oldwiki.archive.openwrt.org/doc/uci>

BIBLIOGRAPHY

- [10] The UCI System [Old OpenWrt Wiki] . Mar 2019, [Online; accessed 18. Mar. 2019]. Available from: <https://oldwiki.archive.openwrt.org/doc/uci>
- [11] openwrt/luci . Feb 2019, [Online; accessed 15. Feb. 2019]. Available from: <https://github.com/openwrt/luci/wiki/ModulesHowTo>
- [12] The Programming Language Lua . Jan 2019, [Online; accessed 15. Feb. 2019]. Available from: <https://www.lua.org>
- [13] openwrt/luci . Feb 2019, [Online; accessed 15. Feb. 2019]. Available from: <https://github.com/openwrt/luci/wiki/Templates>
- [14] COMPAL CH7465LG Modem / User manual . Mar 2019, [Online; accessed 24. Mar. 2019]. Available from: <https://business.upc.cz/pdf/soho/UPC-Mercury-modem-uzivatelsky-manual-v5.pdf>
- [15] Eberhardt, G.; Bácsi, G.; et al. Security evaluation of the Compal Broadband networks CH7465LG "Mercur" Modem. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, SEARCH-LAB, 2016, pp. 12–13. Available from: https://www.search-lab.hu/media/Compal_CH7465LG_Evaluation_Report_1.1.pdf
- [16] Documentation [Project: Turrís] . Apr 2019, [Online; accessed 1. Mar. 2019]. Available from: <https://doc.turris.cz/doc/en/start>
- [17] Hardware :: Project:Turrís . Feb 2019, [Online; accessed 11. Feb. 2019]. Available from: <https://project.turris.cz/en/hardware>
- [18] Turrís Omnia . Feb 2019, [Online; accessed 12. Feb. 2019]. Available from: <https://omnia.turris.cz/en>
- [19] Turrís - Technical specification of each MOX module . Feb 2019, [Online; accessed 12. Feb. 2019]. Available from: <https://mox.turris.cz/en/specification>
- [20] Turrís / foris-controller . Mar 2019, [Online; accessed 4. Mar. 2019]. Available from: <https://gitlab.labs.nic.cz/turris/foris-controller>
- [21] Welcome to Python.org . Mar 2019, [Online; accessed 4. Mar. 2019]. Available from: <https://www.python.org>
- [22] Bottle: Python Web Framework — Bottle 0.13-dev documentation . Mar 2019, [Online; accessed 4. Mar. 2019]. Available from: <https://bottlepy.org/docs/dev>
- [23] Home - Lighttpd - fly light . Jan 2019, [Online; accessed 4. Mar. 2019]. Available from: <https://www.lighttpd.net>

-
- [24] Turrís / foris . Apr 2019, [Online; accessed 5. Mar. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/foris>
- [25] Jinja2 Documentation (2.10) . May 2018, [Online; accessed 5. Mar. 2019]. Available from: <http://jinja.pocoo.org/docs/2.10>
- [26] vex . Nov 2017, [Online; accessed 7. Apr. 2019]. Available from: <https://github.hubspot.com/vex>
- [27] Code minification · WebPlatform Docs . Feb 2017, [Online; accessed 7. Mar. 2019]. Available from: <https://webplatform.github.io/docs/concepts/programming/javascript/minification>
- [28] WebSockets . Mar 2019, [Online; accessed 10. Mar. 2019]. Available from: <https://developer.mozilla.org/en-US/docs/Glossary/WebSockets>
- [29] ubus (OpenWrt micro bus architecture) [Old OpenWrt Wiki] . Mar 2019, [Online; accessed 3. Mar. 2019]. Available from: <https://oldwiki.archive.openwrt.org/doc/techref/ubus>
- [30] MQTT . Mar 2019, [Online; accessed 4. Mar. 2019]. Available from: <http://mqtt.org>
- [31] Eclipse Mosquitto . Jan 2018, [Online; accessed 25. Mar. 2019]. Available from: <https://mosquitto.org>
- [32] Turrís / foris-client . Mar 2019, [Online; accessed 18. Mar. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/foris-client>
- [33] Kaliski, B. PKCS 5: Password-Based Cryptography Specification Version 2.0 . Apr 2019, [Online; accessed 21. Apr. 2019]. Available from: <https://tools.ietf.org/html/rfc2898>
- [34] JSON Schema . Mar 2019, [Online; accessed 11. Mar. 2019]. Available from: <https://json-schema.org>
- [35] Turrís / foris-schema . Mar 2019, [Online; accessed 11. Mar. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/foris-schema>
- [36] Čihař, M. Weblate - web-based localization . Jun 2019, [Online; accessed 16. Jun. 2019]. Available from: <https://weblate.org/en-gb>
- [37] Inc., S. Top 10 Python Web Frameworks to Learn in 2018 . *Hacker Noon*, Apr 2019. Available from: <https://hackernoon.com/top-10-python-web-frameworks-to-learn-in-2018-b2ebab969d1a>
- [38] Welcome | Flask (A Python Microframework) . Apr 2019, [Online; accessed 8. Apr. 2019]. Available from: <http://flask.pocoo.org>

BIBLIOGRAPHY

- [39] Extensions Registry | Flask (A Python Microframework) . Apr 2019, [Online; accessed 8. Apr. 2019]. Available from: <http://flask.pocoo.org/extensions>
- [40] Modular Applications with Blueprints — Flask 1.0.2 documentation . May 2018, [Online; accessed 8. Apr. 2019]. Available from: <http://flask.pocoo.org/docs/1.0/blueprints>
- [41] Vanilla JS . Apr 2019, [Online; accessed 8. Apr. 2019]. Available from: <http://vanilla-js.com>
- [42] React – A JavaScript library for building user interfaces . Apr 2019, [Online; accessed 8. Mar. 2019]. Available from: <https://reactjs.org>
- [43] Neuhaus, J. Angular vs. React vs. Vue: A 2017 comparison . *Medium*, Sep 2018. Available from: <https://medium.com/unicorn-supplies/angular-vs-react-vs-vue-a-2017-comparison-c5c52d620176>
- [44] What is Babel? · Babel . Apr 2019, [Online; accessed 8. Apr. 2019]. Available from: <https://babeljs.io/docs/en>
- [45] JSX | XML-like syntax extension to ECMAScript . Oct 2017, [Online; accessed 8. Apr. 2019]. Available from: <https://facebook.github.io/jsx>
- [46] Otto, Mark. and Jacob, Thornton. and Bootstrap contributors . Bootstrap . Mar 2019, [Online; accessed 8. Apr. 2019]. Available from: <https://getbootstrap.com>
- [47] Flask-Session — Flask-Session 0.3.0 documentation . Apr 2019, [Online; accessed 21. Apr. 2019]. Available from: <https://pythonhosted.org/Flask-Session>
- [48] Atkins, D.; Austein, R. Threat Analysis of the Domain Name System (DNS) . Apr 2019, [Online; accessed 15. Apr. 2019]. Available from: <https://tools.ietf.org/html/rfc3833>
- [49] Prototypes, Specifications, and Diagrams in One Tool | Axure Software . Apr 2019, [Online; accessed 16. Apr. 2019]. Available from: <https://www.axure.com>
- [50] Turrís / reforis . May 2019, [Online; accessed 1. May 2019]. Available from: <https://gitlab.labs.nic.cz/turris/reforis>
- [51] libfuse/sshfs . Apr 2019, [Online; accessed 30. Apr. 2019]. Available from: <https://github.com/libfuse/sshfs>
- [52] rsync . May 2019, [Online; accessed 1. May 2019]. Available from: <https://rsync.samba.org>

-
- [53] PyCharm . May 2019, [Online; accessed 1. May 2019]. Available from: <https://www.jetbrains.com/pycharm>
- [54] What is eMMC Memory – software support | Reliance Nitro . May 2019, [Online; accessed 1. May 2019]. Available from: <https://www.datalight.com/solutions/technologies/emmc/what-is-emmc>
- [55] Zhang, T.; Zuck, A.; et al. Flash Drive Lifespan *is* a Problem. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, The University of North Carolina at Chapel Hill & Technion Israel Institute of Technology, 2017, pp. 1–2. Available from: <http://www.cs.technion.ac.il/~dan/papers/fbrick-hotos-2017.pdf>
- [56] OpenWrt Project: Filesystems . May 2019, [Online; accessed 1. May 2019]. Available from: <https://openwrt.org/docs/techref/filesystems>
- [57] Git . May 2019, [Online; accessed 1. May 2019]. Available from: <https://git-scm.com>
- [58] CZ.NIC company GitLab . May 2019, [Online; accessed 1. May 2019]. Available from: <https://gitlab.labs.nic.cz>
- [59] The first single application for the entire DevOps lifecycle - GitLab . May 2019, [Online; accessed 1. May 2019]. Available from: <https://about.gitlab.com>
- [60] GitLab Continuous Integration Delivery . Jun 2019, [Online; accessed 12. Jun. 2019]. Available from: <https://about.gitlab.com/product/continuous-integration>
- [61] Booch, G.; Maksimchuk, R.; et al. *Object-oriented Analysis and Design with Applications, Third Edition*. Addison-Wesley Professional, third edition, 2007, ISBN 9780201895513.
- [62] What is a Container? | Docker . May 2019, [Online; accessed 12. Jun. 2019]. Available from: <https://www.docker.com/resources/what-container>
- [63] Logilab. Pylint - code analysis for Python | www.pylint.org . Apr 2018, [Online; accessed 13. Jun. 2019]. Available from: <https://www.pylint.org>
- [64] pycodestyle 's documentation — pycodestyle 2.5.0 documentation . Mar 2019, [Online; accessed 13. Jun. 2019]. Available from: <https://pycodestyle.readthedocs.io/en/latest>

- [65] PEP 8 – Style Guide for Python Code . Jun 2019, [Online; accessed 13. Jun. 2019]. Available from: <https://www.python.org/dev/peps/pep-0008>
- [66] ESLint - Pluggable JavaScript linter . Jun 2019, [Online; accessed 13. Jun. 2019]. Available from: <https://eslint.org>
- [67] Application Factories — Flask 1.0.2 documentation . May 2018, [Online; accessed 30. Apr. 2019]. Available from: <http://flask.pocoo.org/docs/1.0/patterns/appfactories>
- [68] pallets/werkzeug . May 2019, [Online; accessed 1. May 2019]. Available from: <https://github.com/pallets/werkzeug>
- [69] foris __ ws/authentication/filesystem.py . master . Turris / foris-ws . May 2019, [Online; accessed 1. May 2019]. Available from: https://gitlab.labs.nic.cz/turris/foris-ws/blob/master/foris_ws/authentication/filesystem.py
- [70] webpack. Jun 2019, [Online; accessed 22. Jun. 2019]. Available from: <https://webpack.js.org>
- [71] File Structure – React . May 2019, [Online; accessed 1. May 2019]. Available from: <https://reactjs.org/docs/faq-structure.html>
- [72] Introducing Hooks – React. Jun 2019, [Online; accessed 28. Jun. 2019]. Available from: <https://reactjs.org/docs/hooks-intro.html>
- [73] facebook/prop-types . May 2019, [Online; accessed 2. May 2019]. Available from: <https://github.com/facebook/prop-types>
- [74] Turris / reforis-diagnostics . Jun 2019, [Online; accessed 18. Jun. 2019]. Available from: <https://gitlab.labs.nic.cz/turris/reforis-diagnostics>
- [75] 2. Writing the Setup Script — Python 3.7.3 documentation . Jun 2019, [Online; accessed 18. Jun. 2019]. Available from: <https://docs.python.org/3/distutils/setupscript.html>
- [76] Package Discovery and Resource Access using pkg __ resources — setuptools 41.0.1 documentation . Jun 2019, [Online; accessed 18. Jun. 2019]. Available from: https://setuptools.readthedocs.io/en/latest/pkg_resources.html#entry-points
- [77] Package Discovery and Resource Access using pkg __ resources — setuptools 41.0.1 documentation . Jun 2019, [Online; accessed 18. Jun. 2019]. Available from: https://setuptools.readthedocs.io/en/latest/pkg_resources.html

-
- [78] Flask-Babel — Flask Babel 1.0 documentation . Jun 2019, [Online; accessed 16. Jun. 2019]. Available from: <https://pythonhosted.org/Flask-Babel>
- [79] gnu.org . Jun 2019, [Online; accessed 16. Jun. 2019]. Available from: <https://www.gnu.org/software/gettext>
- [80] GNU gettext utilities: PO Files . Jun 2019, [Online; accessed 16. Jun. 2019]. Available from: https://www.gnu.org/software/gettext/manual/html_node/PO-Files.html#PO-Files
- [81] python-babel/babel . Jun 2019, [Online; accessed 17. Jun. 2019]. Available from: <https://github.com/python-babel/babel/blob/master/contrib/babel.js>
- [82] React Styleguidist: isolated React component development environment with a living style guide. Jun 2019, [Online; accessed 27. Jun. 2019]. Available from: <https://react-styleguidist.js.org>
- [83] OpenWrt Project: Creating packages. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://openwrt.org/docs/guide-developer/packages>
- [84] cznic/foris/reforis · master · Turrís / turrís-os-packages. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://gitlab.labs.nic.cz/turrís/turrís-os-packages/tree/master/cznic/foris/reforis>
- [85] Jenkins. Jun 2019, [Online; accessed 27. Jun. 2019]. Available from: <https://jenkins.io>
- [86] TurrísPackages repository. Jun 2019, [Online; accessed 27. Jun. 2019]. Available from: <https://repo.turrís.cz/hbd/omnia/packages/turríspackages>
- [87] unittest.mock — mock object library — Python 3.7.4rc1 documentation. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://docs.python.org/3/library/unittest.mock.html>
- [88] ikostia/surrogate. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://github.com/ikostia/surrogate/blob/master/surrogate.py>
- [89] Jest · Delightful JavaScript Testing. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://jestjs.io>
- [90] testing-library/react-hooks-testing-library. Jun 2019, [Online; accessed 23. Jun. 2019]. Available from: <https://github.com/testing-library/react-hooks-testing-library>

BIBLIOGRAPHY

- [91] jest-mock-axios. Jun 2019, [Online; accessed 24. Jun. 2019]. Available from: <https://www.npmjs.com/package/jest-mock-axios>
- [92] Testing Library · Simple and complete testing utilities that encourage good testing practices. Jun 2019, [Online; accessed 24. Jun. 2019]. Available from: <https://testing-library.com>
- [93] Nielsen, J.; Landauer, T. K. A mathematical model of the finding of usability problems. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, ACM, 1993, pp. 206–213.

Original Foris screenshots

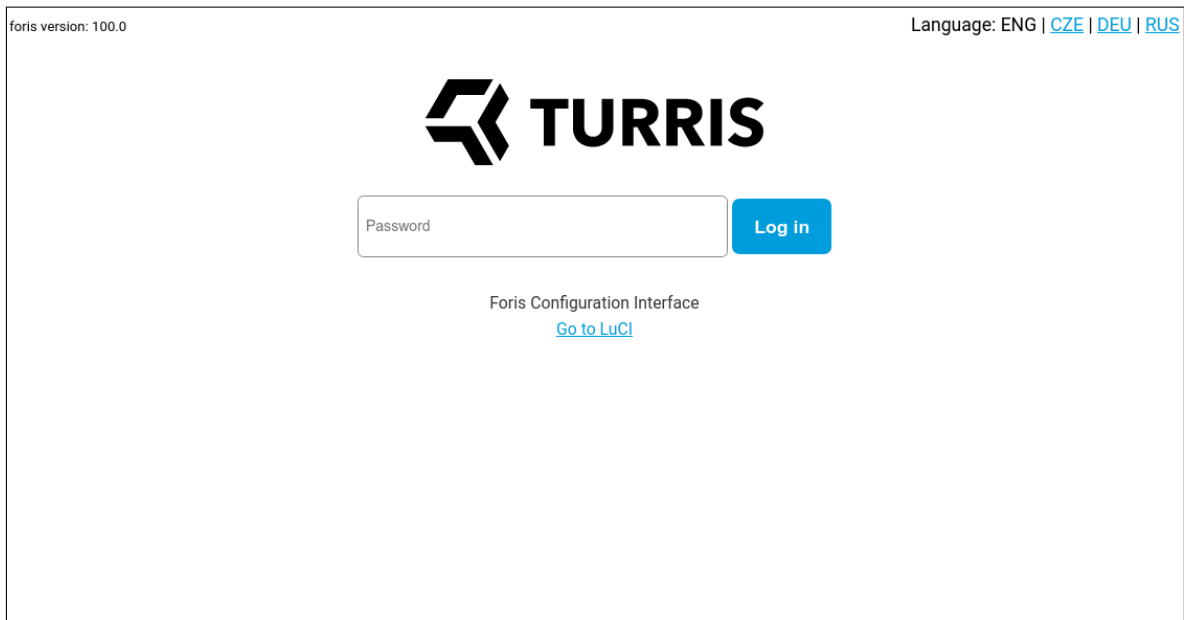


Figure A.1: *Original Foris. Screenshot of the login page.*

A. ORIGINAL FORIS SCREENSHOTS

TURRIS

Turris OMNIA
foris version: 100.0

NOTIFICATIONS 3

PASSWORD

REMOTE ACCESS

NETWORK INTERFACES

WAN

LAN

GUEST NETWORK

REGION AND TIME

DNS

WI-FI

MAINTENANCE

UPDATER

STORAGE

DIAGNOSTICS

ABOUT

ENGLISH LOG OUT

Notifications

Following notifications occurred and haven't been dismissed since last reboot.

- News from 2019/06/25 00:42:15**
Example_news_notification!
- Update from 2019/06/25 00:42:25**
Example_updates_notification!
- Notification from 2019/06/25 00:43:03**
Reboot_is_required!
[Reboot now](#)

[Dismiss All](#)

Figure A.2: Original Foris. Screenshot of the notifications page.

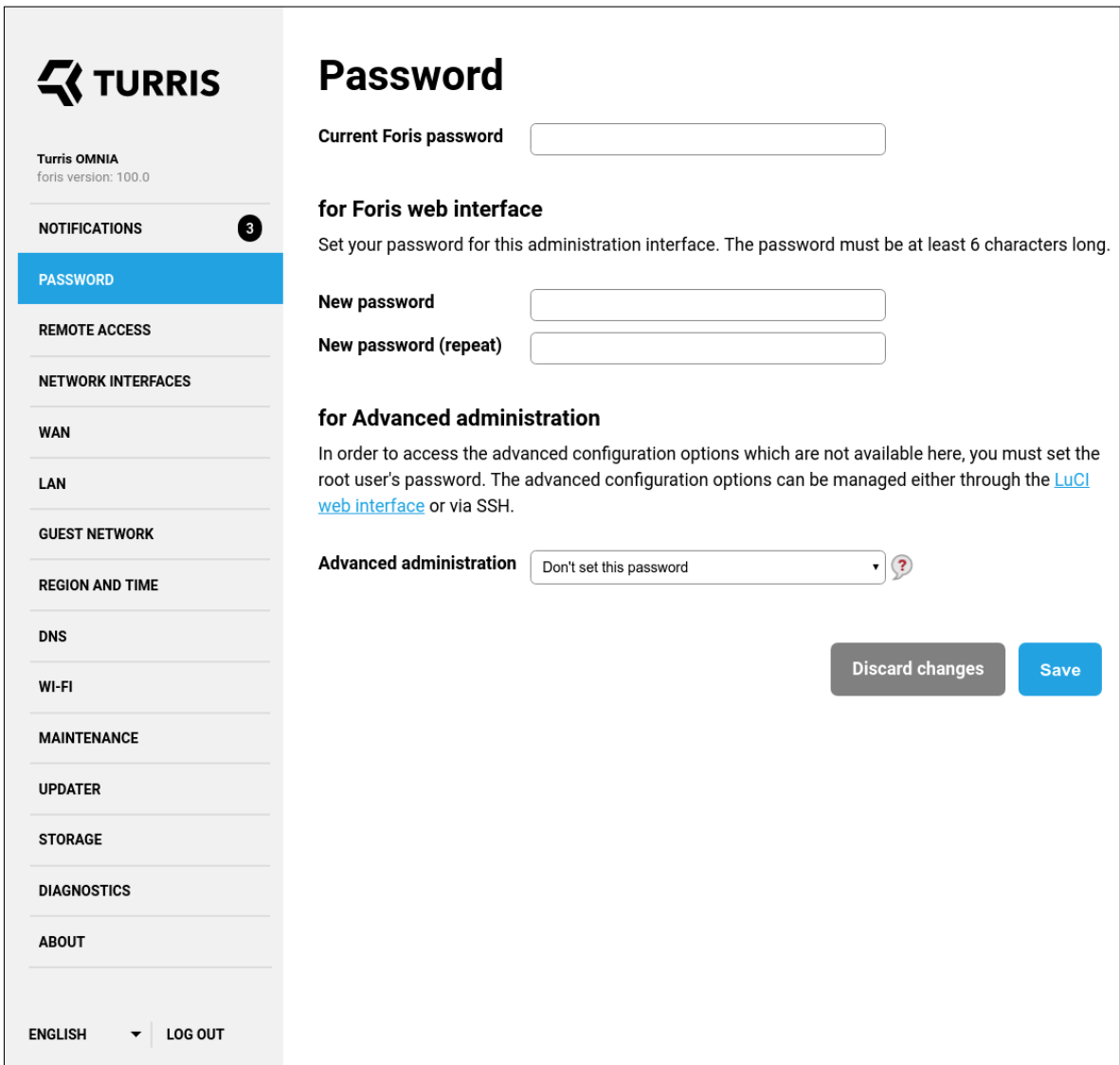


Figure A.3: Original Foris. Screenshot of the password settings page.

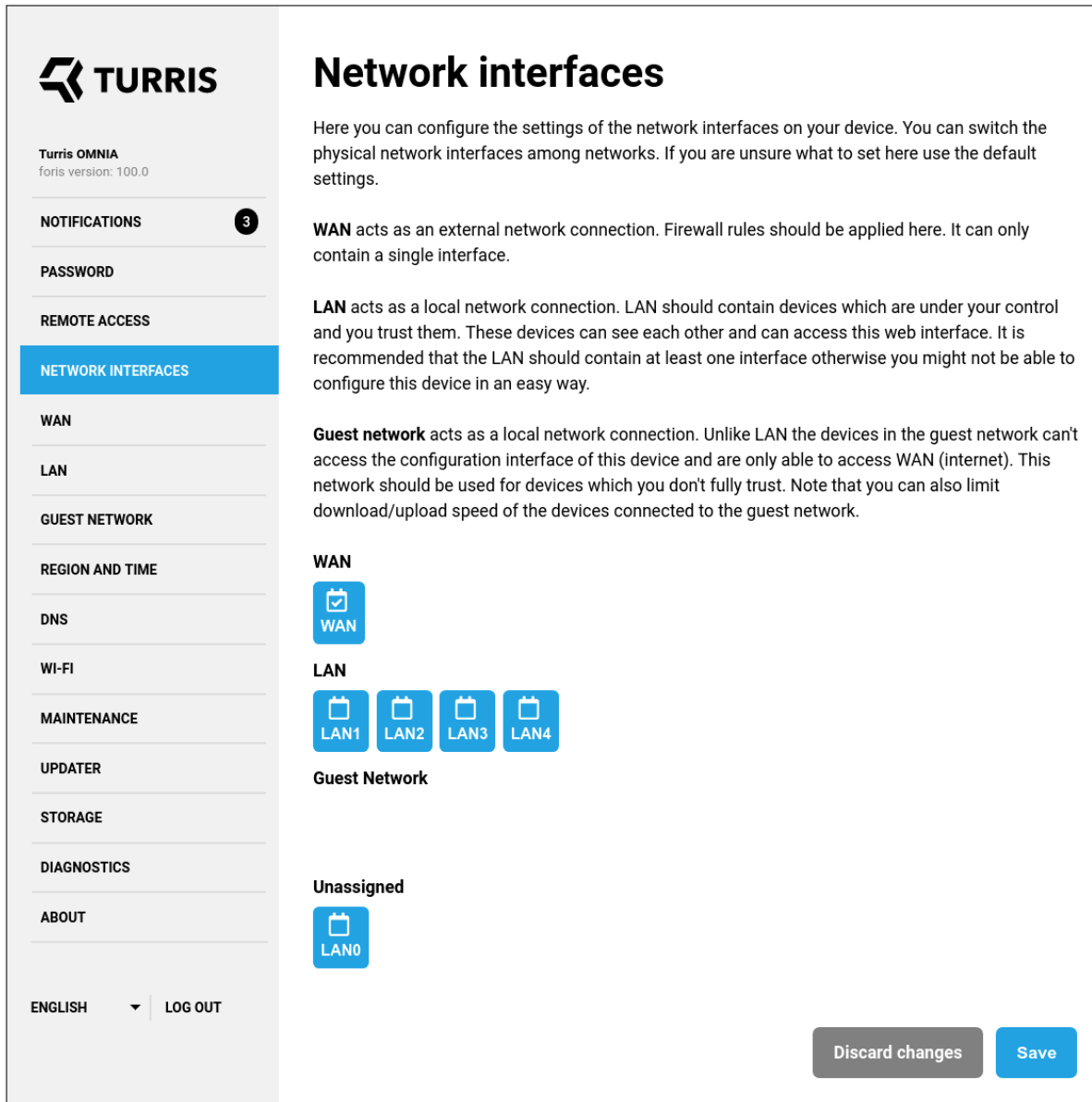



Figure A.4: *Original Foris. Screenshot of the network interfaces page.*



Turris OMNIA
foris version: 100.0

- NOTIFICATIONS 3
- PASSWORD
- REMOTE ACCESS
- NETWORK INTERFACES
- WAN
- LAN
- GUEST NETWORK
- REGION AND TIME
- DNS
- WI-FI
- MAINTENANCE
- UPDATER
- STORAGE
- DIAGNOSTICS
- ABOUT

ENGLISH ▼ | LOG OUT

WAN

Here you specify your WAN port settings. Usually, you can leave this options untouched unless instructed otherwise by your internet service provider. Also, in case there is a cable or DSL modem connecting your router to the network, it is usually not necessary to change this setting.

IPv4 protocol

DHCP hostname

IPv6 protocol

Custom DUID

Custom MAC address

Discard changes
Save

Connection test

Here you can test you connection settings. Remember to click on the **Save** button before running the test. Note that sometimes it takes a while before the connection is fully initialized. So it might be useful to wait for a while before running this test.

Test type	Status
IPv4 connectivity	✓
IPv4 gateway connectivity	✓
IPv6 connectivity	✓
IPv6 gateway connectivity	✓

Test connection

Figure A.5: *Original Foris. Screenshot of the WAN configuration page.*

TURRIS

Turris OMNIA
foris version: 100.0

NOTIFICATIONS **3**

PASSWORD

REMOTE ACCESS

NETWORK INTERFACES

WAN

LAN

GUEST NETWORK

REGION AND TIME

DNS

WI-FI

MAINTENANCE

UPDATER

STORAGE

DIAGNOSTICS

ABOUT

ENGLISH ▾ | LOG OUT

LAN

! All network interfaces of this network are currently down.

This section contains settings for the local network (LAN). The provided defaults are suitable for most networks.

Note: If you change the router IP address, all computers in LAN, probably including the one you are using now, will need to obtain a **new IP address** which does **not** happen **immediately**. It is recommended to disconnect and reconnect all LAN cables after submitting your changes to force the update. The next page will not load until you obtain a new IP from DHCP (if DHCP enabled) and you might need to **refresh the page** in your browser.

LAN mode Router ?

Router IP address 192.168.1.1 ?

Network netmask 255.255.255.0 ?

Enable DHCP ?

Enable this option to automatically assign IP addresses to the devices connected to the router.

Discard changes Save

Figure A.6: *Original Foris. Screenshot of the LAN configuration page.*

TURRIS

Turris OMNIA
foris version: 100.0

NOTIFICATIONS **3**

PASSWORD

REMOTE ACCESS

NETWORK INTERFACES

WAN

LAN

GUEST NETWORK

REGION AND TIME

DNS

WI-FI

MAINTENANCE

UPDATER

STORAGE

DIAGNOSTICS

ABOUT

ENGLISH ▾ | LOG OUT

Guest network

! This network currently doesn't contain any devices. The changes you make here will become fully functional after you assign a network interface to this network.

Guest network is used for [guest Wi-Fi](#). It is separated from your ordinary LAN. Devices connected to this network are allowed to access the internet, but are not allowed to access the configuration interface of the this device nor the devices in LAN.

Enable guest network

Router IP in guest network ?

Guest network netmask ?

Enable DHCP ?

Guest Lan QoS ?

Discard changes Save

Figure A.7: *Original Foris. Screenshot of the guest network configuration page.*

TURRIS

Turris OMNIA
foris version: 100.0

- NOTIFICATIONS 3
- PASSWORD
- REMOTE ACCESS
- NETWORK INTERFACES
- WAN
- LAN
- GUEST NETWORK
- REGION AND TIME
- DNS**
- WI-FI
- MAINTENANCE
- UPDATER
- STORAGE
- DIAGNOSTICS
- ABOUT

ENGLISH ▾ | LOG OUT

DNS

Router Turris uses its own DNS resolver with DNSSEC support. It is capable of working independently or it can forward your DNS queries your internet service provider's DNS resolver.

The following setting determines the behavior of the DNS resolver. Usually, it is better to use the ISP's resolver in networks where it works properly. If it does not work for some reason, it is necessary to use direct resolving without forwarding.

In rare cases ISP's have improperly configured network which interferes with DNSSEC validation. If you experience problems with DNS, you can **temporarily** disable DNSSEC validation to determine the source of the problem. However, keep in mind that without DNSSEC validation, you are vulnerable to DNS spoofing attacks! Therefore we **recommend keeping DNSSEC turned on** and resolving the situation with your ISP as this is a serious flaw on their side.

Use forwarding

DNS Forwarder CZ.NIC (TLS) ▾

Disable DNSSEC

Enable DHCP clients in DNS ?


Discard changes **Save**

Connection test

Here you can test your internet connection. This test is also useful when you need to check that your DNS resolving works as expected. Remember to click on the **Save** button if you changed your forwarder setting.

Test type	Status
DNS	✓
DNSSEC	✗

Figure A.8: Original Foris. Screenshot of the DNS configuration page.



Turris OMNIA
foris version: 100.0

- NOTIFICATIONS 3
- PASSWORD
- REMOTE ACCESS
- NETWORK INTERFACES
- WAN
- LAN
- GUEST NETWORK
- REGION AND TIME
- DNS
- WI-FI
- MAINTENANCE
- UPDATER
- STORAGE
- DIAGNOSTICS
- ABOUT

ENGLISH ▼ | LOG OUT

Wi-Fi

If you want to use your router as a Wi-Fi access point, enable Wi-Fi here and fill in an SSID (the name of the access point) and a corresponding password. You can then set up your mobile devices, using the QR code available within the form.

Wi-Fi settings

Enable Wi-Fi 1

SSID



Hide SSID ?

Wi-Fi mode 2.4 GHz (g) 5 GHz (a) ?

802.11n/ac mode ?

Network channel ?

Network password ?

Enable guest Wi-Fi ?

Enable Wi-Fi 2

If a number of wireless cards doesn't match, you may try to reset the Wi-Fi settings. Note that this will remove the current Wi-Fi configuration and restore the default values.

Figure A.9: *Original Foris. Screenshot of the Wi-Fi configuration page.*

TURRIS
Turris OMNIA
foris version: 100.0

NOTIFICATIONS **3**

PASSWORD

REMOTE ACCESS

NETWORK INTERFACES

WAN

LAN

GUEST NETWORK

REGION AND TIME

DNS

WI-FI

MAINTENANCE

UPDATER

STORAGE

DIAGNOSTICS

ABOUT

ENGLISH ▾ | LOG OUT

Region and time

It is important for your device to have the correct time set. If your device's time is delayed, the procedure of SSL certificate verification might not work correctly.

Region settings

Please select the timezone the router is being operated in. Correct setting is required to display the right time and for related functions.

Continent or ocean

Country

Timezone

Time settings

Time should be up-to-date otherwise DNS and other services might not work properly.

How to set time


NTP servers:

- 217.31.202.100
- 195.113.144.201
- 195.113.144.238
- 2001:1488:ffff::100
- ntp.nic.cz
- 0.openwrt.pool.ntp.org
- 1.openwrt.pool.ntp.org
- 2.openwrt.pool.ntp.org
- 3.openwrt.pool.ntp.org

Update time

Discard changes **Save**

Figure A.10: *Original Foris. Screenshot of the region and time settings page.*



Turris OMNIA
foris version: 100.0

- NOTIFICATIONS 3
- PASSWORD
- REMOTE ACCESS
- NETWORK INTERFACES
- WAN
- LAN
- GUEST NETWORK
- REGION AND TIME
- DNS
- WI-FI
- MAINTENANCE
- UPDATER
- STORAGE
- DIAGNOSTICS
- ABOUT

ENGLISH ▼ | LOG OUT

Maintenance

Notifications and automatic restarts

You can set the router to notify you when a specific event occurs, for example when a reboot is required, no space is left on device or an application update is installed. You can use Turris servers to send these emails. Alternatively, if you choose to use a custom server, you must enter some additional settings. These settings are the same as you enter in your email client and you can get them from the provider of your email inbox. In that case, because of security reasons, it is recommended to create a dedicated account for your router.

Also, when an automatic restart is required, you can specify the time you want it to occur. If you have email notifications enabled, you can also specify the interval between notification and automatic restart.

Notifications settings

Enable notifications

Automatic restarts after software update

Delay (days) ?

Reboot time ?

[Save](#)

Configuration backup

If you need to save the current configuration of this device, you can download a backup file. The configuration is saved as an unencrypted compressed archive (.tar.bz2). Passwords for this configuration interface and for the advanced configuration are not included in the backup.

[Download configuration backup](#)

Configuration restore

To restore the configuration from a backup file, upload it using following form. Keep in mind that IP address of this device might change during the process, causing unavailability of this interface.

Backup file No file chosen


[Restore from backup](#)

Device reboot

If you need to reboot the device, click on the following button. The reboot process takes approximately 30 seconds, you will be required to log in again after the reboot.

[Reboot](#)

Figure A.11: *Original Foris. Screenshot of the administration page.*



Turris OMNIA
foris version: 100.0

- NOTIFICATIONS 3
- PASSWORD
- REMOTE ACCESS
- NETWORK INTERFACES
- WAN
- LAN
- GUEST NETWORK
- REGION AND TIME
- DNS
- WI-FI
- MAINTENANCE
- UPDATER
- STORAGE
- DIAGNOSTICS
- ABOUT

ENGLISH ▾ | LOG OUT

Updater

Updater is a service that keeps all TurrisOS software up to date. Apart from the standard installation, you can optionally select bundles of additional software that'd be installed on the router. This software can be selected from the following list. Please note that only software that is part of TurrisOS or that has been installed from a package list is maintained by Updater. Software that has been installed manually or using opkg is not affected.

One of the most important features of router Turris are automatic system updates. Thanks to this function your router's software stays up to date and offers better protection against attacks from the Internet.

It is **highly recommended** to have this feature **turned on**. If you decide to disable it, be warned that this might weaken the security of your router and network in case flaws in the software are found.

By turning the automatic updates on, you agree to this feature's license agreement. More information is available [here](#).

- Use automatic updates (recommended)
- Turn automatic updates off

Update approvals

Update approvals can be useful when you want to make sure that updates won't harm your specific configuration. You can e.g. install updates when you're prepared for a possible rollback to a previous snapshot and deny the questionable update temporarily. It isn't possible to decline the update forever and it will be offered to you again together with the next package installation.

- Automatic installation ?
- Delayed updates ?
- Update approval needed ?

Approve update from 2019-06-23 18:49:29

List of changes

Install ca-certificates (20190110-1.14)	Install libopenssl (1.0.2s-1.1)
Install kmod-crypto-wq (4.14.128-1-26...	Install kmod-crypto-rng (4.14.128-1-2...
Install kmod-crypto-iv (4.14.128-1-26...	Install kmod-crypto-seqiv (4.14.128-1-...
Install kmod-crypto-ctr (4.14.128-1-2...	Install kmod-crypto-pcbc (4.14.128-1-...
Install kmod-crypto-gf128 (4.14.128-1-...	Install kmod-crypto-xts (4.14.128-1-2...
Install kmod-crypto-cmac (4.14.128-1-...	Install kmod-crypto-sha512 (4.14.128-...
Install kmod-crypto-ccm (4.14.128-1-2...	Install kmod-crypto-deflate (4.14.128-...
Install fosquitto-monitor (18-0.0)	

Install now
Deny

Package lists

Extensions of network protocols for 3G/LTE
Support for additional protocols and connection types.

Print server
Services allowing to connect a printer to the router and use it for remote printing.

Tor
Service to increase anonymity on the Internet.

If you want to use other language than English you can select it from the following list:

CS
 DA
 DE
 FR
 LT
 PL
 RU
 SK
 HU
 IT
 NB

Discard changes
Save and update

Figure A.12: Original Foris. Screenshot of the updates settings page.

TURRIS

Turris OMNIA
foris version: 100.0

NOTIFICATIONS **3**

PASSWORD

REMOTE ACCESS

NETWORK INTERFACES

WAN

LAN

GUEST NETWORK

REGION AND TIME

DNS

WI-FI

MAINTENANCE

UPDATER

STORAGE

DIAGNOSTICS

ABOUT

ENGLISH ▾ | LOG OUT

About

Device	Turris Omnia
Serial number	47244721172
Turris OS version	4.0
Kernel version	4.14.123

Figure A.13: *Original Foris*. Screenshot of the “About” information page.

Current Foris class diagrams

⁹ The list of config handlers is not complete. Handlers of other modules have a similar structure.

B. CURRENT FORIS CLASS DIAGRAMS

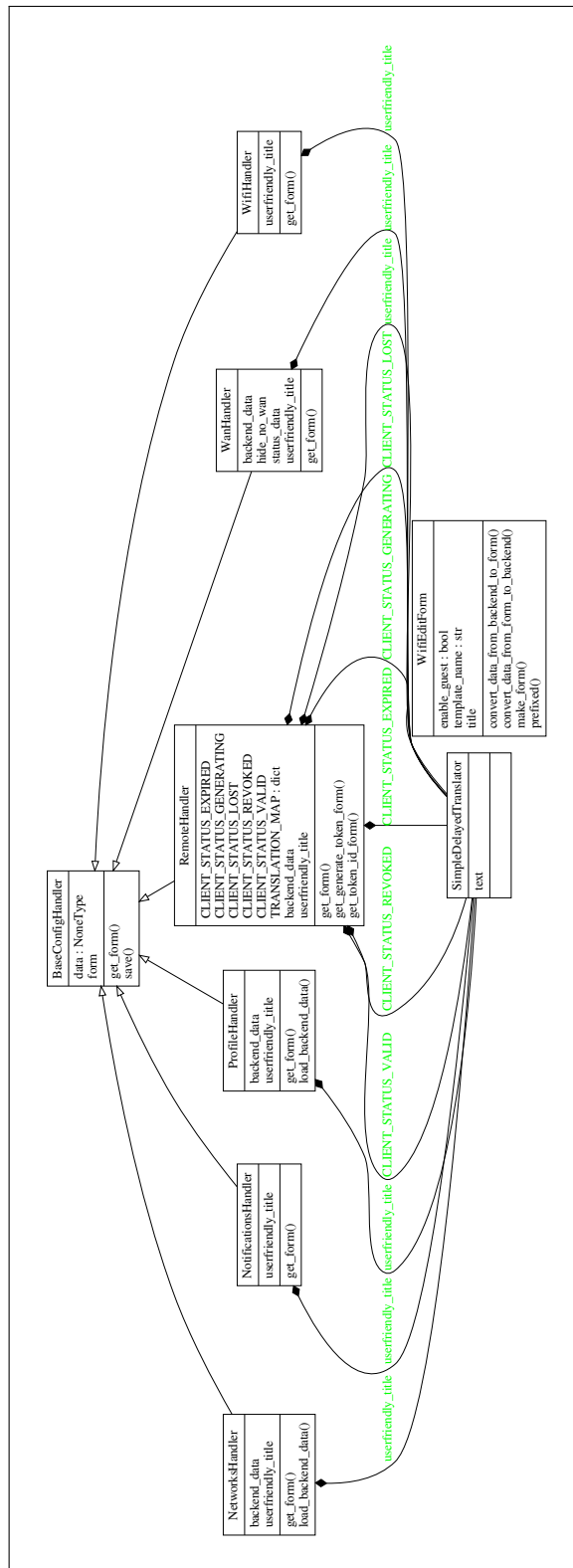


Figure B.1: *Config handler class diagram*⁹

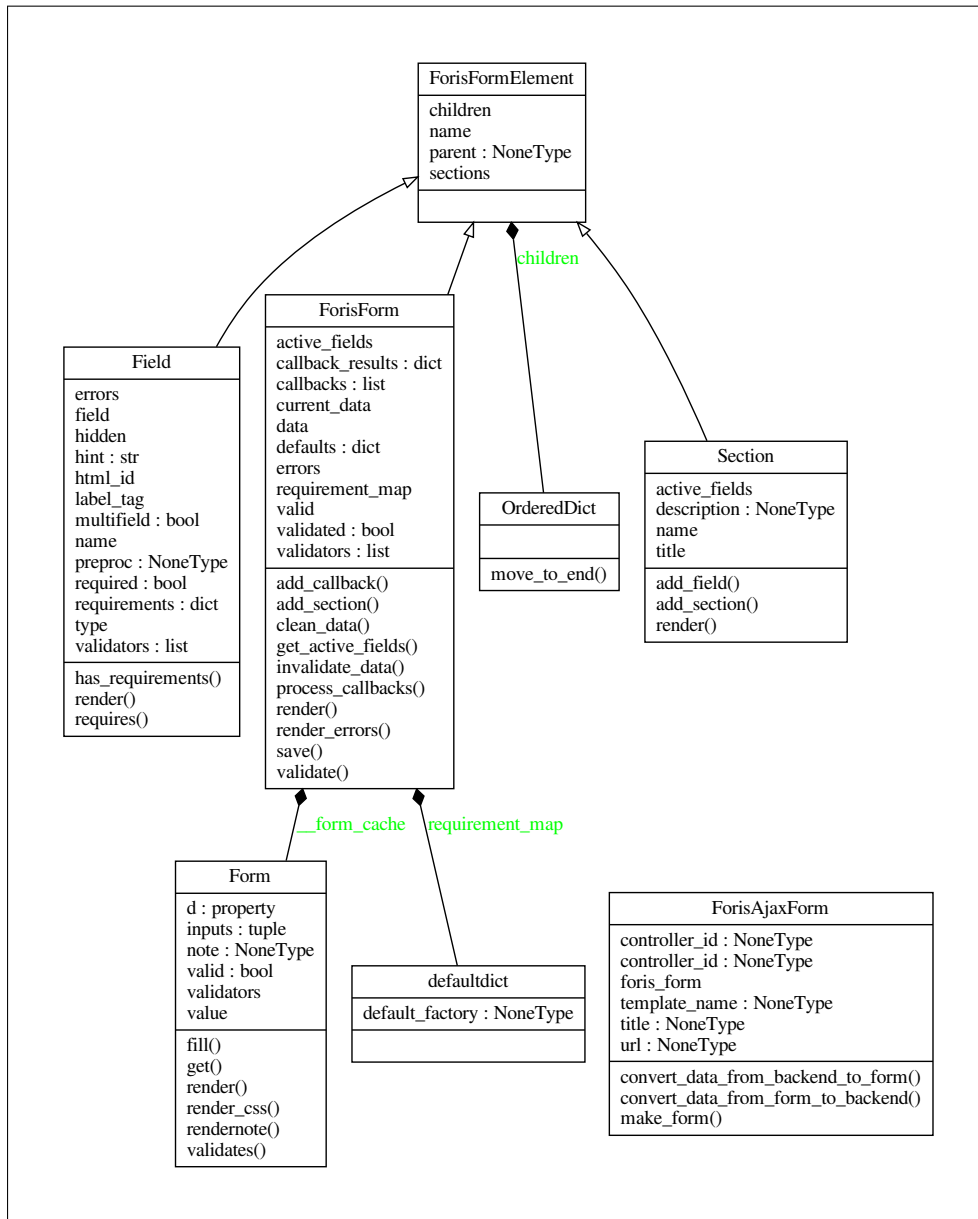


Figure B.2: *Foris forms class diagram.*

B. CURRENT FORIS CLASS DIAGRAMS

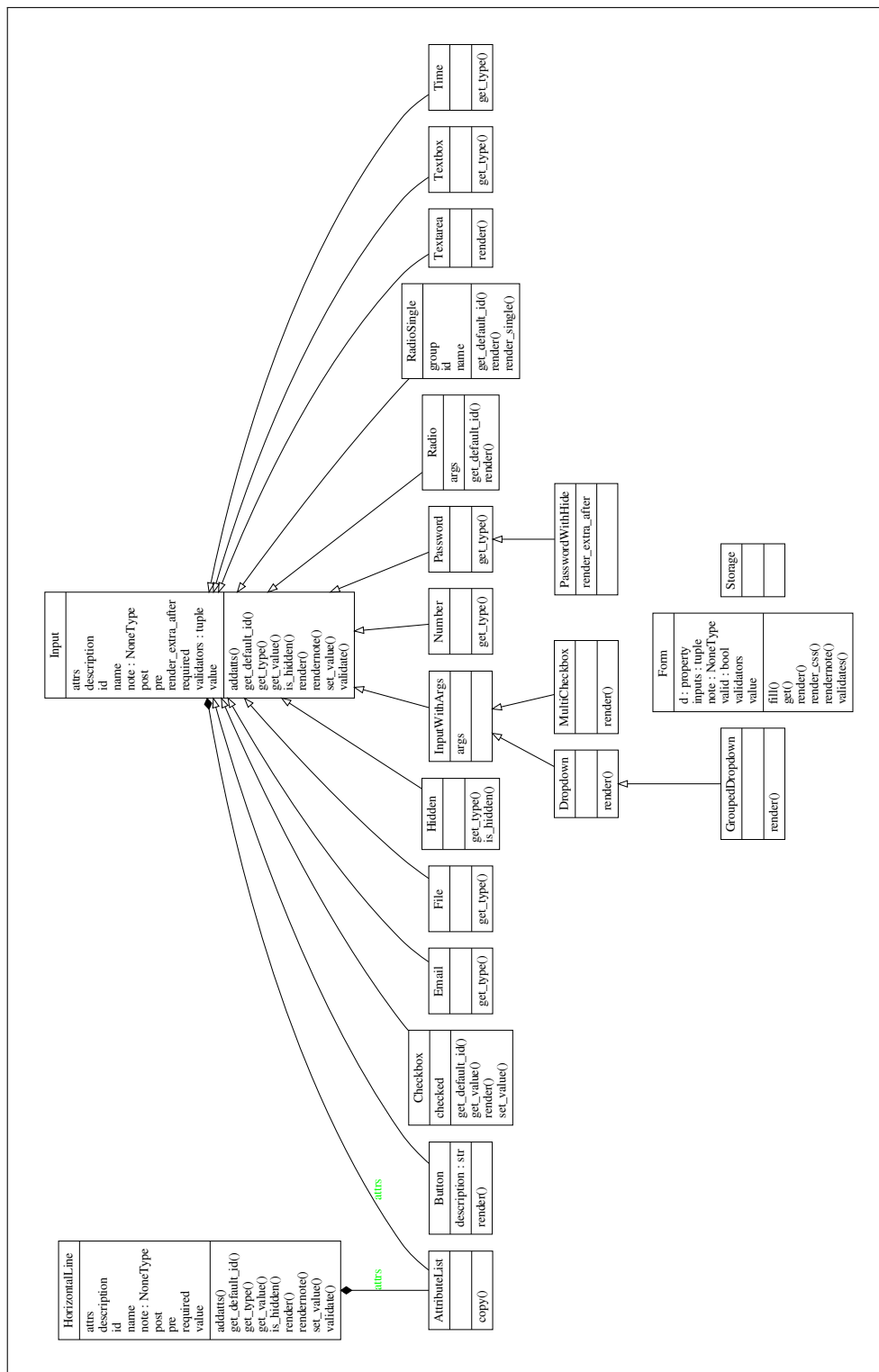


Figure B.3: Foris form fields class diagram.

Wireframes

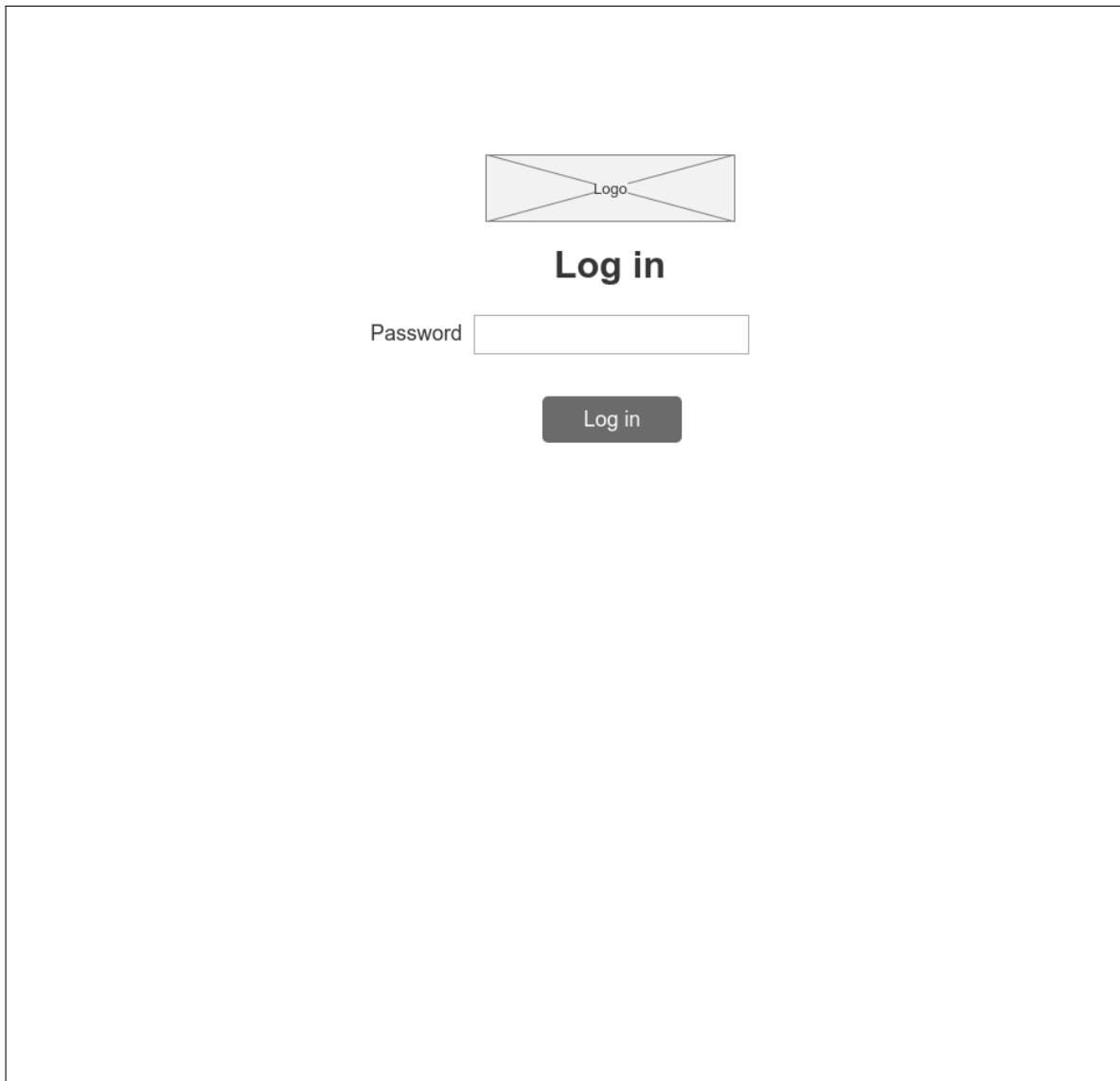


Figure C.1: *Wireframe of the login page.*

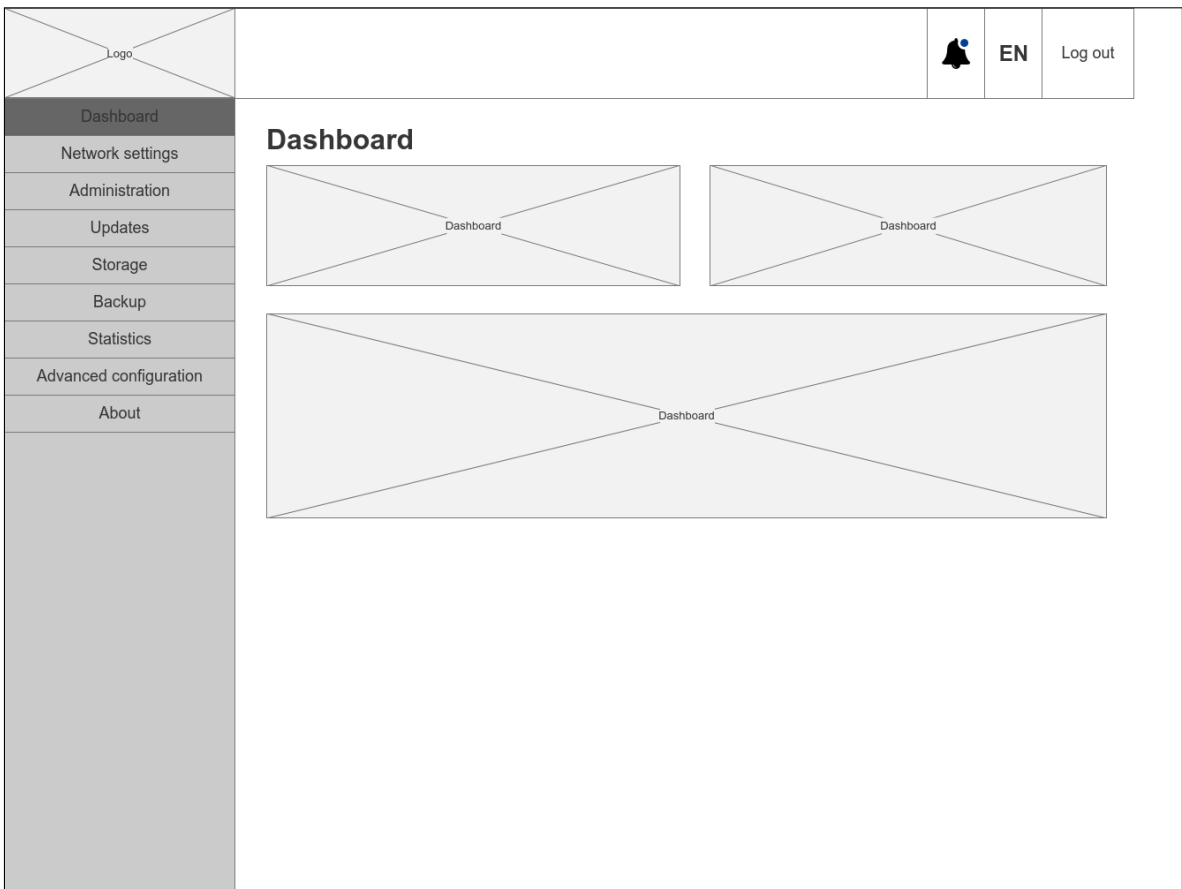


Figure C.2: Wireframe of the dashboard “Home” page.

C. WIREFRAMES

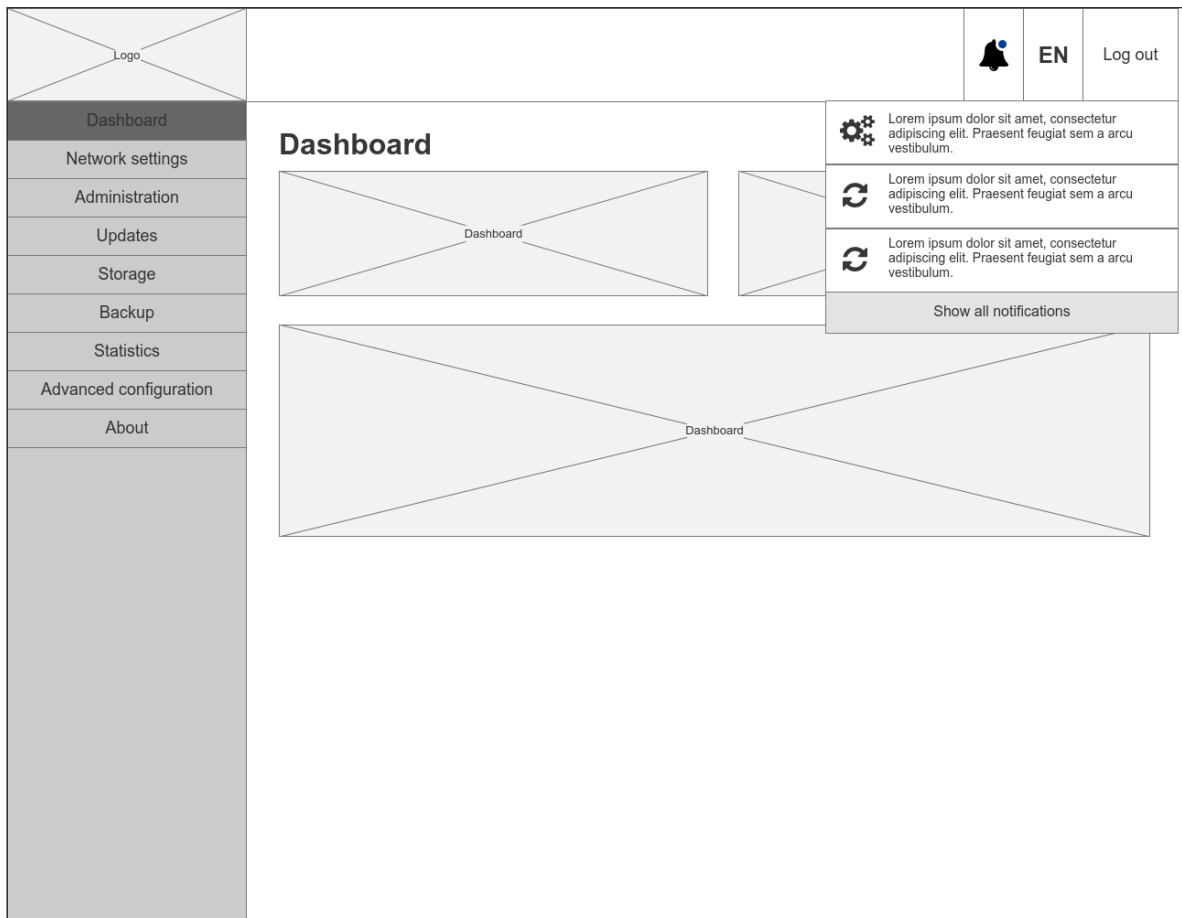


Figure C.3: *Wireframe of the notifications dropdown menu.*

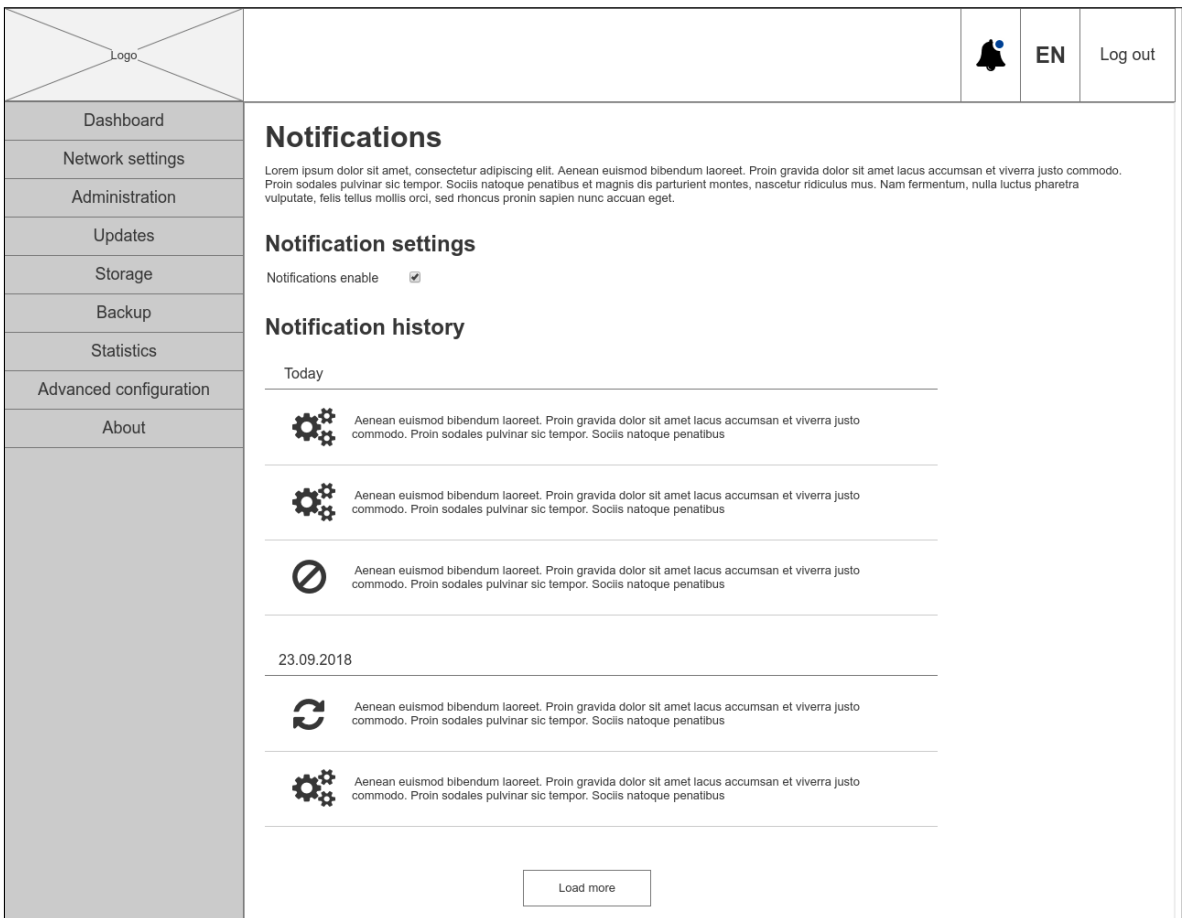


Figure C.4: *Wireframe of the notifications page.*

C. WIREFRAMES


Logo	 EN Log out
Dashboard	<h3>WAN</h3> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean euismod bibendum laoreet. Proin gravida dolor sit amet lacus accumsan et viverra justo commodo. Proin sodales pulvinar sic tempor. Sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Nam fermentum, nulla luctus pharetra vulputate, felis tellus mollis orci, sed rhoncus proin sapien nunc accuan eget.</p> <p>IPv4 protocol <input type="text" value="DHCP Automatic configuration"/></p> <p>DHCP hostmane <input type="text"/></p> <p>Custom MAC address <input type="checkbox"/></p> <p><input type="button" value="Discard"/> <input type="button" value="Save"/></p> <hr/> <h3>Connection test</h3> <p>Here you can test you connection settings. Remember to click on the Save button before running the test. Note that sometimes it takes a while before the connection is fully initialized. So it might be useful to wait for a while before running this test.</p> <p><input type="button" value="Test Connection"/></p>
Network settings	
WiFi	
WAN	
LAN	
Administration	
Updates	
Storage	
Backup	
Statistics	
Advanced configuration	
About	

Figure C.5: Wireframe of the WAN configuration page.

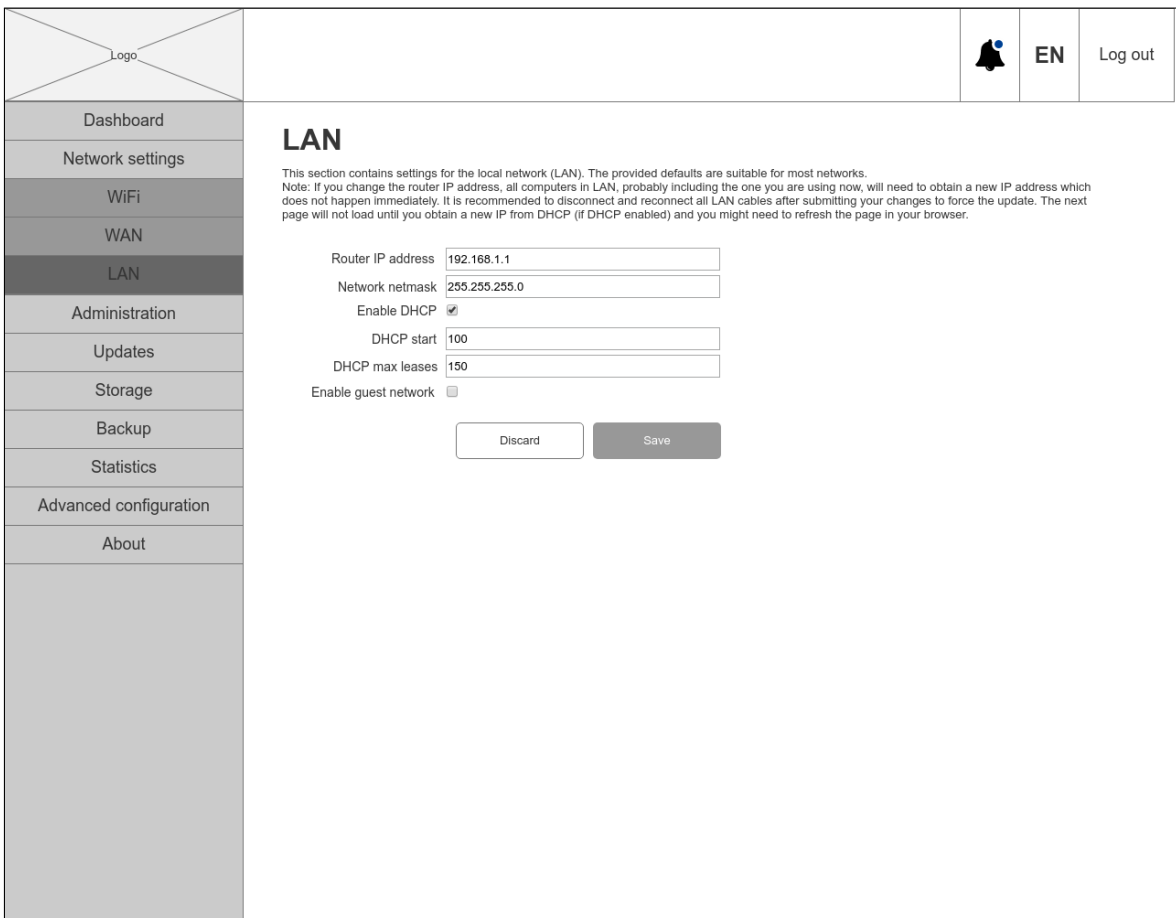


Figure C.6: Wireframe of the LAN configuration page.

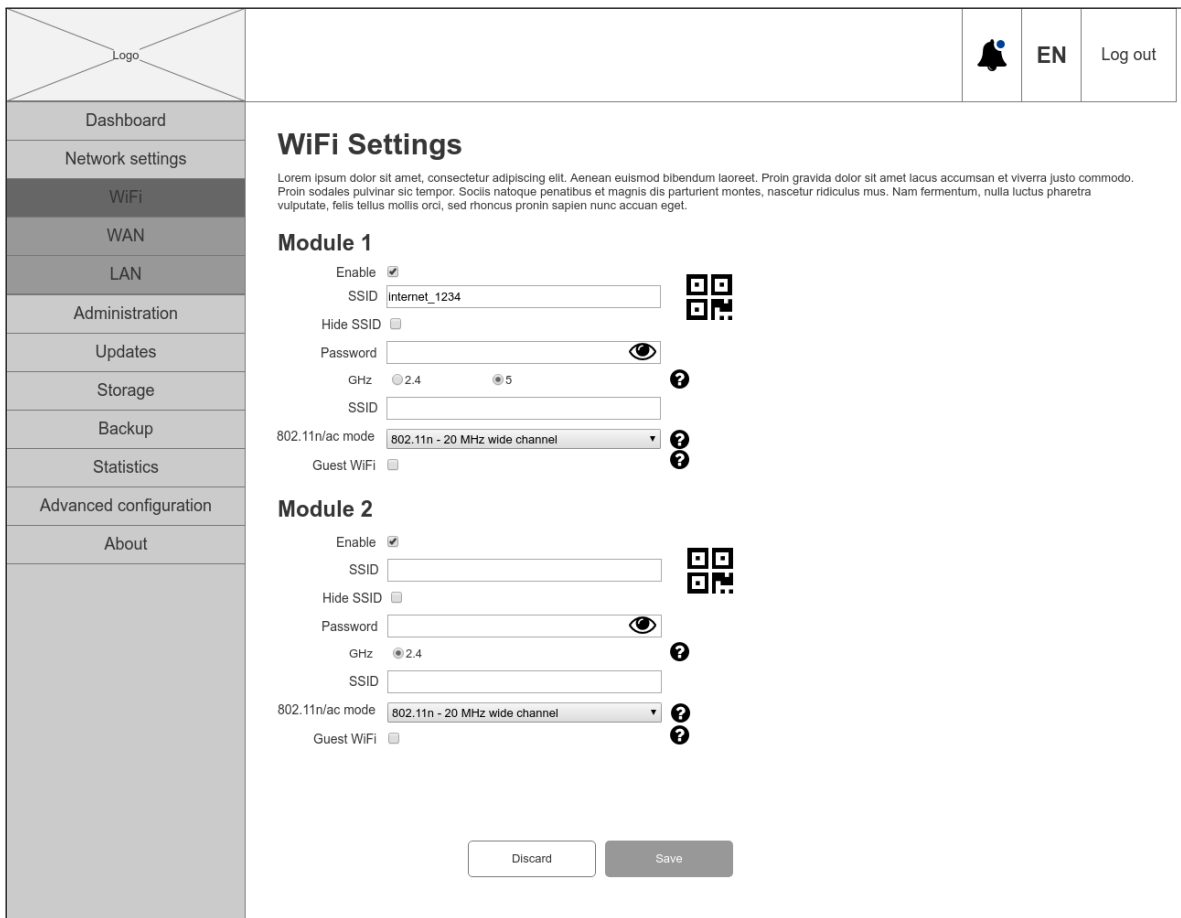


Figure C.7: Wireframe of the Wi-Fi configuration page.

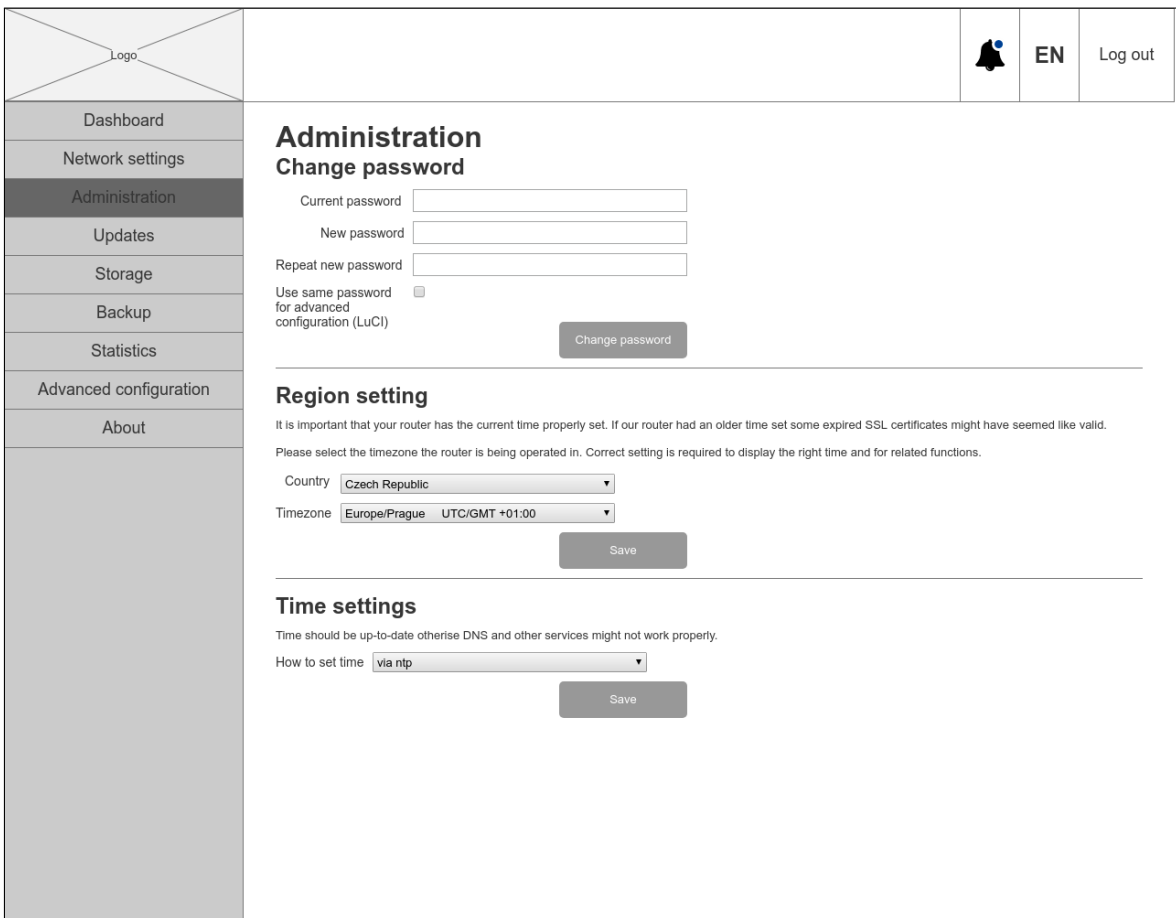


Figure C.8: Wireframe of the administration page.

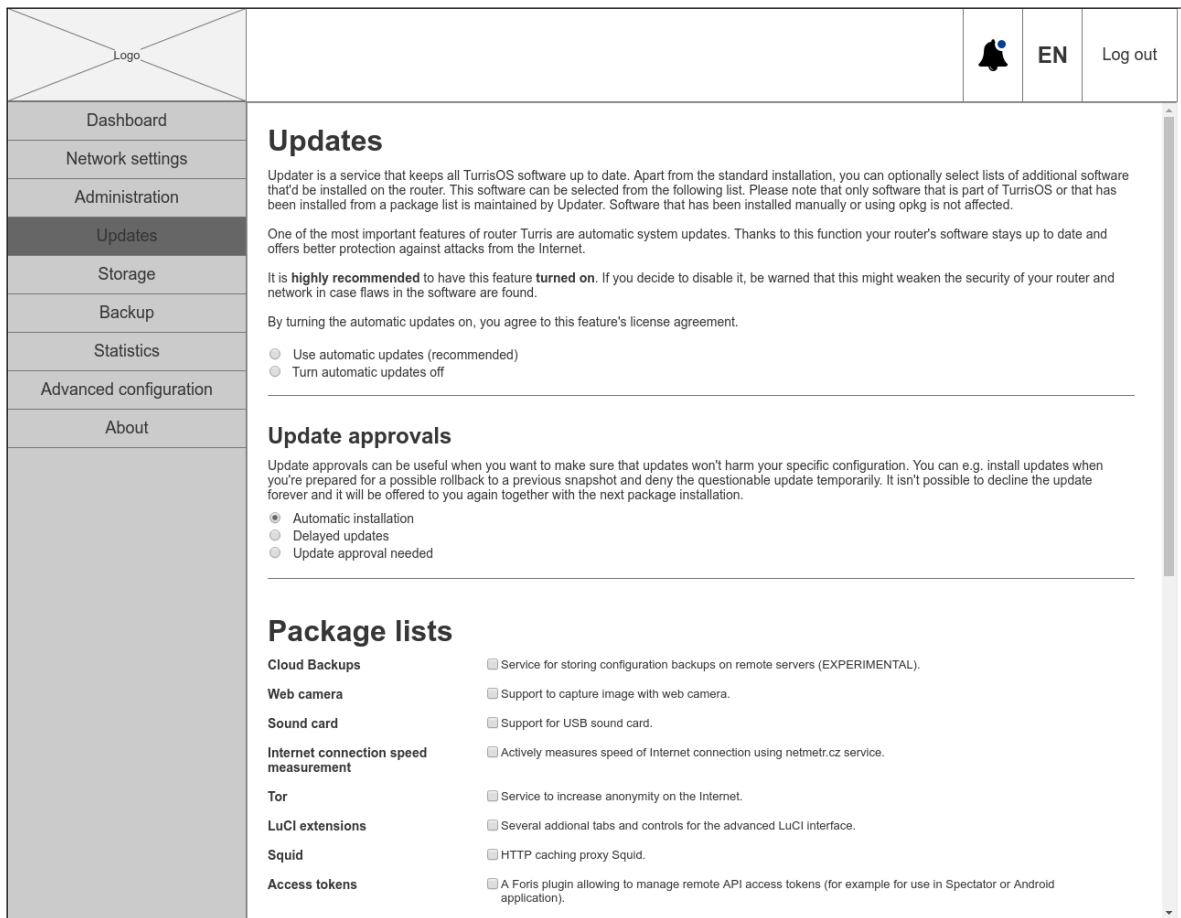


Figure C.9: Wireframe of the updates settings page.

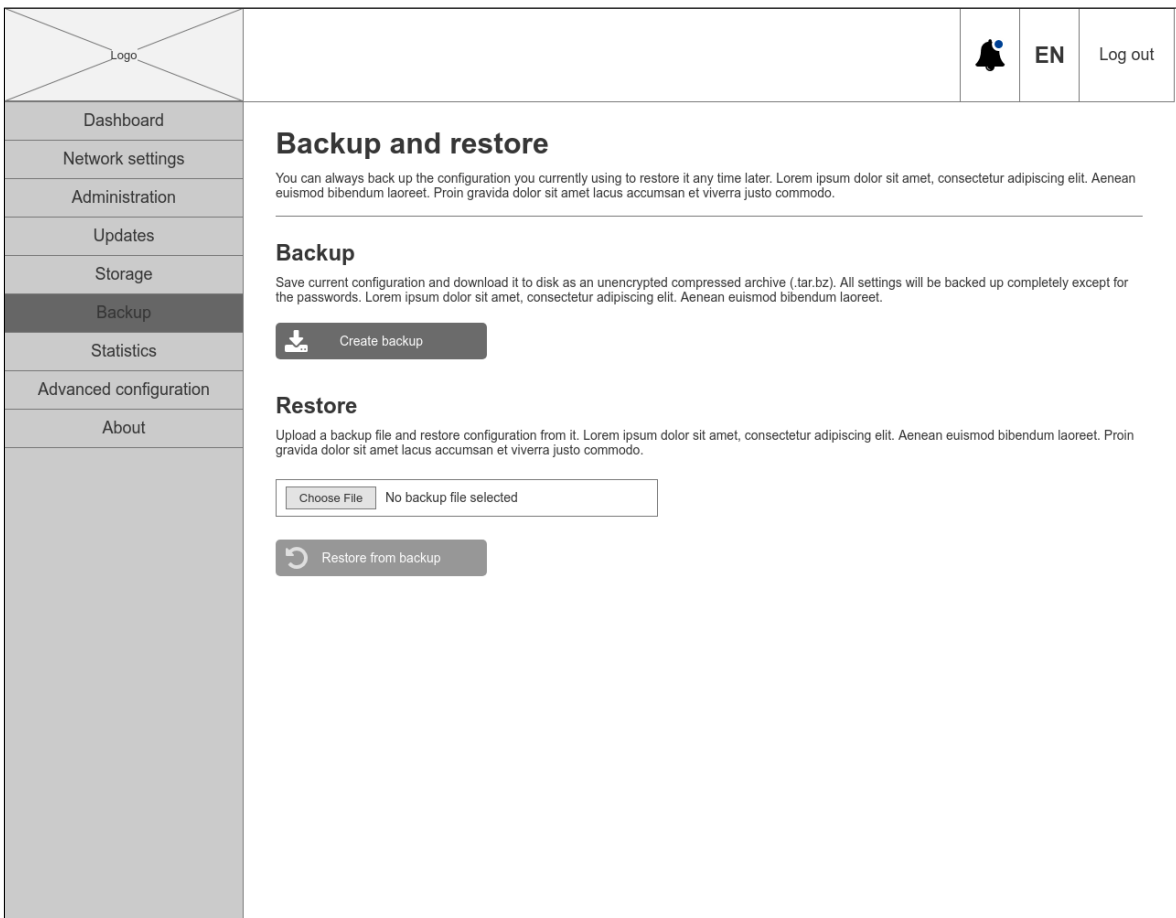


Figure C.10: *Wireframe of the backups creation and restoration page.*

C. WIREFRAMES

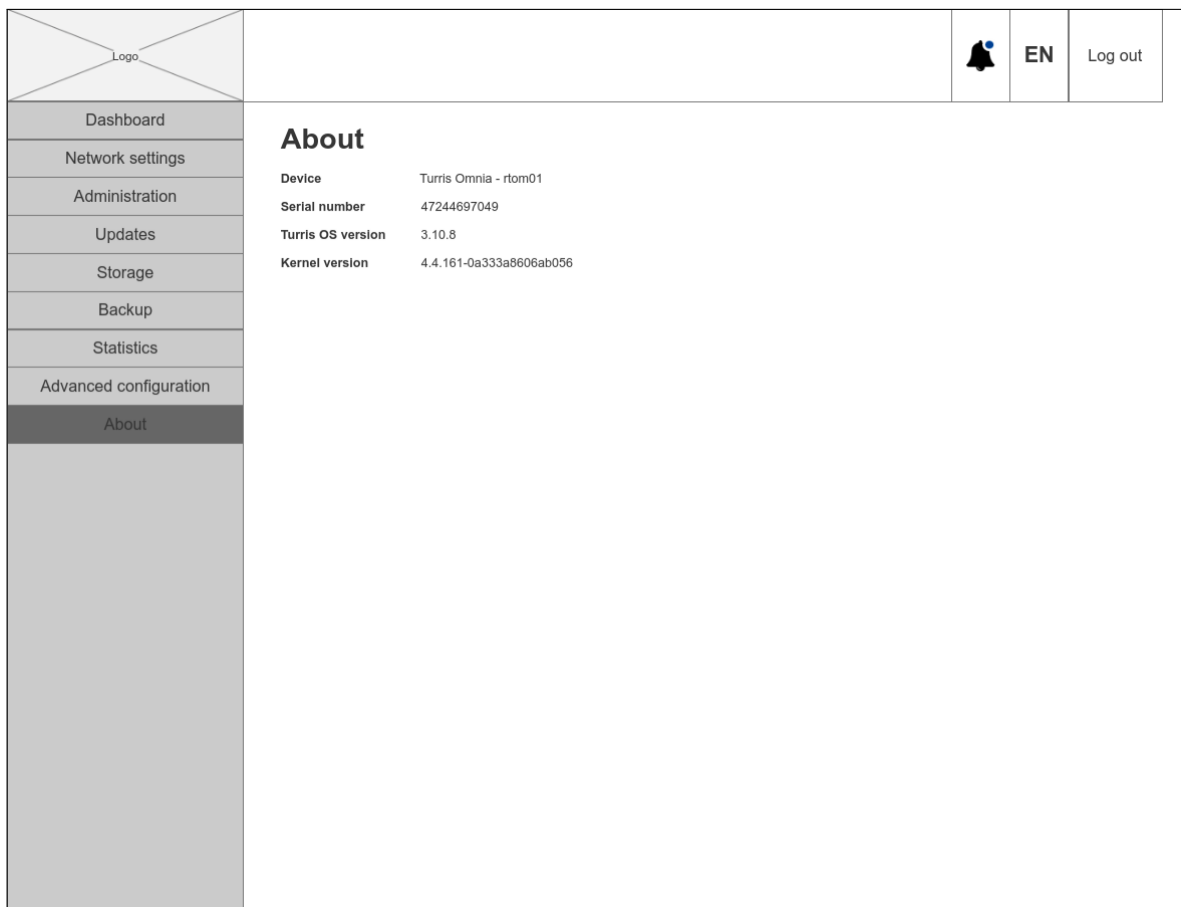


Figure C.11: Wireframe of the “About” information page.

Hi-fi prototype screenshots

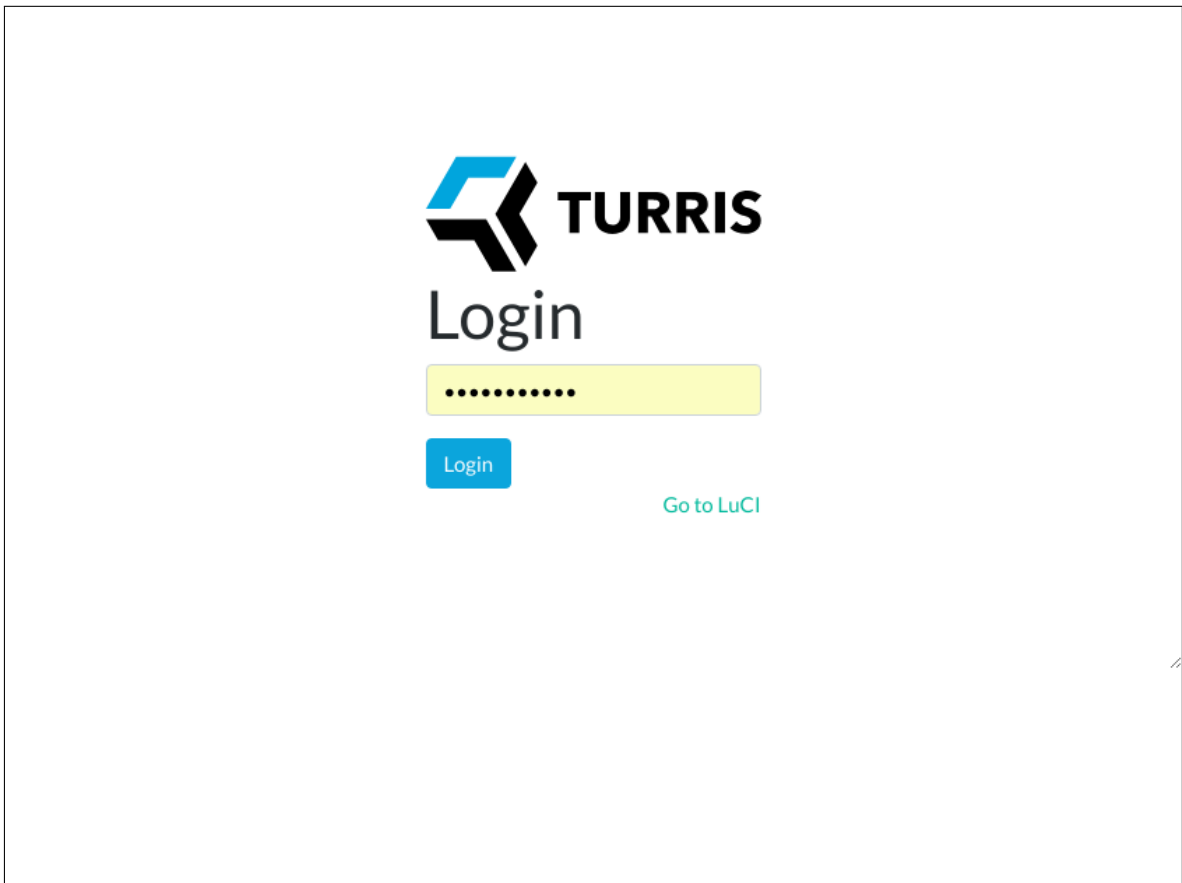


Figure D.1: *Hi-fi prototype. Screenshot of the login page.*

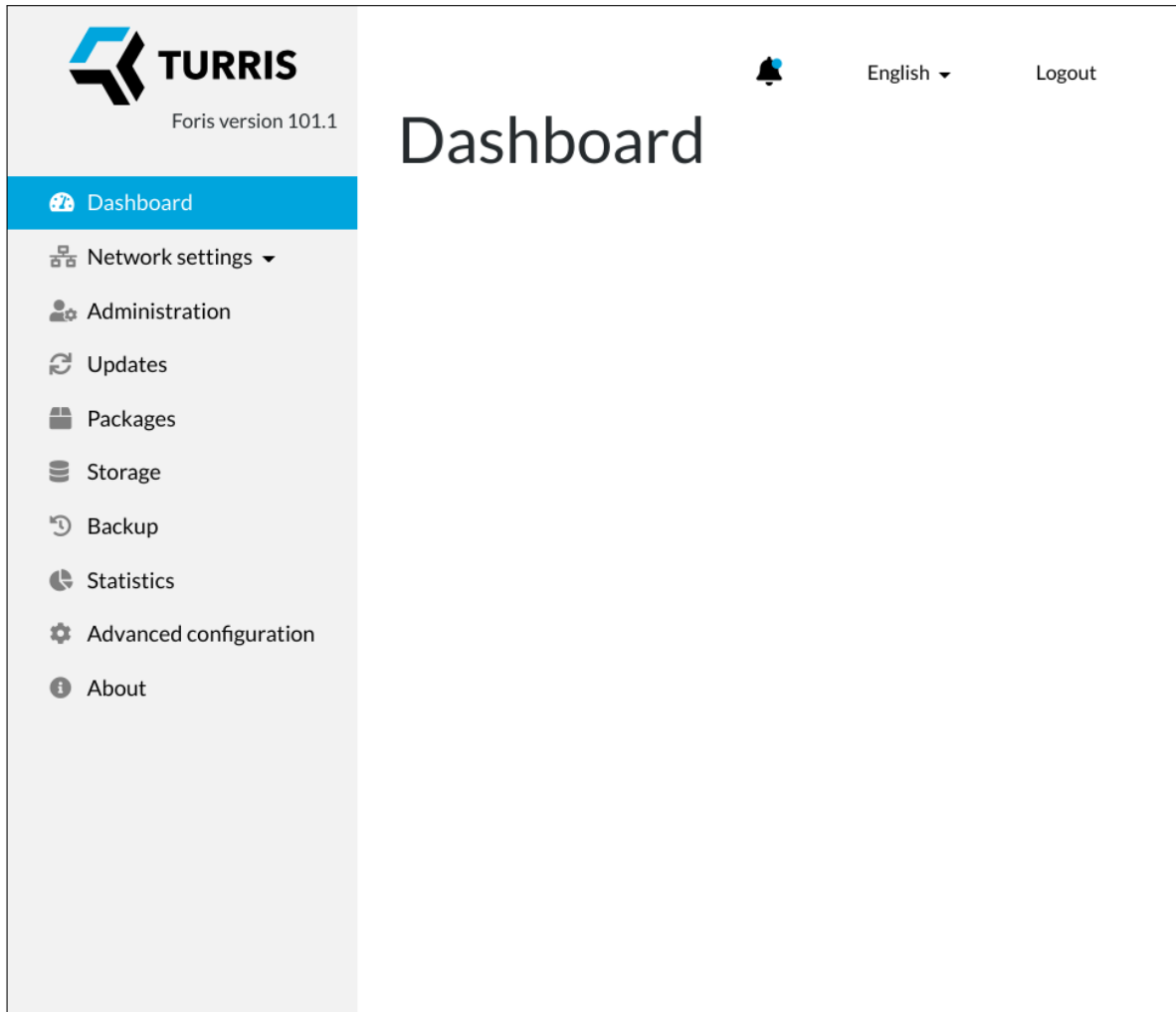


Figure D.2: *Hi-fi prototype. Screenshot of the dashboard (home) page.*

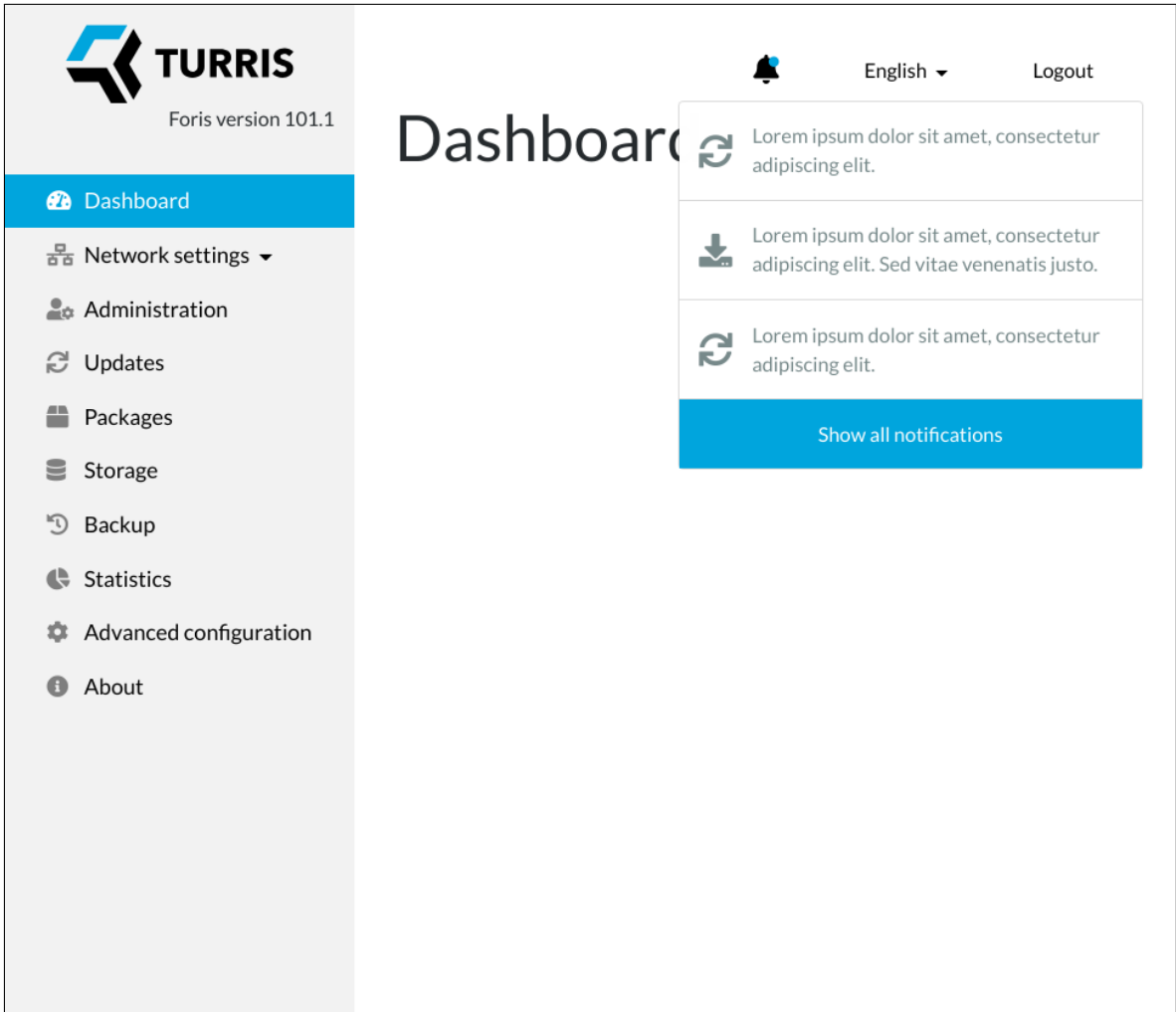


Figure D.3: *Hi-fi prototype. Screenshot of the notifications dropdown menu.*

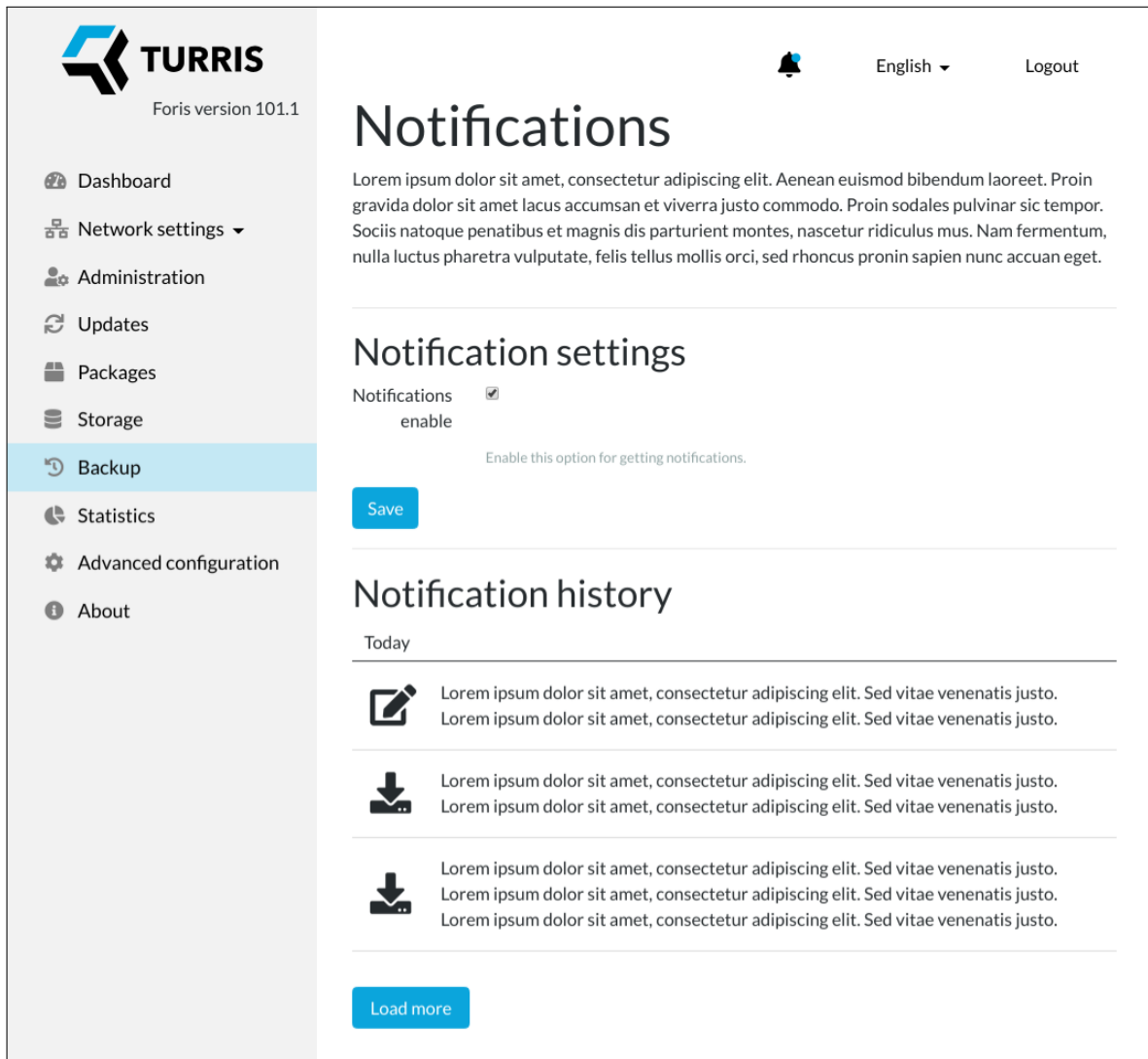


Figure D.4: *Hi-fi prototype. Screenshot of the notifications page.*

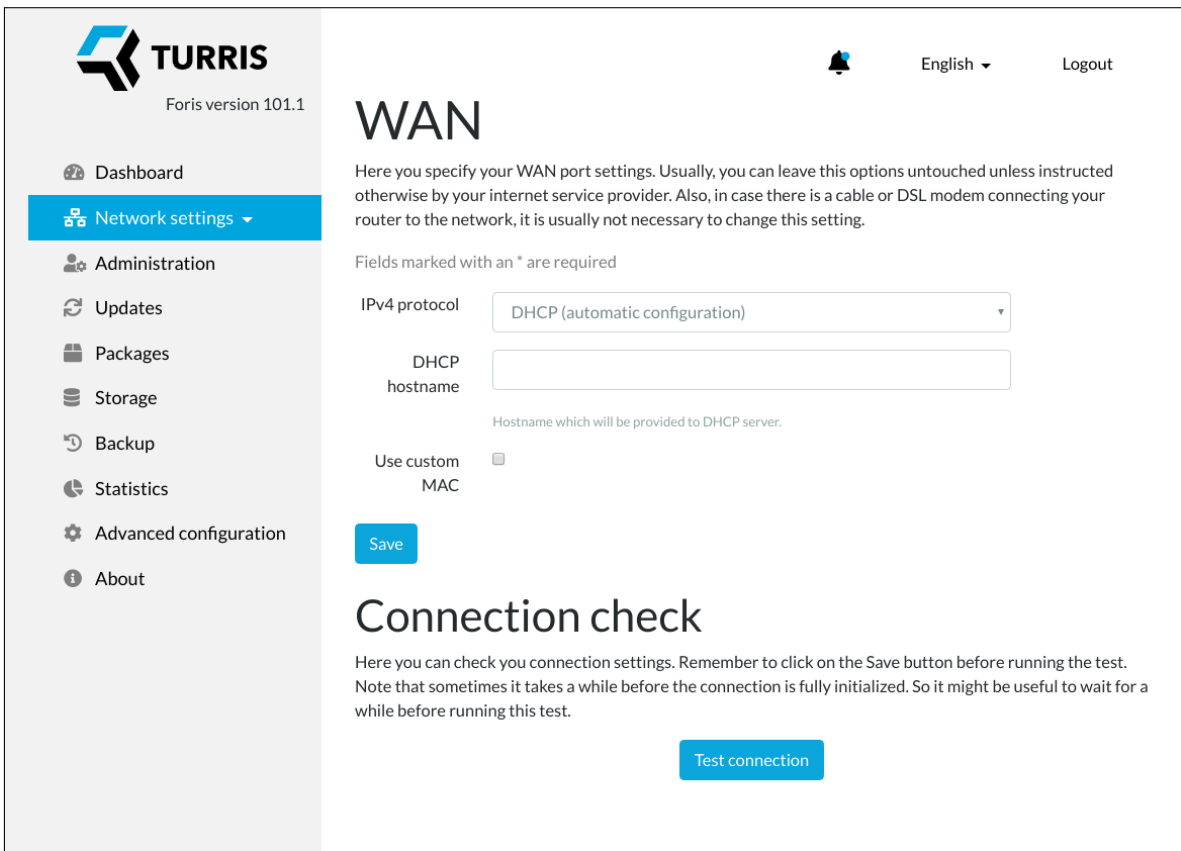


Figure D.5: *Hi-fi prototype. Screenshot of the WAN configuration page.*

D. HI-FI PROTOTYPE SCREENSHOTS

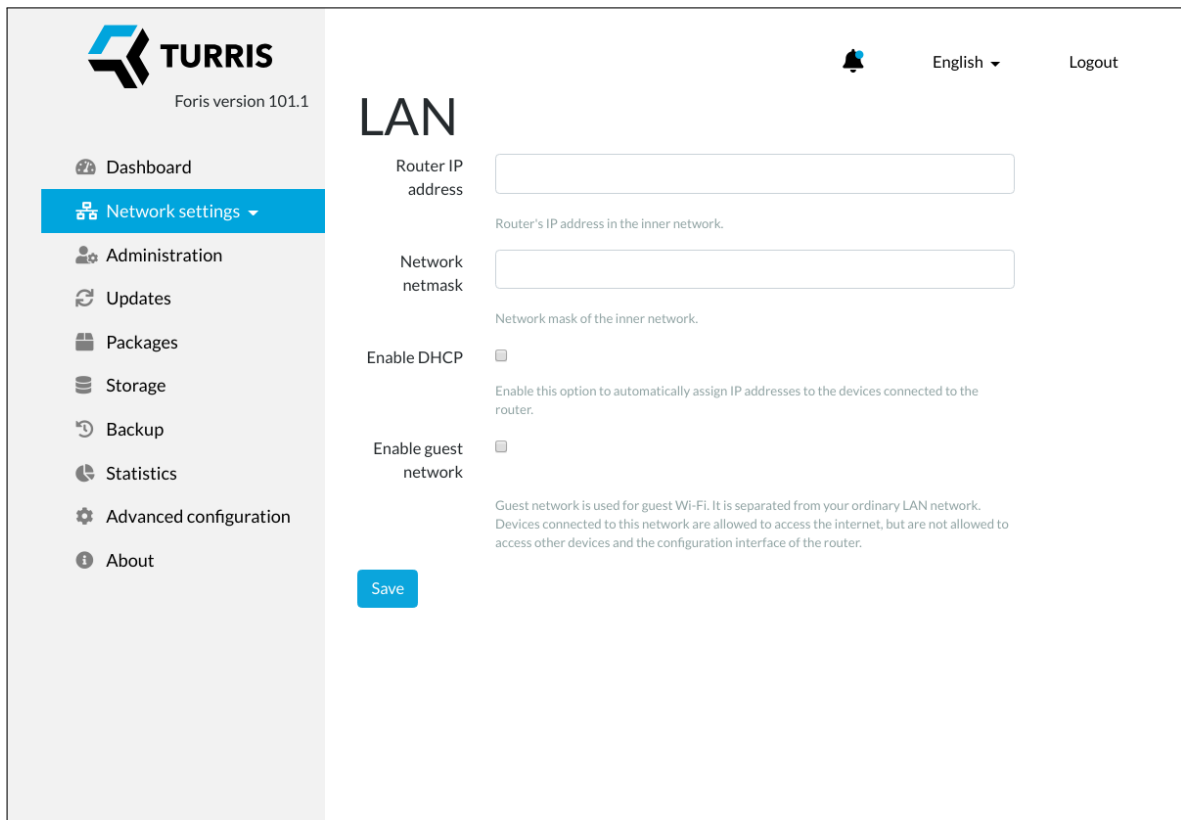


Figure D.6: *Hi-fi prototype. Screenshot of the LAN configuration page.*

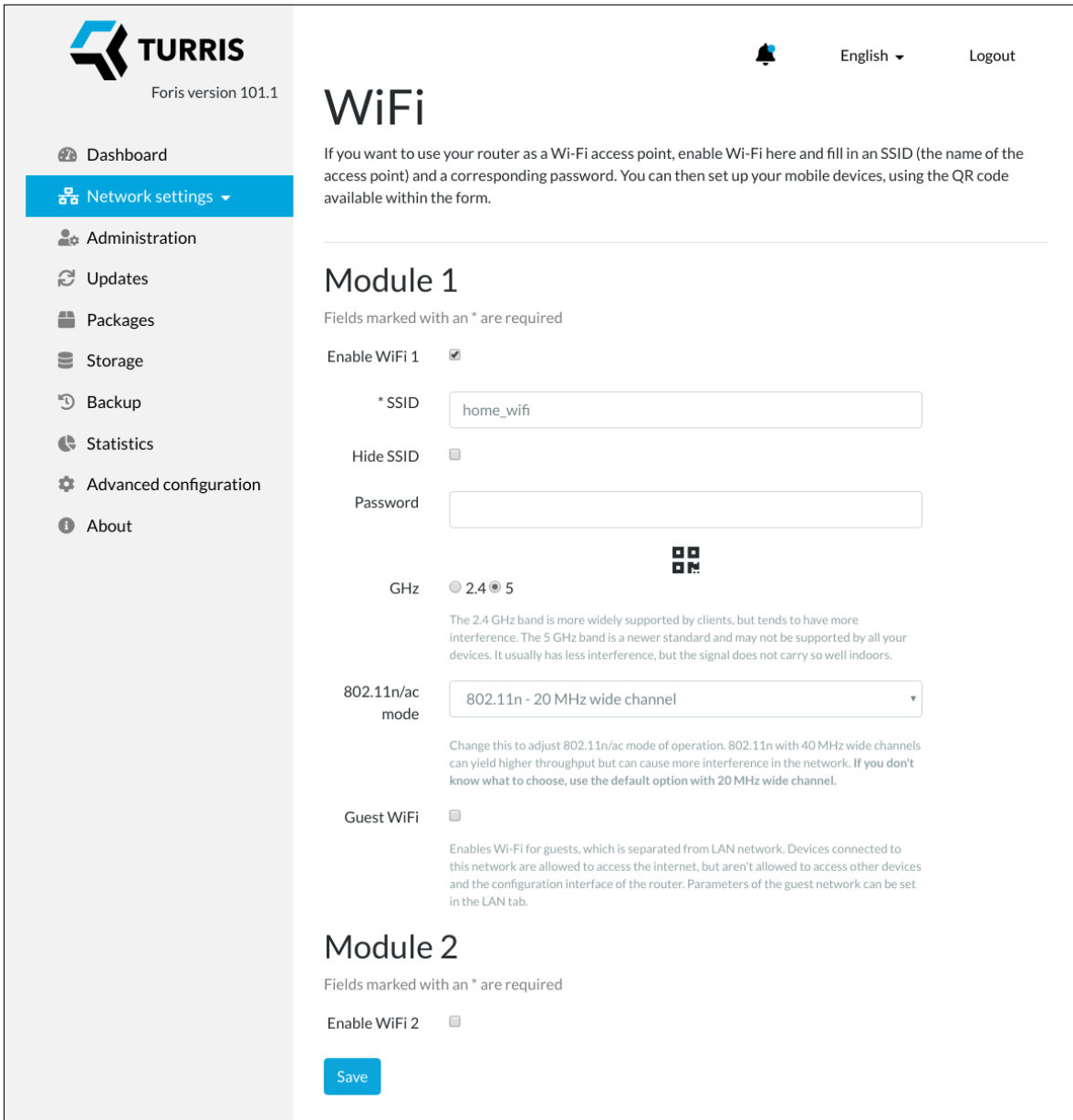


Figure D.7: *Hi-fi prototype. Screenshot of the Wi-Fi configuration page.*

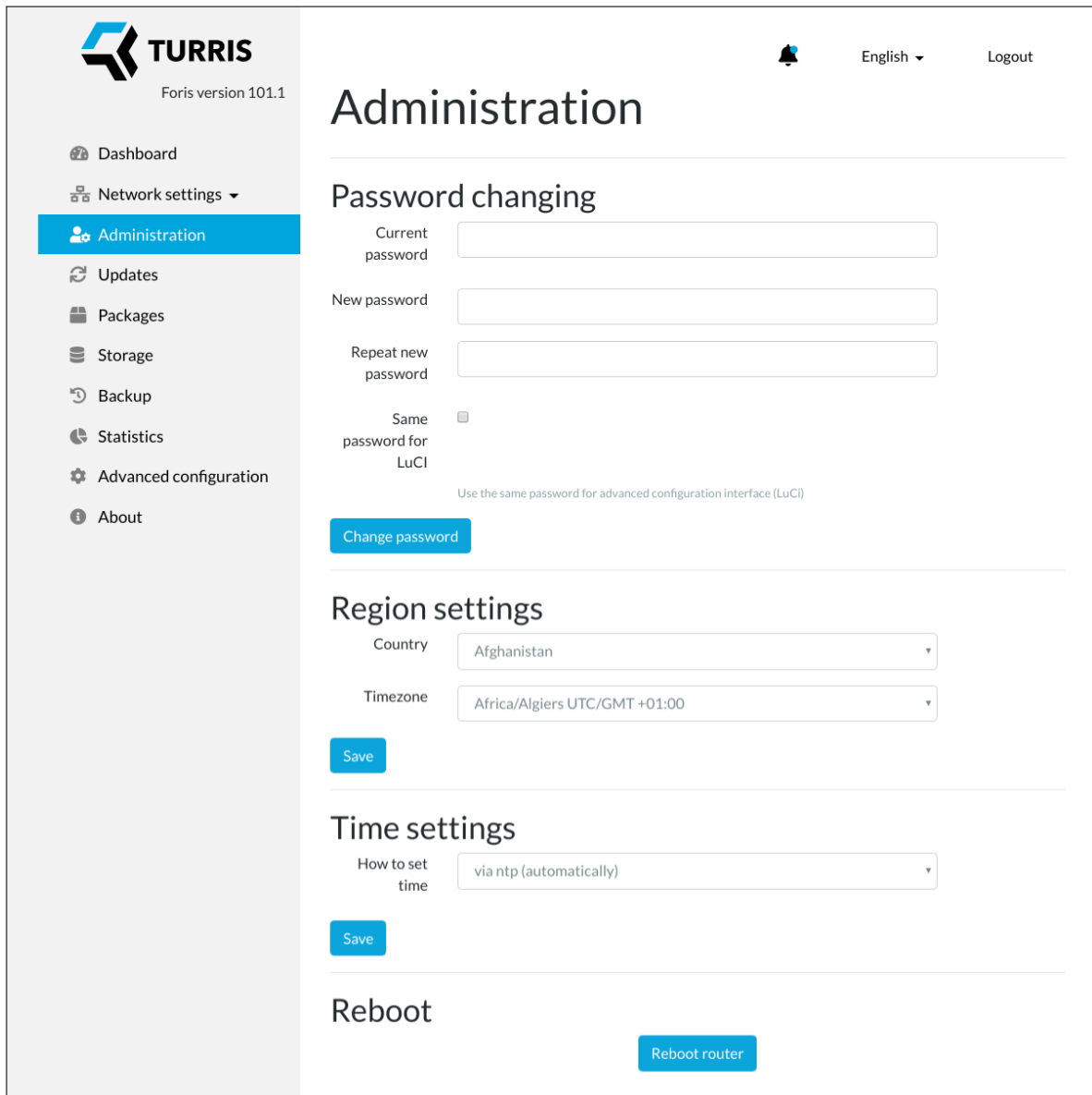


Figure D.8: *Hi-fi prototype. Screenshot of the administration page.*

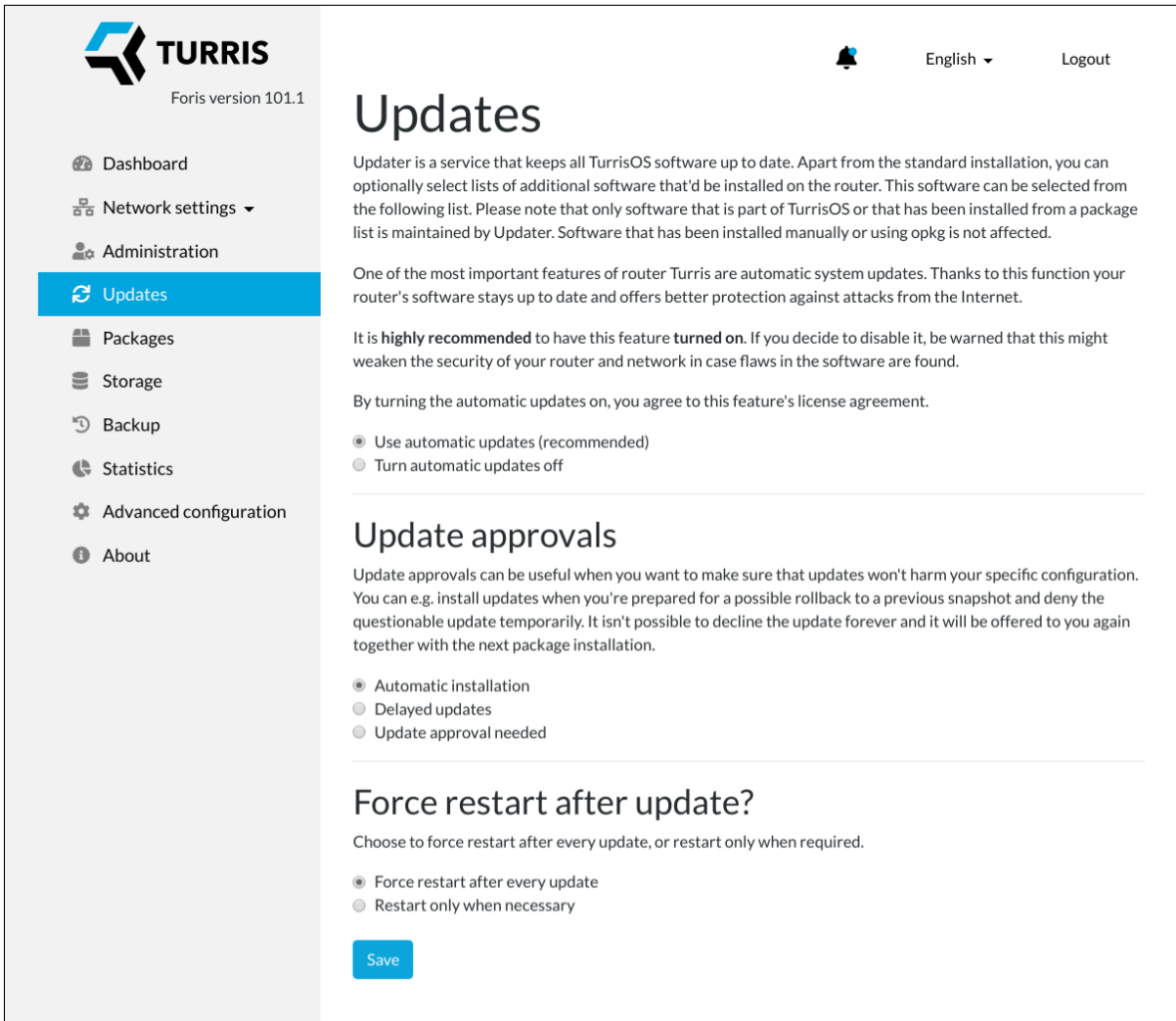


Figure D.9: *Hi-fi prototype. Screenshot of the updates settings page.*

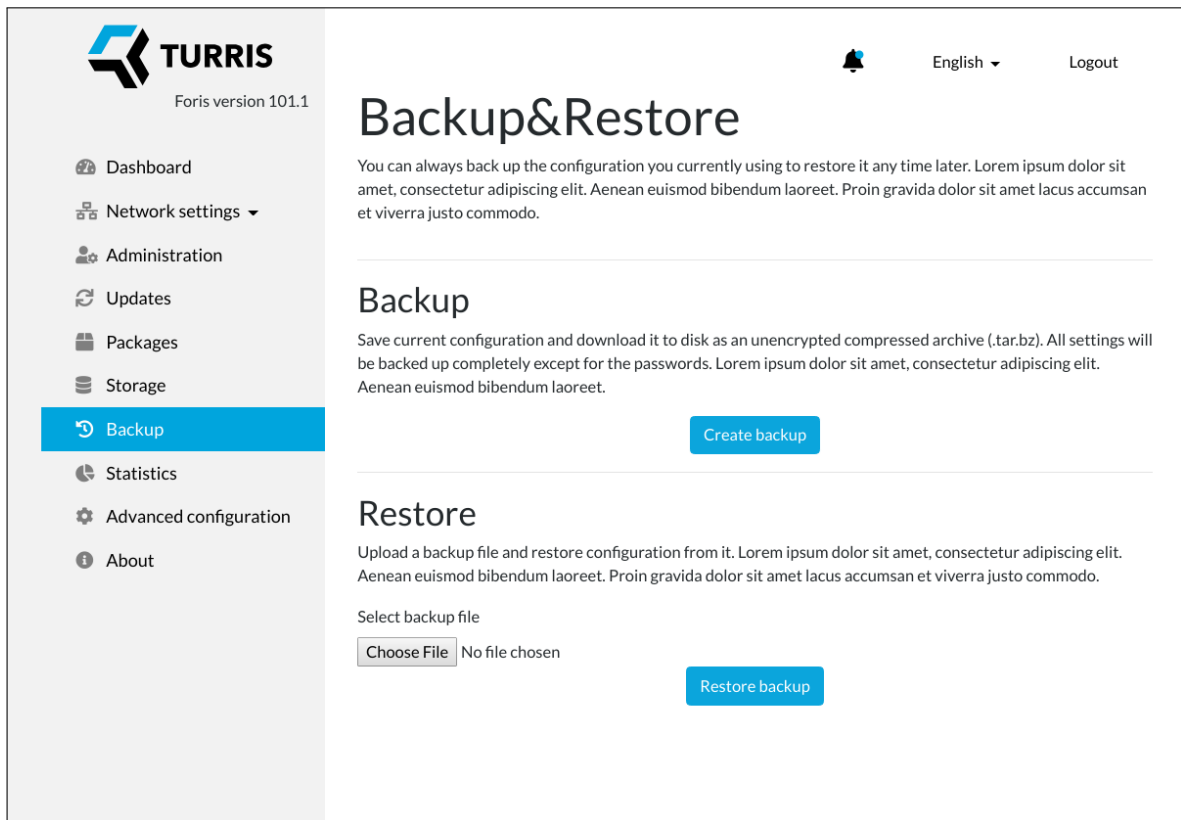


Figure D.10: *Hi-fi prototype. Screenshot of the backups creation and restoration page.*

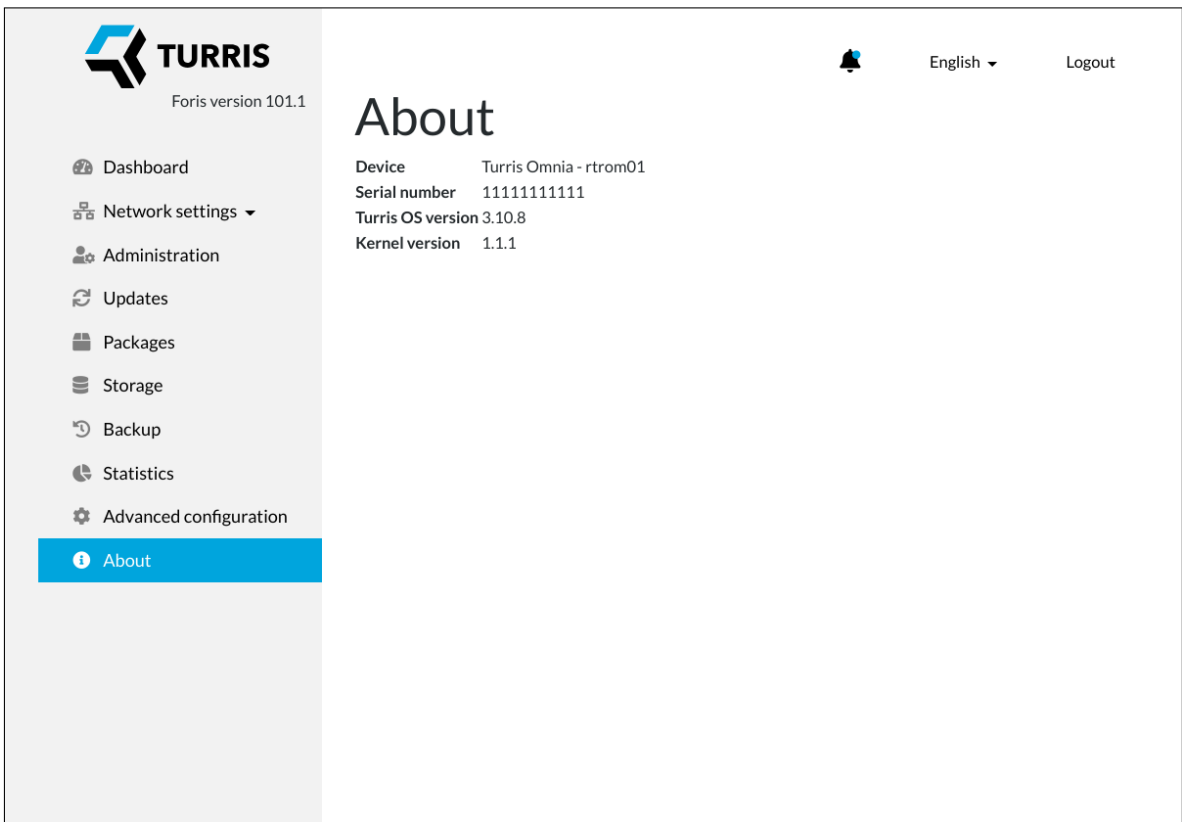


Figure D.11: *Hi-fi prototype*. Screenshot of the “About” information page.

API endpoints

- Notifications

Description: Provide list of the router notifications.

URL: /notifications

HTTP Method: GET

foris-controller module: router_notifications

foris-controller action: list

- Notification setting

Description: Get and update notifications settings.

URL: /notifications-settings

HTTP Methods: GET, POST

foris-controller module: router_notifications

foris-controller actions: get_settings, update_settings

- Network settings

- WAN

Description: Get and update WAN settings.

URL: /wan

HTTP Methods: GET, POST

foris-controller module: wan

foris-controller actions: get_settings, update_settings

foris-controller data: ["ipv4", "ipv6"]

- DNS settings test

Description: Trigger DNS settings test. The results of the test are obtained via WebSockets and can be displayed in real-time.

URL: /connection-test

HTTP Method: GET

foris-controller module: wan

foris-controller actions: connection_test_trigger

foris-controller request data: ["dns"]

Note: The `foris-controller` module and action are the same as in WAN connection test but have different request data. `Foris-controller` uses the same module and action for WAN and DNS tests for a historical reason.

– Connection test

Description: Trigger WAN connection test. The results of the test are obtained via WebSockets and can be displayed in real-time.

URL: /connection-test

HTTP Method: GET

foris-controller module: wan

foris-controller actions: connection_test_trigger

foris-controller request data: ["wan"]

– LAN

Description: Get and update LAN settings.

URL: /lan

HTTP Methods: GET, POST

foris-controller module: lan

foris-controller actions: get_settings, update_settings

– Wi-Fi

Description: Get and update Wi-Fi settings.

URL: /lan

HTTP Methods: GET, POST

foris-controller module: wifi

foris-controller actions: get_settings, update_settings

– DNS

Description: Get and update DNS settings.

URL: /lan

HTTP Methods: GET, POST

foris-controller module: dns

foris-controller actions: get_settings, update_settings

– Interfaces

Description: Get and update interface settings.

URL: /interfaces

HTTP Methods: GET, POST

foris-controller module: networks

foris-controller actions: get_settings, update_settings

– Guest network

Description: Get and update guest network settings.

URL: /guest-network

HTTP Methods: GET, POST

foris-controller module: guest

foris-controller actions: get_settings, update_settings

- Updates

Description: Get and update updater settings.

URL: /updates

HTTP Methods: GET, POST

foris-controller module: updater

foris-controller actions: get_settings, update_settings

- Updates approvals

Description: Get updates approval information or approve updates.

URL: /approvals

HTTP Methods: GET, POST

foris-controller module: updater

foris-controller actions: get_settings, resolve_approval

- Packages

Description: Get and update packages settings.

URL: /packages

HTTP Methods: GET, POST

foris-controller module: updater

foris-controller actions: get_settings, update_settings

- Password

Description: Set Foris and root password.

URL: /password

HTTP Method: POST

foris-controller module: password

foris-controller action: set

- Region and time settings

Description: Get and update region (timezone) and time settings.

URL: /region-and-time

HTTP Methods: GET, POST

foris-controller module: time

foris-controller actions: get_settings, update_settings

- Current router time

Description: Get router current time.

URL: /time

HTTP Method: GET

foris-controller module: time

foris-controller action: get_router_time

- Region and time settings

Description: Get and update region (timezone) and time settings.

URL: /region-and-time

HTTP Methods: GET, POST

foris-controller module: time

foris-controller actions: get_settings, update_settings

- Reboot

Description: Trigger reboot of the router. The state of the router after the reboot was triggered is checked via WebSockets and health check endpoint.

URL: /reboot

HTTP Method: GET

foris-controller module: maintain

foris-controller action: reboot

- Languages

Description: Get list of installed interface languages.

URL: /languages

HTTP Method: GET

- Language

Description: Get or set the current language.

URL: /language

HTTP Methods: GET, POST

foris-controller module: web

foris-controller actions: get_data, set_language

- Guide

- Guide

Description: Get guide state.

URL: /guide

HTTP Methods: GET

foris-controller module: web

foris-controller actions: get_data, get_guide

- Guide workflow

Description: Set guide workflow.

URL: /guide-workflow

HTTP Methods: POST

foris-controller module: web

foris-controller actions: update_guide

- Finish (skip) guide

Description: Set guide state as finished.

URL: /finish-guide

HTTP Methods: POST

foris-controller module: web

foris-controller actions: update_guide

- Health check

Description: Check if router configuration interface is up. This endpoint is used during device reboot and network restart to check whether the web server is up or down.

URL: /health-check

HTTP Method: GET

reForis web interface screenshots

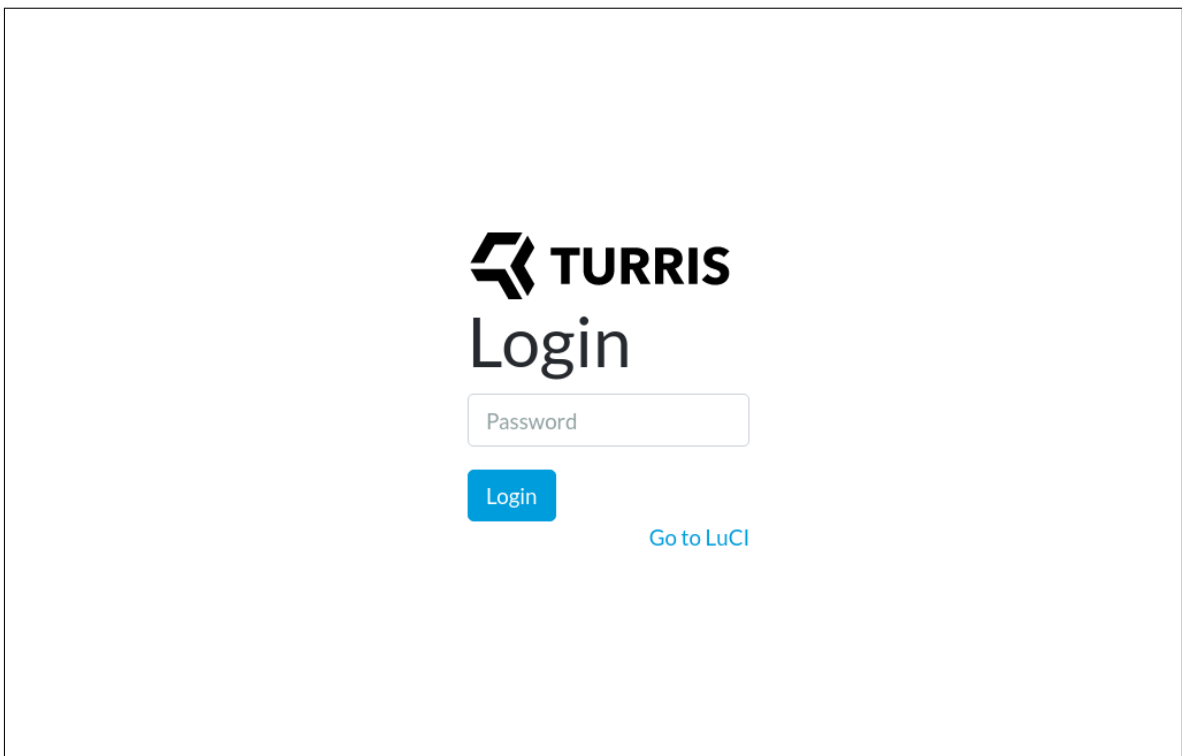


Figure F.1: *reForis*. Screenshot of the login page.

TURRIS
reForis version 0.2

en Logout

Overview

Approve update from 2019-06-23 18:49:29

Action	Name	Current	New
install	ca-certificates		20190110-1.14
install	libopenssl		1.0.2s-1.1
install	libcurl		7.60.0-4.1
install	updater-ng		63.1-1.2
install	python3-base		3.6.8-3.6-5.8
install	kmod-crypto- xts		4.14.128-1- 26d72286c6cd7547488323b0f0c7a362.2
install	kmod-crypto- cmac		4.14.128-1- 26d72286c6cd7547488323b0f0c7a362.2
install	kmod-crypto- sha512		4.14.128-1- 26d72286c6cd7547488323b0f0c7a362.2
install	kmod-crypto- ccm		4.14.128-1- 26d72286c6cd7547488323b0f0c7a362.2
install	kmod-crypto- deflate		4.14.128-1- 26d72286c6cd7547488323b0f0c7a362.0
install	fosquitto- monitor		18-0.0

Deny Install now

Figure F.2: reForis. Screenshot of the overview page.

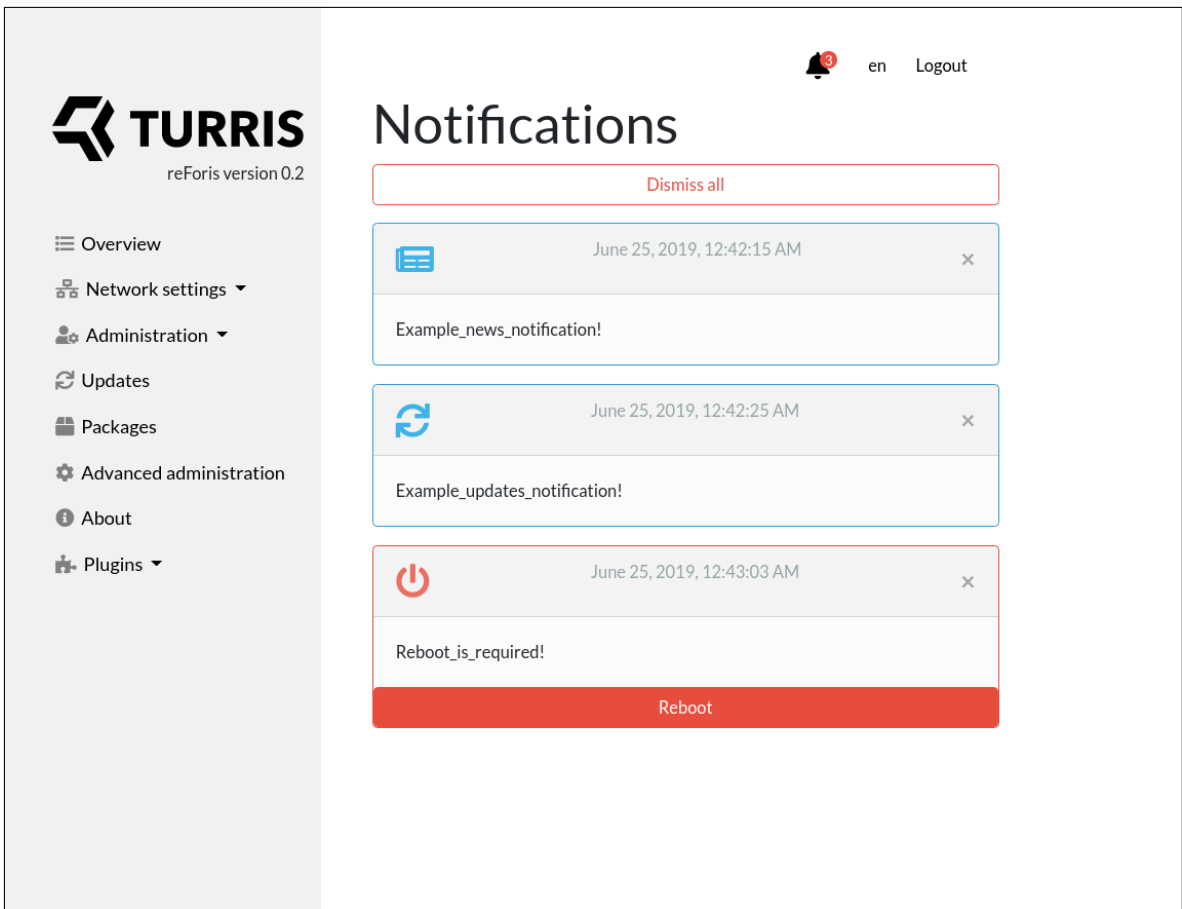


Figure F.3: *reForis*. Screenshot of the notifications page.

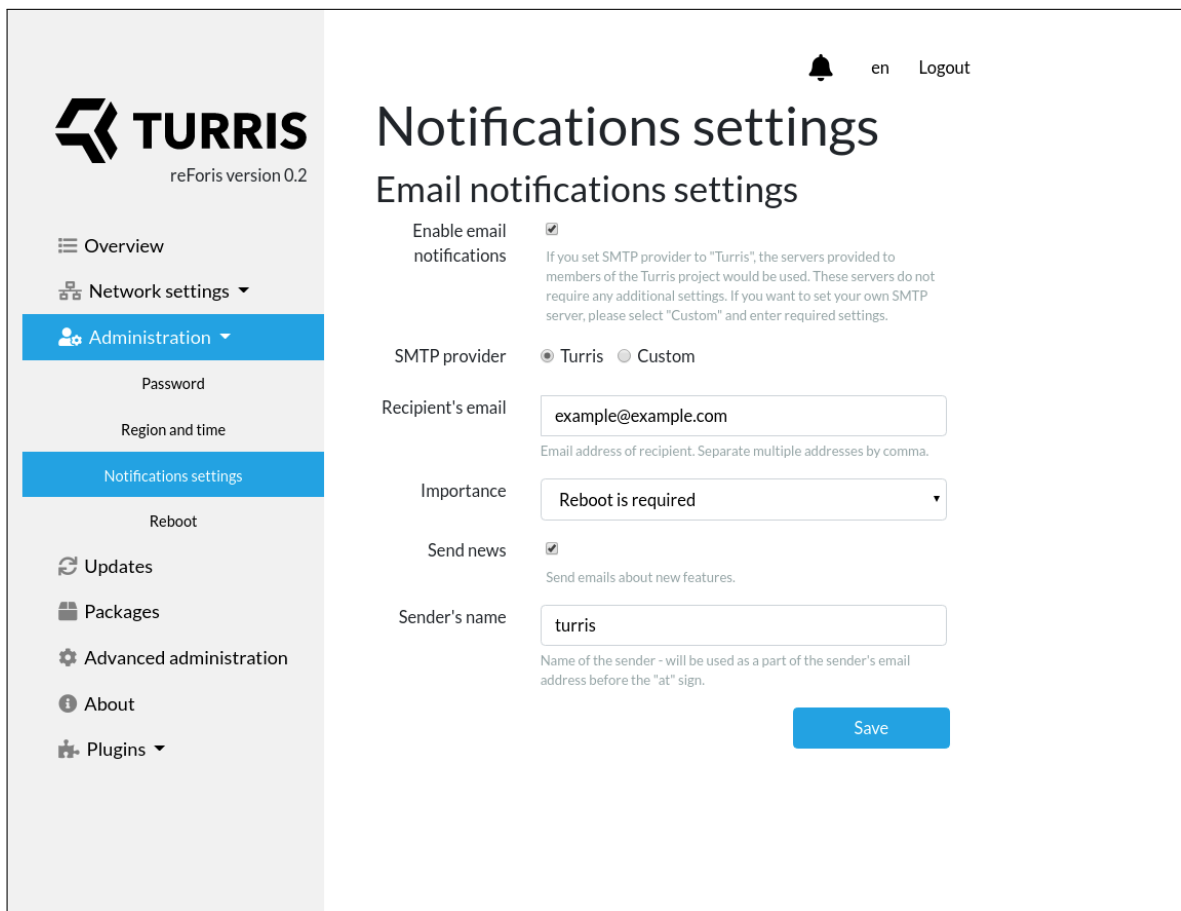




Figure F.4: reFORIS. Screenshot of the notifications page.




en
Logout

WAN

Here you specify your WAN port settings. Usually, you can leave this options untouched unless instructed otherwise by your internet service provider. Also, in case there is a cable or DSL modem connecting your router to the network, it is usually not necessary to change this setting.

WAN IPv4

IPv4 protocol DHCP (automatic configuration) ▼

DHCP hostname

Hostname which will be provided to DHCP server.

WAN IPv6

IPv6 protocol DHCPv6 (automatic configuration) ▼

Custom DUID

DUID which will be provided to the DHCPv6 server.

MAC

Custom MAC address

Useful in cases, when a specific MAC address is required by your internet service provider.

Save

Connection test

Here you can test you connection settings. Remember to click on the **Save button** before running the test. Note that sometimes it takes a while before the connection is fully initialized. So it might be useful to wait for a while before running this test.

IPv6 connectivity	✓
IPv6 gateway connectivity	✓
IPv4 connectivity	✓
IPv4 gateway connectivity	✓

Test connection again

Figure F.5: *reForis*. Screenshot of the WAN configuration page.

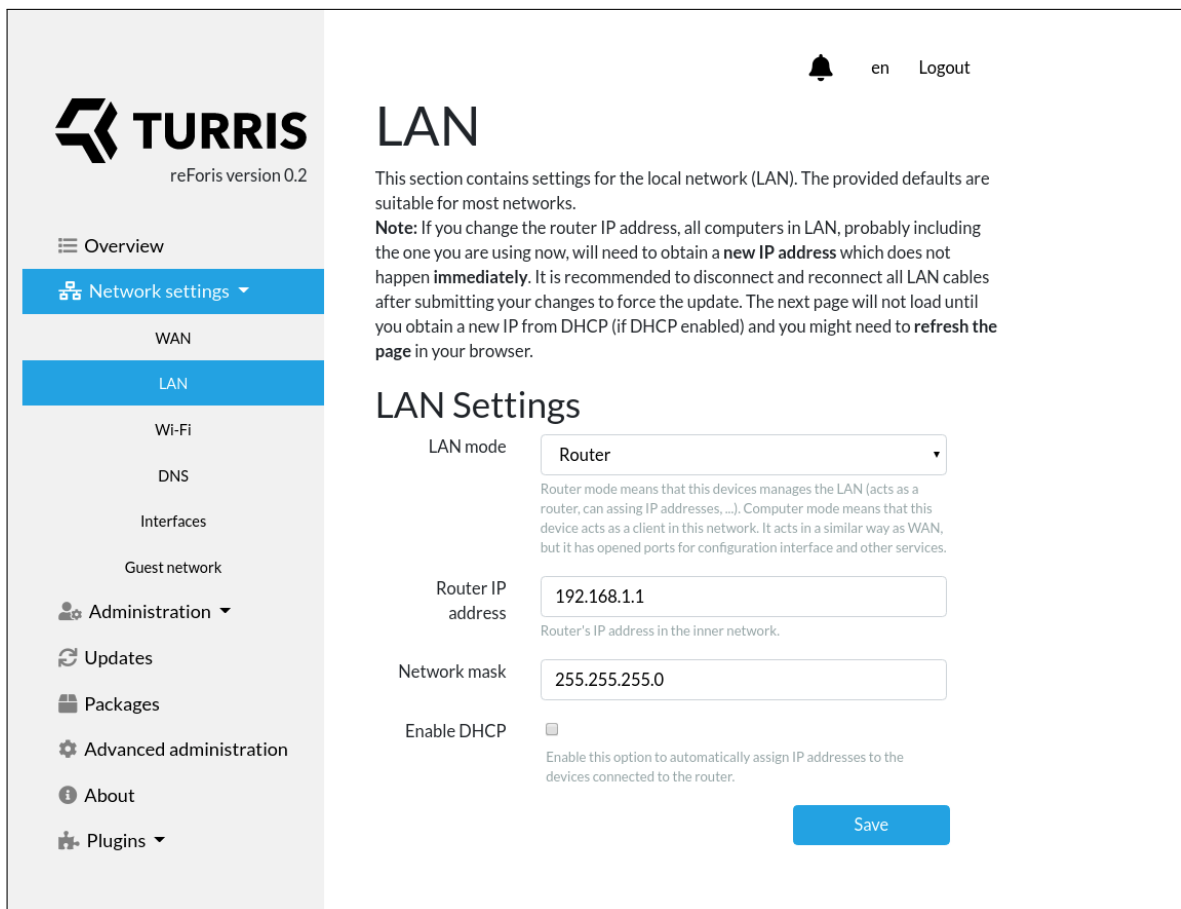




Figure F.6: reForis. Screenshot of the LAN configuration page.


en
Logout



reForis version 0.2

- Overview
- Network settings ▾
- WAN
- LAN
- Wi-Fi
- DNS
- Interfaces
- Guest network
- Administration ▾
- Updates
- Packages
- Advanced administration
- About
- Plugins ▾

Wi-Fi

If you want to use your router as a Wi-Fi access point, enable Wi-Fi here and fill in an SSID (the name of the access point) and a corresponding password. You can then set up your mobile devices, using the QR code available within the form.

Module 1

Enable

SSID

Password WPA2 pre-shared key, that is required to connect to the network.

Hide SSID
If set, network is not visible when scanning for available networks.

GHz 2.4 5
The 2.4 GHz band is more widely supported by clients, but tends to have more interference. The 5 GHz band is a newer standard and may not be supported by all your devices. It usually has less interference, but the signal does not carry so well indoors.

802.11n/ac mode
Change this to adjust 802.11n/ac mode of operation. 802.11n with 40 MHz wide channels can yield higher throughput but can cause more interference in the network. If you don't know what to choose, use the default option with 20 MHz wide channel.

Channel

Enable Guest Wifi
Enables Wi-Fi for guests, which is separated from LAN network. Devices connected to this network are allowed to access the internet, but aren't allowed to access other devices and the configuration interface of the router. Parameters of the guest network can be set in the Guest network tab.

Module 2

Enable

Save

Figure F.7: reForis. Screenshot of the Wi-Fi configuration page.

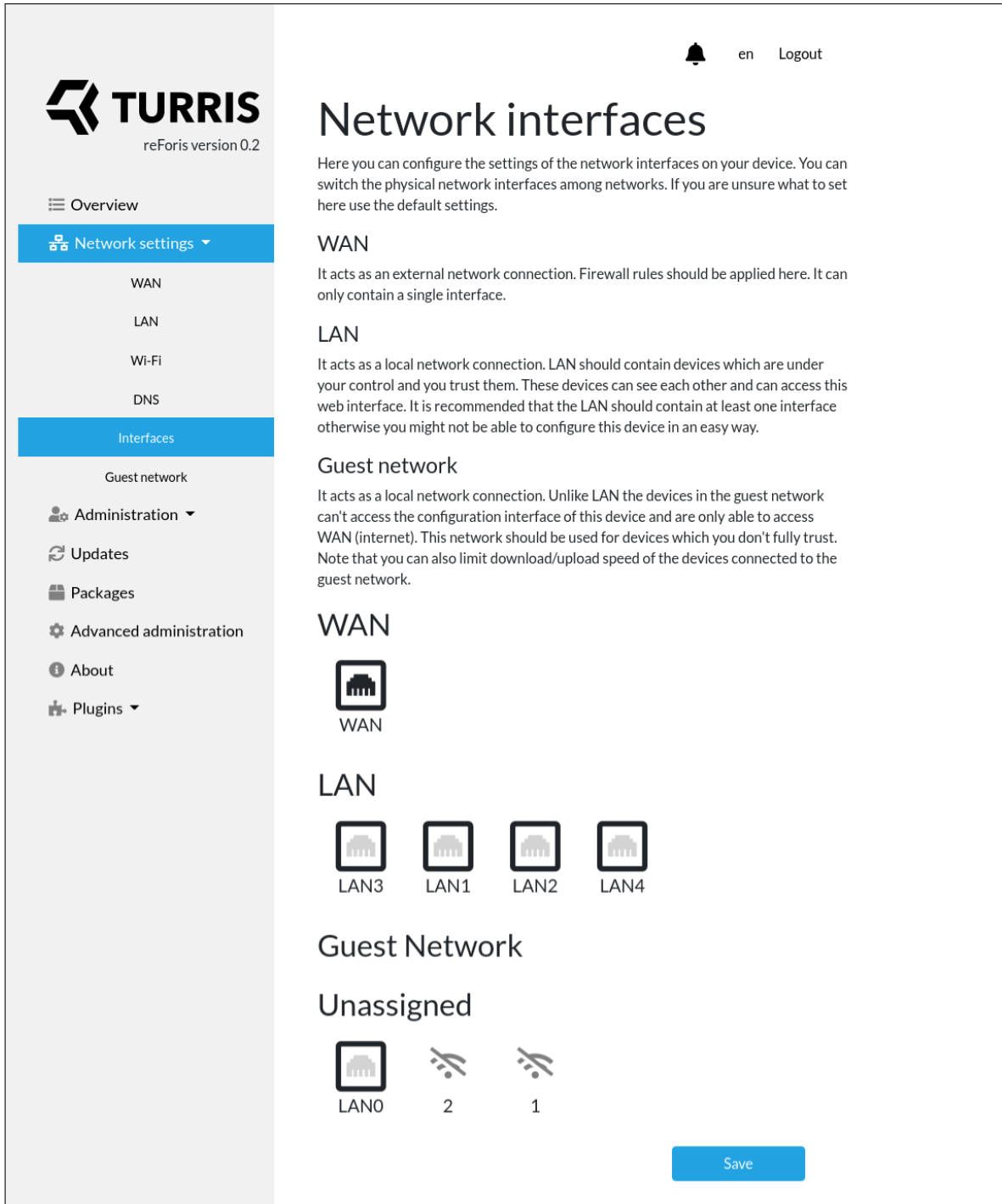


Figure F.8: reFORIS. Screenshot of the network interfaces page.

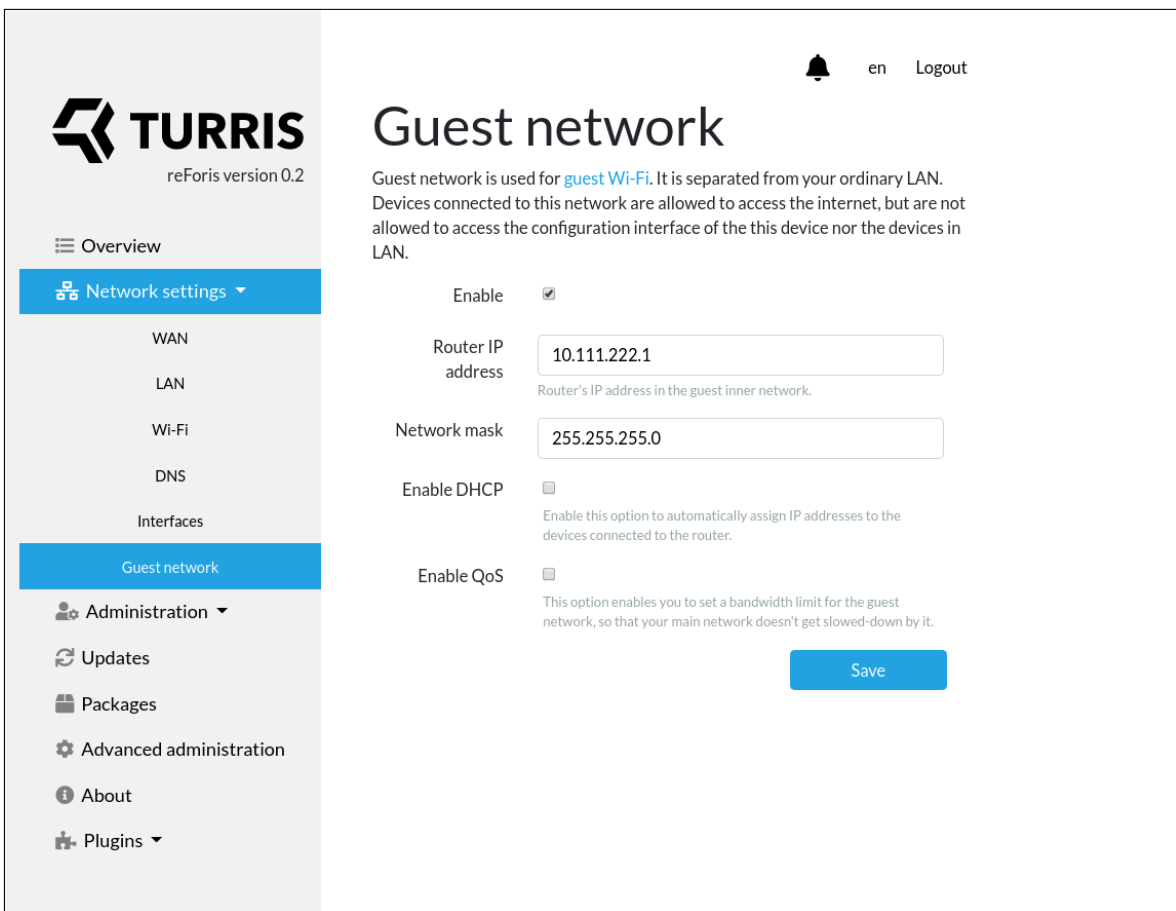


Figure F.9: reForis. Screenshot of the guest network configuration page.

TURRIS
reForis version 0.2

en Logout

DNS

Router Turris uses its own DNS resolver with DNSSEC support. It is capable of working independently or it can forward your DNS queries your internet service provider's DNS resolver.

The following setting determines the behavior of the DNS resolver. Usually, it is better to use the ISP's resolver in networks where it works properly. If it does not work for some reason, it is necessary to use direct resolving without forwarding.

In rare cases ISP's have improperly configured network which interferes with DNSSEC validation. If you experience problems with DNS, you can **temporarily** disable DNSSEC validation to determine the source of the problem. However, keep in mind that without DNSSEC validation, you are vulnerable to DNS spoofing attacks! Therefore we **recommend keeping DNSSEC turned on** and resolving the situation with your ISP as this is a serious flaw on their side.

Use forwarding

Enable DNSSEC

Enable DHCP clients in DNS
This will enable your DNS resolver to place DHCP client's names among the local DNS records.

Save

Connection test

Here you can test your internet connection. This test is also useful when you need to check that your DNS resolving works as expected. Remember to click on the **Save button** if you changed your forwarder setting.

DNS	✓
DNSSEC	C

Test is running...

Figure F.10: *reForis*. Screenshot of the DNS configuration page.

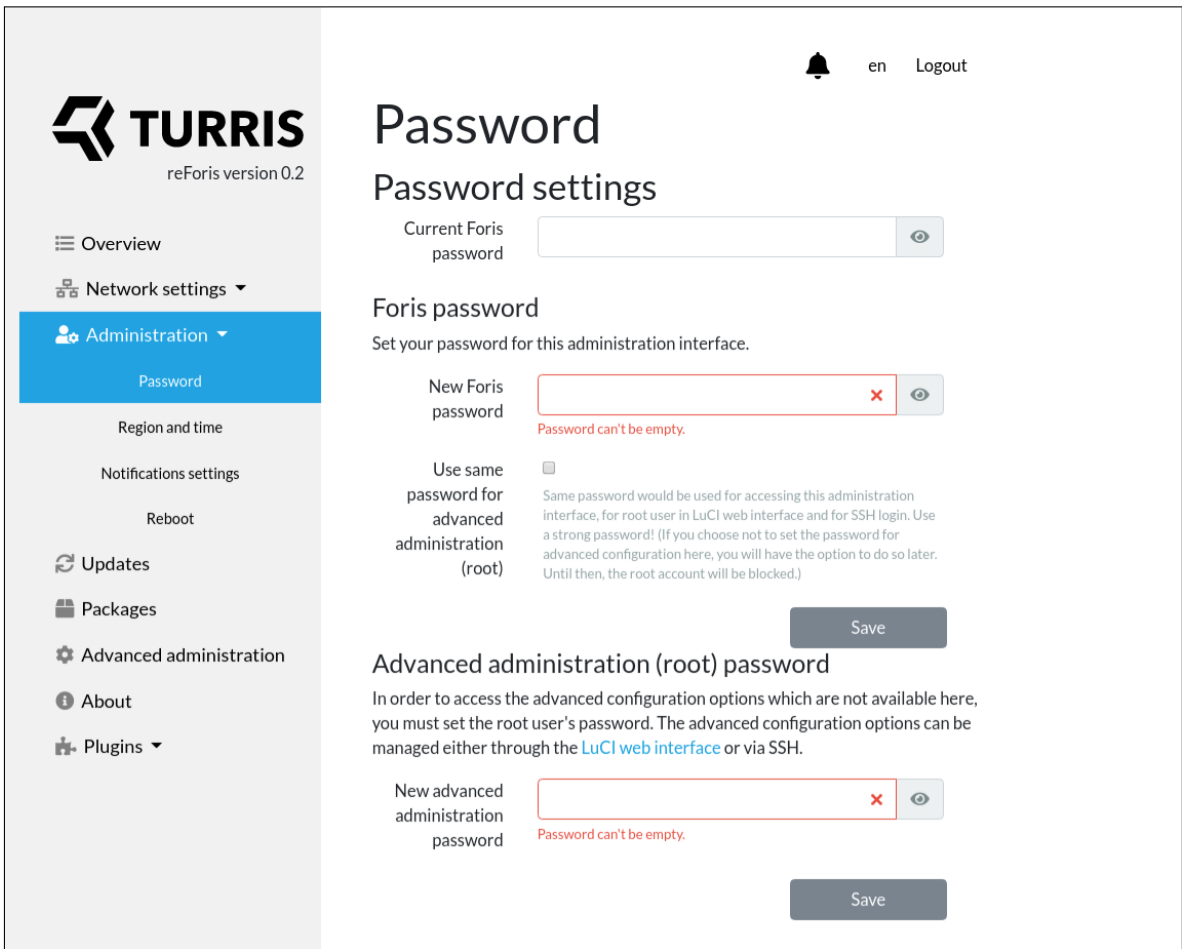


Figure F.11: *reForis*. Screenshot of the password settings page.

TURRIS
reForis version 0.2

en Logout

Region and time

It is important for your device to have the correct time set. If your device's time is delayed, the procedure of SSL certificate verification might not work correctly.

Region settings

Please select the timezone the router is being operated in. Correct setting is required to display the right time and for related functions.

Continent or ocean: Europe

Country: Czech Republic

Timezone: Prague

Time settings

Time should be up-to-date otherwise DNS and other services might not work properly.

How to set time: Via ntp

NTP servers

- 217.31.202.100
- 195.113.144.201
- 195.113.144.238
- 2001:1488:ffff::100
- ntp.nic.cz
- 0.openwrt.pool.ntp.org
- 1.openwrt.pool.ntp.org
- 2.openwrt.pool.ntp.org
- 3.openwrt.pool.ntp.org

Time: 2019-06-25 00:40:06

Save

Figure F.12: *reForis*. Screenshot of the region and time settings page.

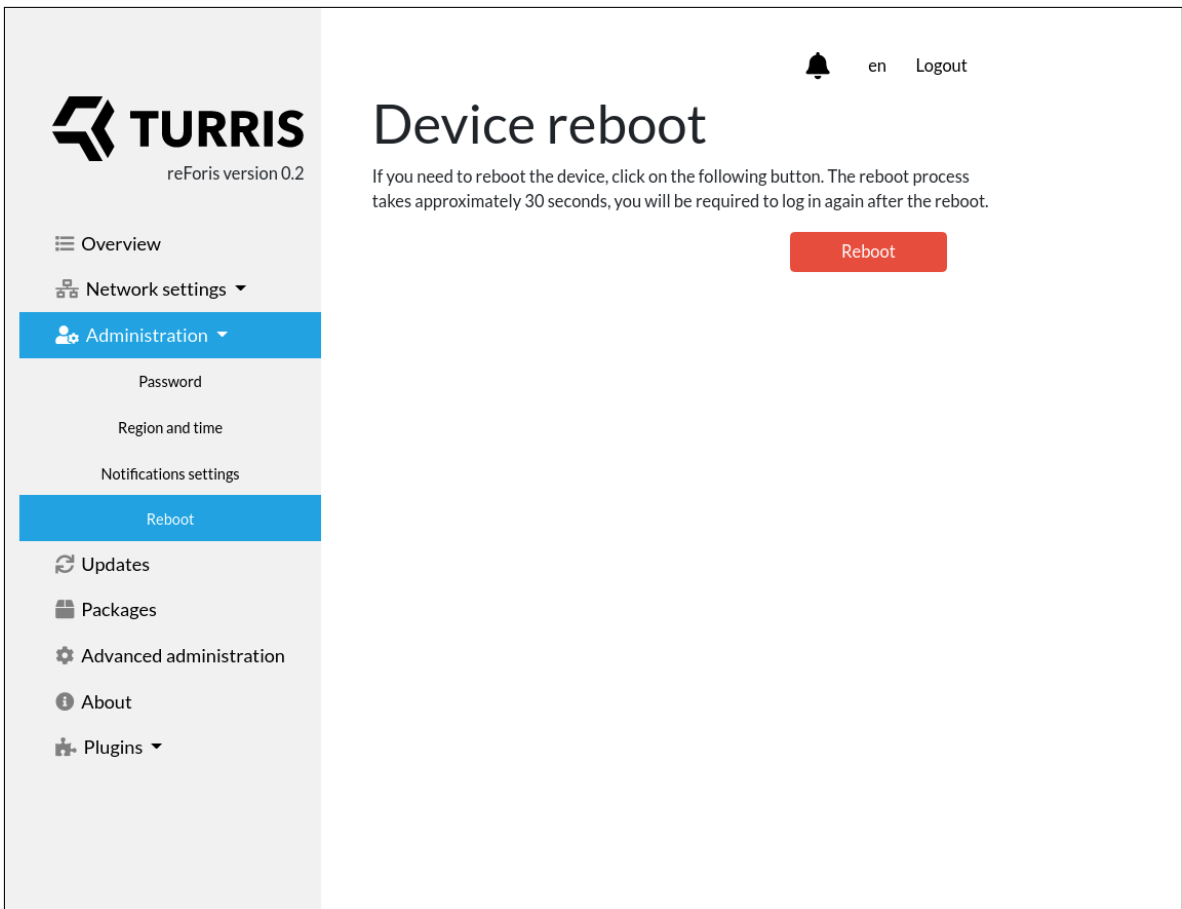


Figure F.13: *reForis*. Screenshot of the reboot device page.

TURRIS
reForis version 0.2

- Overview
- Network settings ▾
- Administration ▾
- Updates**
- Packages
- Advanced administration
- About
- Plugins ▾

en Logout

Updates

One of the most important features of router Turris are automatic system updates. Thanks to this function your router's software stays up to date and offers better protection against attacks from the Internet. It is **highly recommended** to have this feature **turned on**. If you decide to disable it, be warned that this might weaken the security of your router and network in case flaws in the software are found. By turning the automatic updates on, you agree to this feature's license agreement. More information is available [here](#).

Enable automatic updates
(recommended)

Update approvals

Update approvals can be useful when you want to make sure that updates won't harm your specific configuration. You can e.g. install updates when you're prepared for a possible rollback to a previous snapshot and deny the questionable update temporarily. It isn't possible to decline the update forever as it will be offered to you again together with the next package installation.

- Automatic installation
Updates will be installed without user's intervention.
- Delayed updates
Updates will be installed with an adjustable delay. You can also approve them manually.
- Update approval needed
You have to approve the updates, otherwise they won't be installed.


Automatic restarts after software update


Delay (days)
Number of days that must pass between receiving the request for restart and the automatic restart itself.

Reboot time
Time of day of automatic reboot in HH:MM format.

Save

Figure F.14: reForis. Screenshot of the updates settings page.


en
Logout



reForis version 0.2

- Overview
- Network settings ▾
- Administration ▾
- Updates
- Packages
- Advanced administration
- About
- Plugins ▾

Packages

Apart from the standard installation, you can optionally select bundles of additional software that'd be installed on the router. This software can be selected from the following list. Please note that only software that is part of TurrisOS or that has been installed from a package list is maintained by automatic updates system. Software that has been installed manually or using opkg is not affected.

Packages

Extensions of network protocols for 3G/LTE	<input type="checkbox"/>	Support for additional protocols and connection types.
Data Collection	<input type="checkbox"/>	Software for participation in data collection and dynamic distributed firewall.
Device detection	<input type="checkbox"/>	Software for detecting new devices on local network (EXPERIMENTAL).
NAS	<input checked="" type="checkbox"/>	Services allowing to connect a disk to the router and use it as network data store.
OpenVPN	<input type="checkbox"/>	An easy setup of OpenVPN server from Foris.
Pakon	<input type="checkbox"/>	Software for in depth monitoring of your traffic.
Print server	<input type="checkbox"/>	Services allowing to connect a printer to the router and use it for remote printing.
Tor	<input type="checkbox"/>	Service to increase anonymity on the Internet.

Languages

If you want to use other language than English you can select it from the following list:

- CS
- DA
- DE
- RU
- SK
- HU
- IT
- NB

Save

Figure F.15: *reForis*. Screenshot of the packages settings page. 161

The screenshot displays the 'About' page of the reFORIS web interface. On the left, a sidebar menu contains the following items: Overview, Network settings (with a dropdown arrow), Administration (with a dropdown arrow), Updates, Packages, Advanced administration, About (highlighted in blue), and Plugins (with a dropdown arrow). The main content area features the 'About' title and a table of system information. In the top right corner, there is a notification bell icon, the language 'en', and a 'Logout' link. The table lists the following details:

Device	Turris Omnia
Serial number	0000000B00013C14
Turris OS version	4.0
Kernel version	4.14.123

Figure F.16: *reForis*. Screenshot of the "About" information page.

Usability testing script and quizzes

G.1 Quiz before test

- How are you :)?
- Have you ever owned a router?
 - If yes - have you ever set it up by yourself?
- Do you have technical skills?
 - If yes - Have you ever set up network equipment in the past?
- Have you ever participated in usability testing?

G.2 Quiz after test

- What is your overall impression of the Foris interface?
- Were the scripts clear?
- What did you like about the interface?
- What did you not like about the interface?
- Do you have any suggestions for improvement?

G.3 Script

G.3.1 Introduction

Foris web application is a configuration interface for Turriss routers which provides access to the router network settings and maintains. The Turriss routers are more powerful and advanced than a regular home router.

You have a Turriss router at home, and you want to change some of its configuration. Please follow the steps below.

1. Login to the application with password **forispassword**.
2. Ensure that WAN connection is configured correctly.
3. Enable the first Wi-Fi Module and set it up with the same configuration as described in the initial configuration below.

5 GHz:

SSID: mywifi

Password: mywifipassword

4. Configure guest Wi-Fi network in the first Wi-Fi module with the following settings.

SSID: guestwifi

Password: guestwifipassword

5. Set Google as the DNS forwarder and ensure that DNS config is properly set.
6. Setup your actual country and timezone.
7. Install NAS package.
8. Restart the router.
9. You are all set, thank you!

G.4 Moderator script

G.4.1 Introduction

The Foris web application is a configuration interface for Turriss routers which provides access to the router network settings and maintains. The Turriss routers are more powerful and advanced than a regular home router.

You have Turriss router at home, and you want to change some configurations.

Please follow the steps below.

Is description of the product clear to the user? Answer questions.

Test starts from the login page.

1. Login to the application with password **forispassword**.
User enters the password into the form and clicks the login button.
2. Ensure that WAN connection is configured correctly.
User clicks on the “Network settings” dropdown menu and navigates to the WAN settings. After, the user clicks on the “connection test” button and waits for the results.
3. Enable the first Wi-Fi Module and set it up with the same configuration as in the initial configuration below.

5 GHz:

SSID: mywifi

Password: mywifipassword

The user navigates to the “Wi-Fi” page. Then he clicks on the “Enable” checkbox under “1 Module”. User fills the form using the settings provided. Then the user clicks the “Save” button.

4. Configure guest Wi-Fi network in the first Wi-Fi module with the following settings.

SSID: gueswifi

Password: guestwifipassword

In the same “Wi-Fi” page and the same module. User clicks on the “Enable guest network” checkbox. User fills the form using the settings provided. Then the user clicks the “Save” button.

5. Set Google as DNS forwarder and ensure that DNS config is properly set.
The user navigates to the “DNS” page. User clicks on the “Use forwarding” checkbox. The user chooses “Google” in the appeared select element. Then the user applies settings by clicking on the “Save” button. At the end of this step, the user clicks on the connection test button and waits for the results.
6. Setup your actual country and timezone.
User clicks on the “Administration” dropdown menu and navigates to the “Region and time” page. User chooses his region, country, and city

in the select element. Then the user applies settings by clicking on the “Save” button.

7. Install NAS package.

The user navigates to “Packages” and notices an alert about automatic updates that should be enabled to manage packages. Then the user navigates to the “Updates” page and enables automatic updates. After that the user returns to the “Packages” page where he finds an NAS package and enables it. Then the user clicks the “Save” button.

8. Restart the router.

The user navigates to the “Administration” page, then scrolls down to “Device reboot”. Then the user presses the “Reboot” button and confirms action in the alert. The user waits until the reboot is done.

9. You are all set, thank you!

Acronyms

API	Application Programming Interface
CD	Continuous Delivery
CDN	Content Delivery Network
CI	Continuous Integration
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
IoT	Internet of Things
LAN	Local Area Network
M2M	Machine-To-Machine
OS	Operating System
RAM	Random-Access Memory
regex	Regular Expression
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience

H. ACRONYMS

VPN Virtual Private Network

WAN Wide Area Network

WS WebSockets

Contents of enclosed microSD card

```
├── readme.txt..... the file with disk contents description
├── thesis.....the directory of LATEX source codes of the thesis
├── DP_Bogdan_Bodnar.pdf ..... the thesis text in PDF format
├── src.....the directory of source codes
│   ├── reforis..... the directory of source codes
│   │   ├── docs.....the directory of the Flask application documentation
│   │   ├── js.....the directory of the React application source codes
│   │   └── docs.....the directory of the React application documentation
│   └── reforis_diagnostics ..... the directory of demo plugin source code
└── usability_testing.....the directory with usability testing records
```