Department of Informatics
Technical University of Munich

TECHNICAL UNIVERSITY OF MUNICH

DEPARTMENT OF INFORMATICS

INTERDISCIPLINARY PROJECT IN ELECTRICAL ENGINEERING IN INFORMATICS

Extension of the virtual iLab-Isle

Lars Wüstrich

Technical University of Munich

Department of Informatics

Interdisciplinary Project in Electrical Engineering in Informatics

Extension of the virtual iLab-Isle

Erweiterung der virtuellen iLab-Insel

Author:	Lars Wüstrich
Supervisor:	Prof. Dr.-Ing. Georg Carle
Advisor:	Dr. Marc-Oliver Pahl
	Stefan Liebald, M. Sc.
Date:	August 31, 2018

I confirm that this Interdisciplinary Project in Electrical Engineering is my own work and I have documented all sources and material used.

Garching, August 31, 2018

Location, Date

Signature

Abstract

The virtual iLab isle – also called vLab – is a virtualization of the physical iLab isle that is used as part of the practical courses iLab1 and iLab2 at the Technical University of Munich. In the interdisciplinary project, the vLab is extended in order to make it more usable for students of the newly created Massive Open Online Course (MOOC) "iLabX". In order to achieve this goal, the vLab Control Center (VCC), a graphical user interface that integrates the content of the MOOC into the vLab, is implemented. In addition to that the VCC is offering the possibility for students to directly provide feedback to the instructors either into a forum that is hosted on edX – the platform where the MOOC is also located at – or to report bugs that occurred to the creators of the course directly. Another effort to make the system more usable is to remove unneeded software and to integrate new software that is needed throughout the course. Finally, since this system will be used by a variety of users he according documentation for both the vLab and the VCC are created. This document explains all design decisions in detail that were taken during the development of the introduced software. Furthermore it describes how the vLab was adapted in order to integrate it into the MOOC.

Zusammenfassung

Die virtuelle iLab Insel ist eine Virtualisierung der physischen iLab Inseln welche in den Praktika iLab1 und iLab2 eingesetzt werden um inhalte im Bereich von Computer-Netzen zu vermitteln. Die Virtualisierung enthält jegliche Möglichkeiten die nötig sind um komplexe Netzwerke in einer lokalen Umgebung zu emulieren ohne dafür zusätzlich Hardware zu benötigen. In diesem interdisziplinären Projekt wird die virtuelle Maschine erweitert um sie in einem neu eintwickelten Massive Open Online Course (MOOC) zu integrieren und einzusetzen. Dazu wird die virtuelle Maschine um das so genannte vLab Control Center (VCC) erweitert um die Nutzbarkeit für Studenten des MOOCs zu erhöhen. Das VCC beinhaltet Funktionalitäten die die Inhalte des MOOCs integrieren und aktuell halten und bietet den Studenten die Möglichkeit Feedback ohne Umwege an die Veranstalter des Kurses zu liefern. Ein weiterer Schritt um die Nutzbarkeit zu erhöhen ist die Beseitigung von nicht verwendeter Software sowie die Integration von neuer Software die von Teilen des Kurses benötigt wird.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

After some time on the sideline, Massive Open Online Courses (MOOCs) have taken over public attention. There are MOOCs for any subjects and fields, starting from Music Production to Software Engineering. Many courses are created by institutions like Harvard, Berkeley, Stanford and Georgia Tech. Some of those Universities are even offering degrees that can be obtained without ever physically attending a class [19] [10]. Even though most of the platforms where such MOOCs are hosted offer lots of options to test the content that is being taught, many courses in the field of computer networks only teach theoretical aspects and lack practical applications. In 2016, the Technical University of Munich (TUM) has also begun to add some new Online Courses for the online platform *edX*, one of the leading platforms for such courses. One of these newly developed courses is called "iLabX". It aims to teach both the theoretical and the practical parts of basic networking. These basics include the concepts of addressing on different layers of the the communication stack or explain how routing between different devices work. In contrary to the approach that most MOOCs in this field are taking, iLabX tries to teach the theory and has students practice the taught knowledge. In order to enable students to solve the tasks given in the practical part, a specifically for the course developed environment. This environment is capable of simulating network setups that are needed to demonstrate how the concepts that are taught in the MOOC work. The current version of this environment provides most of the needed functionality that is required to complete the tasks from the MOOC but has some deficits with regards to usability. This project aims to improve the usability of the environment such that users of all skill levels can make of it. To maximize the usability of the environment, content from iLabX should be directly integrated into it.

The project is split into the following parts: First of all, the existing version of the environment needs to be assessed such that existing bugs can be found and be fixed. Afterwards, the environment is supposed to be extended by the *vLab Control Center* (VCC), an application that integrates the content of the MOOC into the environment, which should to make it easier for students to work on the exercises, get the latest news and to give feedback to the course instructors. In order to see what requirements the VCC must fulfill to make the vLab more usable, an analysis of the existing system is done in Chapter 2. Afterwards, the implementation of the VCC and how it fulfills the specified requirements is explained in Chapter 3. This Chapter also includes an evaluation by applying commonly used heuristics to the VCC to measure the quality of its usability. In Chapter 4 it is discussed, how the vLab got changed in order to provide the necessary functionality to the students as well what changes needed to be made to ensure that programs run without crashing. Finally, in Chapter 5 it is described what tutorials are available and where they can be found. The final Chapter 6 evaluates how the new version of the vLab has improved and what can be done in the future to enhance the user experience even more.

Chapter 2

Analysis

One of the main features of the MOOC *iLabX* is that it aims to teach both the theoretical and the practical aspects of networking. In order to make this possible in the scope of an online course, the physical isles of the practical courses iLab1 and iLab2 have been virtualized. Section 2.1 describes the current state of the virtual environment.

2.1 Previous Work

In Section 2.1.1 the original physical isles are that have been virtualized introduced. The resulting virtualization that has been developed before is described in 2.1.2. Finally, the requirements that are needed to integrate the vLab into the MOOC are explained in Section 2.2.

2.1.1 The physical iLab-Isle

The physical iLab-Isles were originally created for a practical course iLab1 that is offered at Technical University of Munich. This course teaches different aspects of networking. Topics include routing and different network protocols that are used in the context of networking. In order to teach the different aspects of networking, the course uses a concept that imparts the knowledge in three parts. The first two parts – a lecture and the so called PreLab teach the theoretical aspects of the content. In the third part – the Lab – the physical iLab-Isle is needed to show the concepts work in practice. The physical iLab-Isle consists of six Linux computers where three of them are connected to a display and a keyboard. These computers can be connected to two switches and

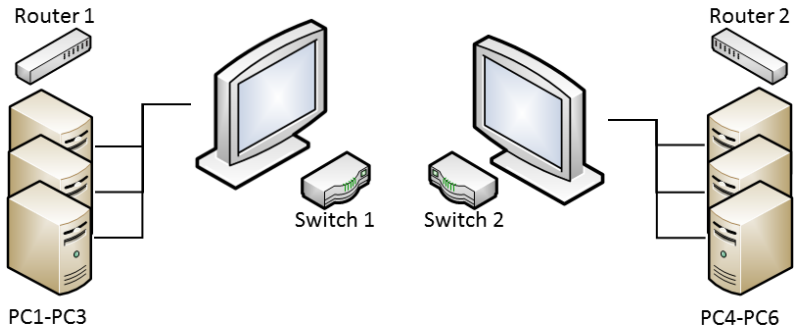

Figure 2.1: Schematic of a physical iLab isle

two routers. During a Lab, students use cables to connect the devices to set up a specific topology of a network that is used to demonstrate how specific mechanisms work through different examples.

A schematic of the described environment that is given to iLab students can be seen in Figure 2.1. The operating system running on the computers of the isle is called *iLab-Operating System* (iLab-OS) which is based on the Linux distribution Debian. To fit the needs of the practical courses it has been heavily customized. This customization included stripping out programs that were not needed to complete the tasks while other software packets were added such that students can configure the computers accordingly without having to install additional programs. Even though the system has been changed, it still offers a familiar environment to work in.

2.1.2 The virtual iLab-Isle

In a preceding interdisciplinary project [16] the environment of the physical iLab-Isle was virtualized into a single virtual machine – the *vLab*. Since the vLab is a virtual machine it can be run on many computers. It is provided as an image that can be imported into VirtualBox, an application that is capable of virtualizing an operating system within another operating system. This means the vLab can be run as an application on many computers that have processors based on x86 technology [18]. Since VirtualBox exists for the three majorly used operating systems Windows, Linux and macOS the vLab can be used by a large audience of users, independent from what operating system they use. Using a virtualized operating system also has the advantage that the guest operating

system is isolated from the system that is running VirtualBox. This means students can not do any harm to their own operating system while working within the vLab. After importing the vLab into VirtualBox, it can be started without any further installation steps. After starting the vLab, the user is presented with a Linux environment. Since the vLab is based on the earlier introduced iLab-OS, it comes with preinstalled software that is related to networking. Tools like Wireshark are accessible directly from the sidebar. The main functionality that is used in the vLab is the Common Open Research Emulator (CORE)[1][2]. The CORE application makes it possible to emulate complex networks within the vLab. To achieve this, CORE session can create isolated containers that can be used as if they were real computers. The containers that are part of the setup can be connected via simulated cables to each other. In addition to that, CORE is also capable of emulating switches, hubs and routers that can also be used in a setup. Therefore, except for the WiFi capabilities, the same setups that can be constructed with the physical isle, can also be created within CORE without the need for special hardware. Since CORE has no physical limits like the physical iLab-Isle even more complex setups can be created within CORE. Each container can be accessed by a shell, as if a SSH connection was established to it via the Internet. In addition to that it is possible to run applications on each container and configure it independently from the others. A picture of the vLab with a started CORE session can be seen in Figure 2.2. In addition to that CORE has been adapted such that Wireshark can be started directly on each of the containers without using the shell but by clicking on the container in CORE and then selecting the program to launch it. In this IDP the vLab is integrated into iLabX as an external tool such that students participating in the online course can also do practical tasks without requiring any special hardware.

2.2 Integrating the vLab to the MOOC

The aim of this IDP is to integrate the environment described in 2.1 into the MOOC. Since this course is not only targeted at people who are familiar with Linux environments but also to students of other skill levels, the vLab needs to be improved with regards to usability. In this section it is discussed what is required to achieve this. Similar to the iLab course, iLabX is split into a theoretical and a practical part. The vLab is only used throughout the practical part of the MOOC. In order to teach the practical aspects of the course, each part has a corresponding setup that can be imported and started in CORE. Students that are using the vLab provided by [16] need to download these setups, import them into CORE and then start a CORE session. This procedure is error prone since it has many steps that can fail. For example, students could download

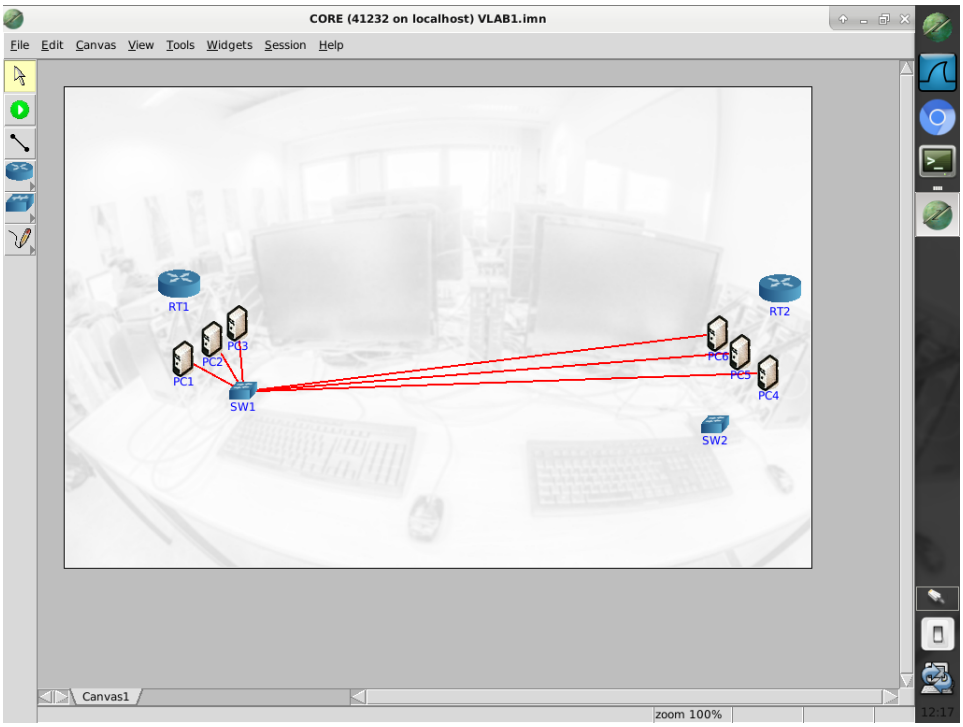

Figure 2.2: A picture of the vLab-OS with an open CORE session as shown in [16]

the wrong setup or fail to import the setup into CORE. In case students download all setup files at the beginning, they need to check if the files have changed throughout the course and need to keep them up to date. In order to make the vLab more usable these files should directly be integrated into the virtual machine and updated accordingly. In addition to that, to make to system more accessible to users that have not used a Linux environment before, the process of setting up the environment should be shortened and made more straightforward. During a run of the MOOC, the instructors of the course also make announcements about changes or give out information about changes through specific channels. In order to stay informed students need to check these channels by themselves. Since the provided channel is not integrated into edX, announcements may stay unnoticed by students. In order to make it easier for students to know about latest changes without having to check an external source, the channel should also be integrated into the vLab. Finally, since all students that take part in iLabX need to use the vLab it is important that all software that needs to be used runs without crashing and is available to students without any further installations. In case that students encounter problems or bugs they should have the possibility to inform the organizers through defined channels about these. In order to answer most of the questions before they occur, users need to be able to read about the functionality and potential problems

in some specified documentation. This has also the advantage that most of the problems that may arise during the course and have already been resolved in the documentation, do not have to be answered by the staff again. This results in the following requirements that need to be fulfilled in order to make the vLab more usable such that it can be integrated into the MOOC.

The first requirement that needs to be fulfilled is the integration of content from the MOOC into the vLab (R1). In order to achieve this, setup files that are needed for the practical tasks should be automatically downloaded and kept up to date within the vLab (R1.2) such that students can directly use them without having to download the files by themselves. The procedure to set the environment up should be simplified such that students can start working on the problems of the course with as few effort as possible (R1.1). In addition to that, the channel that is used by the instructors to make announcements to the students of the course should also be directly integrated into the system (R1.3) to make the distributed information more accessible to the students. Finally, students should also be able to give feedback from within the vLab. Therefore a channel has to be implemented that makes it possible to report and ask related questions directly from the system (R1.4). Since students might use the vLab on a machine that is not always connected to the Internet, the functionality described above should also be provided even when it is offline (R1.5). Since all of the requirements that are included to (R1) are related to the MOOC it is reasonable to provide the functionality within a single application that is running in the vLab. This application is called the vLab Control Center (VCC) and implements solutions that fulfill (R1). One of the main purposes of the VCC is to enhance the usability of the vLab for users of all skill levels. Therefore it does not only need to be functional but also must be easy to use. This should be achieved by adding an intuitive and fast user interface to the VCC. This implementation is explained in Chapter 3. Finally, the vLab itself should be stable and software should run without crashing and should be already provided within the environment (R2) to make it easier for users to complete the course. In order to achieve this programs that are used throughout the course need to be tested for compatibility with other programs already present in the environment. In addition to that it has to be made sure that also customized or self-implemented tools are available to the users without requiring them to install additional software. These changes are described in Chapter 4. As mentioned before, new users to the system need documentation (R3) to be able to use all the provided functionality. This documentation needs to inform users about different features of the environment as well solutions to common problems that can occur when using the vLab. The realization of (R3) is explained in Chapter 5.

The requirements explained above are listed in the following:

(R1) Integration of MOOC Content

- (R1.1) Starting of the CORE setups with as few effort as possible
- (R1.2) Integration of the latest iLabX setup files
- (R1.3) Integration of news and announcements that are related to iLabX
- (R1.4) Channel to report bugs and ask questions
- (R1.5) Offline usability

(R2) Stability and completeness

(R3) Documentation

Chapter 3

The vLab Control Center

The vLab Control Center (VCC) is the main extension of the vLab to integrate the virtual machine into iLabX. Its main purpose is to improve the usability of the vLab for students that take part in the online course by integrating relevant content into the virtual machine and making it easier to use the environment. This should be achieved by implementing the functionalities that are described in Chapter 2.2. Before these features can be implemented it is important to consider what technologies can be used to realize the VCC. In order to improve the usability even more, all these requirements need to be implemented while using a fast and intuitive user interface. In order to realize (R1), the VCC must be able to do the following things:

1. Receive information from specified endpoints on the Internet
2. Cache data locally such that the VCC is still functional when no Internet connection is available
3. Open local applications that are installed on the machine
4. Display information from the edX forums
5. Send feedback data to a provided backend
6. Provide a fast and intuitive Graphical User Interface

In this chapter it is described how (R1) is realized within the context of the VCC. Section 3.1 discusses different possibilities for the technologies that can be used to implement such a program. After this, Sections 3.2, 3.3, 3.4 and 3.5 discuss how the different features that are required for fulfilling (R1) are implemented. Finally, the user interface is evaluated with regards to usability in Section 3.6.

3.1 Technology used for building the VCC

There are lots of different technologies available to realize an application like the VCC. Before the VCC was implemented the two technologies discussed in the following were considered:

1. Python 3 with PyQt
2. Electron

Since the VCC is supposed to be a local application that is running in the vLab, it makes sense to consider a programming language that is directly integrated into the system and does not need any additional installations. One of these is the Python programming language[11]. "Debian is the largest integrated Python distribution"[12], which means that the language is directly integrated into the vLab since it is based on Debian. It is a high-level language that is also capable of object orientated programming. It is widely used and has applications in many different scientific fields. Therefore, lots of modules exist that provide solutions for many problems. These modules can be imported to make the implementation of the features in the VCC easier. An implementation of the VCC therefore would not require additional software that needs to be installed in order to run in the vLab. Since Python is already integrated into the system the chances for compatibility issues are very low since all software that is included in a Debian image is tested extensively with these regards before. Another advantage of using Python for implementing the VCC is that it is a scripting language. This means one does not have to compile the finished application into an executable file. It is sufficient to store the code for the VCC directly in the file system of the vLab instead to be able to run the application. As mentioned above, there are lots of modules that can easily be imported into a Python project. By using this lots of functionality, like methods for networking or executing commands in a shell, can be imported with little effort. A VCC that is written in Python therefore could simply use existing modules to start applications, store data in the file system or fetch data from a source on the Internet, making it easy to implement features 1,2,3 and 5. In addition to that information from edX can be received via the provided API with reasonable effort. As it can be seen using Python to implement the VCC can have a lot of advantages.
On the other hand it has limitations when it comes to building graphical user interfaces. Even though there are also modules like PyQt that make it possible to create user interfaces, it is really hard to create a modern environment with the given tools. Even when this succeeds it is hard to extend the interface easily when new functionality is added to the system that also needs to be integrated into the user interface. Another drawback

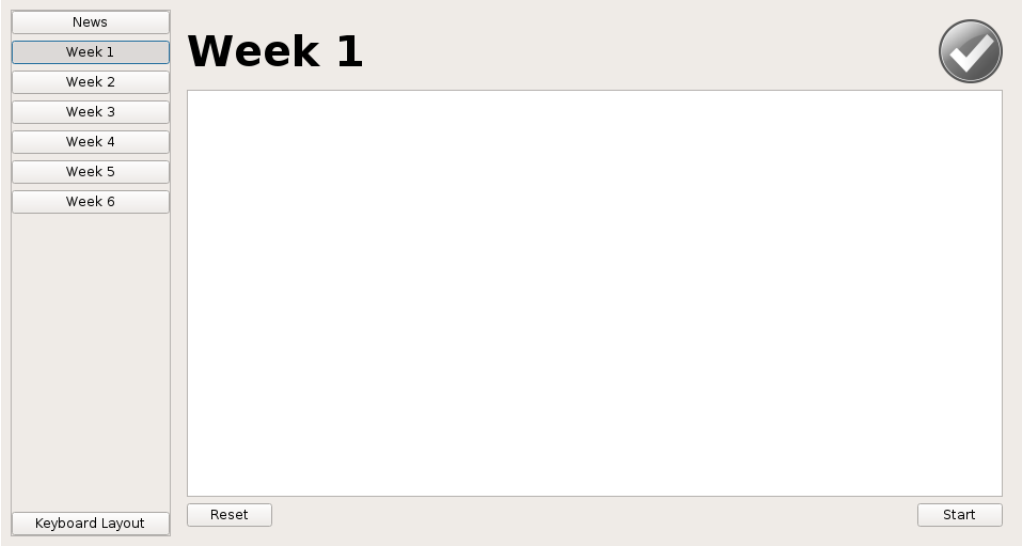

Figure 3.1: Selection of a Week in a prototype of the VCC written in Python

of using the Python language is the lack of support for integrating web content. To be able to render websites within a user interface that is implemented in Python an additional webkit needs to be installed. Those webkits are often out of date and lack the support for the latest web technologies like JavaScript ES6. This may result in content that is not displayed correctly or not at all. This functionally is needed in the VCC to integrate edX directly into the application. Despite all of this, a first prototype of the VCC was developed in Python. This prototype was capable of displaying news and the descriptions of different parts as well as starting CORE with the corresponding configuration. In Figure 3.1 it is shown how this prototype looked like when a week to work on was selected. It can be seen clearly that this interface is functional but is lacking the look of other modern applications. Therefore as a second alternative, the Electron framework [4] was considered. This framework makes it possible to create desktop applications by using web technologies like HTML, CSS and JavaScript. Some prominent examples for applications that are implemented using Electron are the *Atom* text editor or *Visual Studio Code*. This applications demonstrate that it is possible to develop functional desktop applications with Electron. Since Electron is written in JavaScript it is necessary to install additional software in the vLab to be able to run a VCC based on this technology. This includes dependencies needed to run Electron as well as the application itself. One advantage of using Electron, and therefore web technologies, for implementing the VCC that JavaScript is also commonly used [17]. Therefore also many modules exist that can be used to provide lots of functionality without much effort. Using modules to provide functionality also often has the advantage that popular

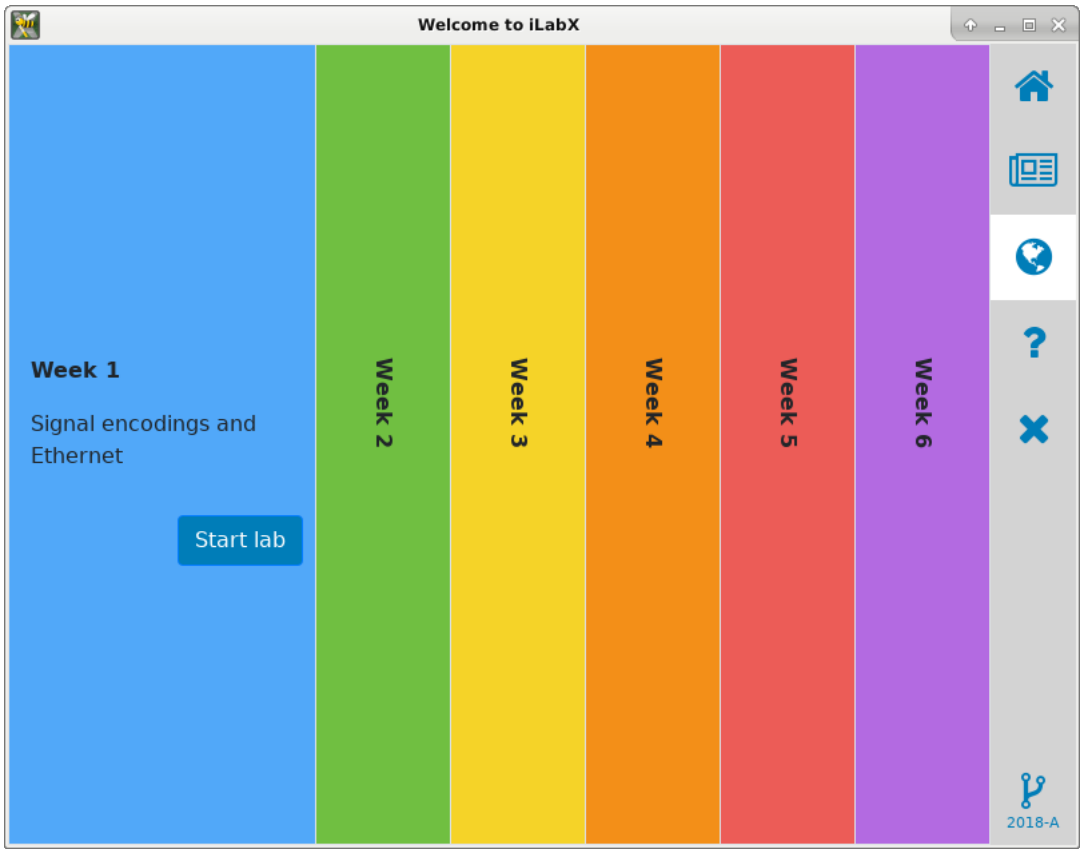

Figure 3.2: Selection of a Week in a version of the VCC that is based on Electron

modules contain less bugs than a self-implemented version of the feature. One drawback of using web technologies is that most of the functionality provided in modules is aimed at the use in web browsers and not made for desktop applications. Even though, there are modules that make it possible to execute shell commands. Therefore it would be possible to implement all the needed functionality like running applications in the vLab control center. Another advantage of using Electron is the built-in web kit it comes with. This makes it easy to integrate edX directly into the VCC as it can render the websites like they were normal parts of the VCC. In addition to that, when using HTML, CSS and JavaScript to build the VCC it is possible to design fast and modern user interfaces. An example for this modern interface can be seen in Figure 3.2 where also a Week/Part of the course is selected to start the corresponding setup.

As it has been discussed before, using either Python or Electron for realizing the VCC has advantages and disadvantages. One reason to use Python would be that it is directly integrated into the vLab since it is based on Debian, the largest distribution that directly

integrates the programming language. In addition to that Python comes with lots of modules that make it easy to add the required functionality to the VCC. The major drawback of Python is that it is hard to create modern user interfaces that can also be easily changed afterwards when some of the functionality changes. In addition to that python lacks the support of reliably integrating web content as there are no web kits available that support the latest web technologies. This is where Electron has its advantages. Since it is a technology that makes it possible to create desktop applications by using JavaScript, HTML and CSS it is easy to directly integrate whole websites into the application. Since JavaScript is also widely used there are, like in Python, a lot of modules that can be used to add functionality to the VCC. Even though these are mostly limited to the usage of the browser, some of them provide the functionality needed to issue shell commands, making it possible to implement solutions for the functionality that requires interaction with the vLab. These advantages and disadvantages can also be seen in Table 3.1.

	Advantage	**Disadvantage**
Python	Integrated in the VM	Webkits not working properly
	Many modules available	GUI hard to develop
	Lightweight	
Electron	Webkit integrated	Additional Software needs to be installed
	UI can easily be created	
	Many modules available	Most modules limited to browser

Table 3.1: Advantages and disadvantages of Python and Electron

After studying the pros and cons the Electron outweighs Python since it makes it easier to create intuitive and fast user-interfaces, even when this comes at the cost of having to put more effort into the features that require interaction with the vLab. In the following subsections it is explained how the different features of the VCC are implemented in order to fulfill (R1) identified in Section 2.2.

3.2 Distributing news

The first feature that is implemented in the VCC is (R13), namely the integration of news and announcements that are related to iLabX. In order to fulfill this, the VCC needs to fetch the content from an online source and display the content accordingly. News and announcements can have different structures depending on the type. For example, they can be consisting only of text, but can also include pictures or videos. Before this data can be integrated and displayed in the application, it needs to be distributed.

Of course there are many different ways to distribute data. In this documentation the different possibilities are divided into two categories that can be used to distribute data:

1. self hosted and
2. third party hosting options

The advantages and disadvantages of these options are be described in the following subsections.

3.2.1 Self hosted

The first possibility is to host services by yourself. This has the advantage of being in full control over the content that is distributed. One can decide which content is shown and who can access the data. In addition to that, one can decide how the data is displayed to the users. This comes at the cost of having to maintain the corresponding resources that are necessary to make the data accessible to others. The servers hosting the service need to be maintained, kept up to date and have to be hardened such that they are not vulnerable to external and internal attacks. Generally, self-hosting a service can be a complicated task.

When deciding to use a self-hosted way to distribute news and announcements, the following two options were considered – using an already existing blog or hosting a RSS (Rich Site Summary) feed. Using the blog located at `vlab.net.in.tum.de` would have the advantage of using a service that is already set up. In addition to that using this method would not introduce more maintenance overhead for a new service. On the other hand, using a blog platform is not a good format to distribute news as the format of a blog post differs from news entries. Therefore this approach was discarded. The second possibility considered for a self-hosted service was to set up a RSS-feed. The RSS format is often used to inform subscribed users about any updates or changes of the source that has set up the feed, e.g. a website. RSS feeds follow a strictly specified format [13] that is a flavor of the XML language. Another advantage of using RSS is that since it follows this straight forward format, hosting it is not complicated. It is sufficient to display the corresponding XML file that contains the news and announcements as items in order to set it up. Users then can get the information by either accessing the file directly or using an RSS reader. Finally, there are also modules that can be imported in JavaScript that are able to parse RSS feeds to further process and display the data. This can be used in the VCC to integrate this feature into the vLab. Another advantage of using RSS is that it is fully opt in. This means only people who want to receive the information can subscribe to the feed. Since the feed is available from a source outside of the vLab,

they can also subscribe to the feed on other devices than the vLab. This also means that people who are not subscribed do not receive the announcements and can rely on getting the information delivered into the VCC. A big disadvantage is that maintaining the RSS feed without additional tools adding new announcements can be a tedious task since every time, the XML has to be edited by hand. In addition to that, the kind of information is limited to text and links by default, making it hard to integrate videos or images directly into the feed.

3.2.2 Third party hosted

As an alternative to hosting the service that distributed news and announcements on your own, one can also rely third parties to provide the needed infrastructure. This has the advantage of not having to set up the infrastructure and maintaining and hardening the service against malicious use. This comes at the cost of not being in full control over how data is displayed and how long it stays available. This is all decided by the provider of the service. On the other hand, those services are often more convenient to use and provide interfaces to both create and receive content, making it easy for the creators and the students of iLabX to send and receive the announcements. One of these services is Twitter [7], an online service that lets users interact with each other by using so called *tweets*. Those tweets are small messages that can also contain videos and images and are publicly available. Similar to RSS feeds, tweets of a single user can be displayed as a single stream of tweets. Twitter therefore combines having a singe public source for distributing data without having high maintenance costs of the infrastructure and provides an easy to use interface for both students and creators. Another advantage of using Twitter over RSS feeds is that the information is displayed in a modern way without having to use additional software. In addition to that, the corresponding Twitter-feed can easily integrated into the VCC by integrating the page into the application. An example for this can be seen Figure 3.3. Therefore, Twitter is used to distribute news and announcements that are related to iLabX.

3.3 Caching information locally

There are two types of data that need to be stored in the vLab. The first type is persistent information that needs to saved in the file system of the vLab so other applications can access and process it. This information can be setup files or information about the labs of the MOOC. Those files rarely change over time but need to be kept up to date to ensure all students are working within the same environment. The other

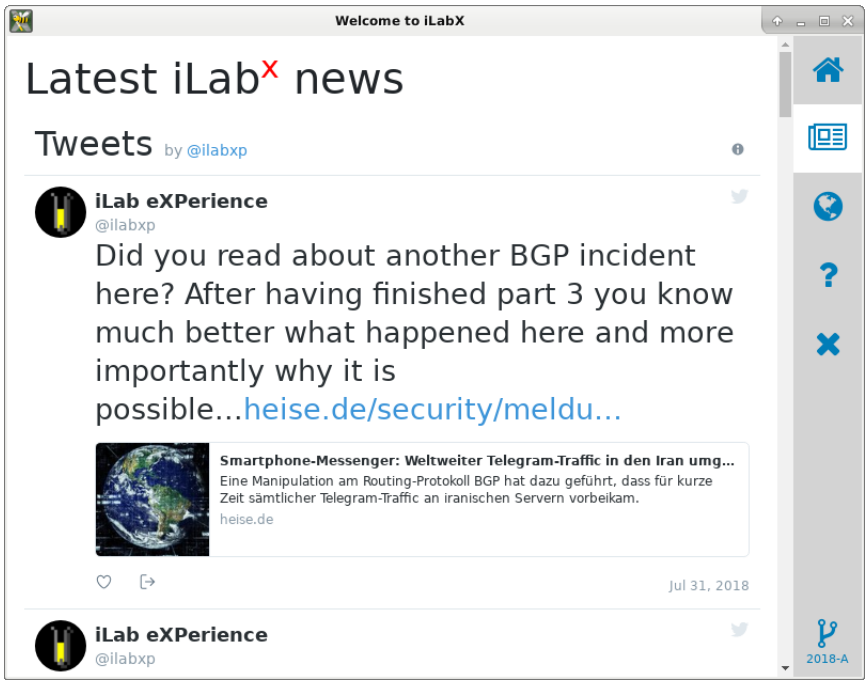

Figure 3.3: Twitter feed that is integrated into the VCC

type of information was described in Section 3.2 – News. News and announcements do not need to be stored persistently in the vLab, as they are not processed or accessed by other applications than the VCC itself. In contrary to the first type of data, news and announcements tend to be updated more often. Since both types of information have such different characteristics, it is reasonable to have two different mechanisms for storing the data locally and keeping it up to date. These two mechanisms are described in the following subsections.

3.3.1 Persistent information

The first type of information that should be stored in the vLab can also be considered to be persistent data. This is because the files containing the information need to be stored in the local file system such that they can be accessed by applications in order to be used. One example for these files are the CORE setup files that configure the environment for the different labs. Since these files do not change very often over time, it is reasonable to look at mechanisms that download the files to the local file systems and only update them when they are changed. The first mechanism that was considered was a self implemented version of a procedure which ensures that the latest version of a file is in the local system. After downloading the files onto the system, the mechanism needs

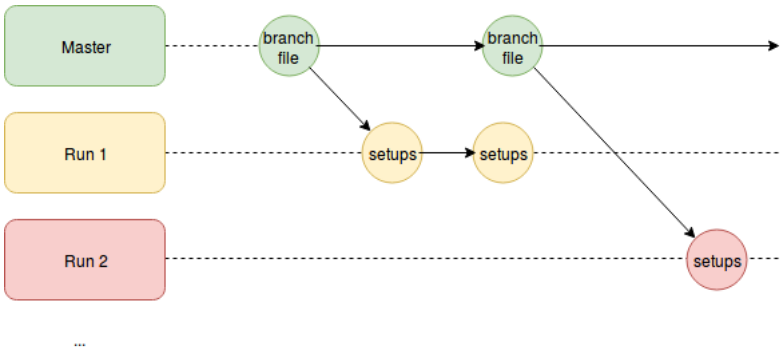

Figure 3.4: Structure of the Git repository used for distributing the setup files

to check regularly if the file at the source has been updated. In case of a modification, it modifies the file with the corresponding changes. In case nothing changed, nothing is done. Even though it is not hard to implement this functionality, there is already an established tool that does exactly this – Git. This well documented and commonly used tool already implements the needed functionality. Moreover it can be used by issuing shell commands, making its integration into the VCC straightforward. In addition to that it can easily be used in conjunction with GitHub [5], a service that provides the needed public backend for free. Therefore it makes sense to use Git over some self-hosted and self-implemented caching mechanism. The structure of the Git repository is described in the following. In the master branch there is single file that contains the names of available branches. Each branch represents a run in the MOOC and contains the content of the run. When starting the VCC for the first time, students need to choose the corresponding branch that represents the run they are in to continue. Each branch then contains the setup files for CORE and descriptions of the labs that are displayed in the VCC. Using this structure has the advantage that students from different runs can still access the setups they were using when they completed the course, even when some other run is currently active. This structure can also be seen in Figure 3.4 while Figure 3.5 shows the dialog that students are presented with in the VCC in order to select their branch. As mentioned before, each branch contains both CORE setup files and descriptions of each lab that is displayed in the VCC.

The description of each lab is delivered in a file which contents are in the JSON format that is then processed by the VCC to display the lab accordingly. The content of each JSON file defines how the corresponding lab is presented in the VCC. This gives the instructors of the course different possibilities to influence how the VCC looks like to the students and to unlock content at a later point in the course. In order to make this possible, the description of each lab has the same structure. Each JSON file has a *week*, *available*, *description* and *background* key that can be set individually. The

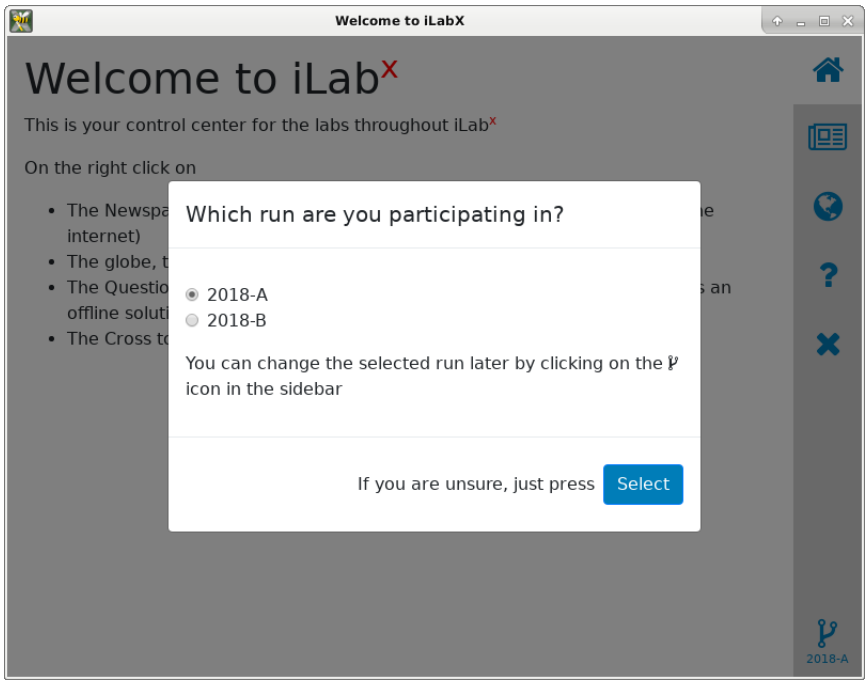

Figure 3.5: Branch selection

```
{
    "week": "1",
    "available": "True",
    "description": "Signal encoding and Ethernet",
    "background": "#51A8F9"
}
```

Figure 3.6: Example of a lab description

week key is used to associate the JSON to a specific part that is displayed in the VCC. In addition to that, instructors can indicate whether the lab should be available. In case the availability is set to true, the description is displayed in a field where the color matches the one set in the background key. In addition to that the VCC adds a button where students can start the corresponding lab with the click of the according button. In case the availability is set to *false*, this button is not added and the selection of the part shows a message that the lab is not available yet. An example of such a JSON description can be seen in Figure 3.6.

Another advantage of delivering content in this way instead of having it preinstalled in the vLab is that the content can be fully controlled by the organizers of iLabX. This means they can update setups throughout the course or make the content available at

some later point by adding and pushing it to the Git repository. With the proposed mechanism, students always have the correct version of the content to work with. The only problem that arises is when students modify the files. In this case a simple git pull is not an ideal way as this command tries to merge changes. To tackle this obstacle, all modifications that are done by students to the configuration files are overridden. In order to do so the following procedure is executed. In the beginning, the HEAD is reset to the latest local push that was received, thus deleting all modifications. After this is done, a normal pull can be performed to update the files to the correct version. One draw back of delivering the configurations over Git is that the VCC needs an Internet connection at the first start-up in order to download the files. This is a general problem that arises with every solution that relies on getting the latest content from the Internet instead of having it preinstalled in the image of the vLab. Therefore the approach to use Git to keep the configuration files in the virtual machine seems to be a reasonable solution.

3.3.2 Transient information

The other type of information that has to be cached locally are news and announcements so they can be displayed in the VCC. Since they change a lot more often than the setup files it is unnecessary to store this kind of information in Git. As described in Section 3.2, news and announcements are distributed via Twitter. Therefore, a mechanism is needed that caches the content from Twitter in the VCC and displays the latest available content to the students when accessed. This can be done in various ways. The two methods considered were (1) implementing an own mechanism that stores the files in the local file system or (2) using a network proxy called *service workers*. One advantage of having a self-implemented mechanism for exactly this purpose would be that it is specifically designed for this task. This way, the mechanism can be implemented as efficient and lightweight as possible. On the other hand, implementing such a mechanism requires more time than using a module that provides most of the needed functionality. In addition to that, an already existing module may not be as efficient as a mechanism that is specifically designed for this task, but it can be reused for similar tasks, making it available for all caching of this type that may be needed in the VCC. Therefore, JavaScript service workers [14][15] are used to provide this functionality in the VCC. Service workers are scripts that are running in a browser that can intercept and handle network request that are being made from the browser. They are therefore working as a network proxy to the VCC. In addition to that, service workers can provide the functionality to cache websites in order to provide offline functionality. In order to

achieve this, they run independently from the VCC in a different *lifecycle*. Until a service worker can be used, it needs to go through three steps of its lifecycle:

1. Registration
2. Installation
3. Activation

In the beginning, the service worker needs to be registered. Essentially this is simply telling Electron and the VCC where it can find the service worker which is serving and caching the content. Whenever the VCC is requesting the content from a website a service worker is registered for, that request is intercepted by the corresponding service worker. After a service worker has been registered, the next step is the installation. This means that the worker is initialized with all its functionality and cache. If the initialization is successful, the service worker is considered installed. If a service worker fails to be installed a new installation process is attempted at a later point (e.g. when the content is requested again). Finally the service worker can be activated. When being activated for the first time, the service worker loads all the content into the local cache. Only after that it can be used by the application. Therefore, the content can only be displayed when the news view is clicked for the second time. After the first activation is finished, the VCC can fetch data from the service worker. To make sure that always the latest content is delivered to VCC, the default implementation of service workers needed some changes such that the most recent content is displayed. In the standard procedure, content from the cache is displayed to enhance the experience, speed up that process and reduce loading times. The service worker then tries to fetch more recent content from the original source into the cache and update it accordingly. In the implementation that is used for the VCC, the service worker first tries to get the content from the source on the Internet and only serves the content from the cache when no Internet connection is available. This makes sure that the latest news and announcements that are currently available are displayed to the user. Another advantage of using service workers is that it provides a caching mechanism that can be used for almost any type of data that is supposed to be cached in the VCC. This means whenever there is the need to cache additional content within the application this can be done straightforward. All that must be done is to register a new service worker. So whenever the VCC tries to fetch data that is served by a service worker the request does not go to the source on the Internet directly, but to the service worker that is serving that content (1), as illustrated in Figure 3.7. In case there is an active Internet connection the service worker request (2.1) and fetch that content from the Internet source (3.1), store it in its cache (4) and deliver it to the VCC (4). In case there is no Internet (2.2), the service worker replies

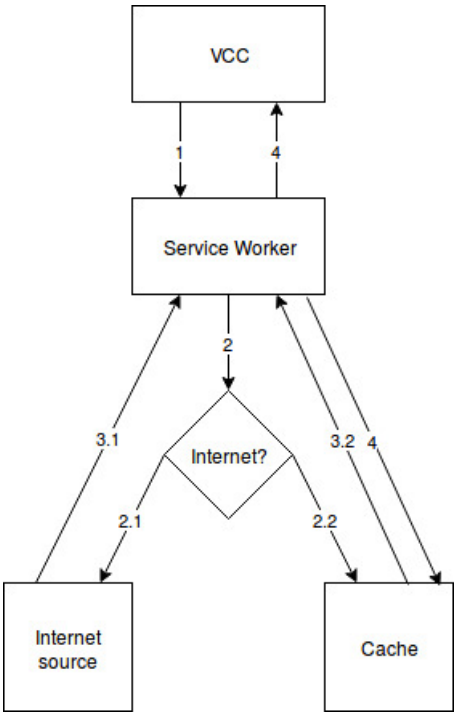

Figure 3.7: Procedure when the VCC requests data that is served by a Service Worker

with the latest content it has stored in its cache (3.2). Finally, this data can simply be integrated into the VCC as if it was fetched by a normal request to the Internet (4). The difference is only that in case there is no Internet connection, the latest content from the cache is displayed. Therefore the users of the VCC never experience a case where no news or announcements are displayed in the VCC.

3.4 Starting local applications

After all the latest data that is needed to solve the tasks from MOOC is loaded into the vLab, the next step is to make it more accessible. News and announcements are directly displayed in the VCC and also the descriptions of all labs are processed by the labs view. In order to make it easier for the users to make use of the CORE setup files, the process of setting up the environment for the labs should be simplified. One example for this is to have an interface that lets the students choose the week they want to work on and then start the CORE Emulator in the correct configuration with the click of a single button. This does not only make the process of setting up the environment easier, but also eliminates potential error sources during that process. Without this simplification, students would first need to start the CORE environment. Afterwards students need

to select the correct setup for the week from the file explorer. Finally, the loaded setup needs to be started before students can actually start working on the given tasks. To simplify this process, the VCC needs to be capable of starting applications that are installed in the vLab. Two different ways to achieve this were identified. The first one involves using the so called IPCMain process. This process makes it possible for the displayed content to communicate with the main process that is running the VCC. Using this, special commands and messages can be sent to the IPCMain process that then triggers certain behavior from the main process, like executing shell commands. This means that, in order to provide functionality, the source code needs to be changed at two different locations. The first is functionality that needs to be implemented such that it can be triggered. This functionality is added in the main process that runs the VCC. The second change needs to be made in the code of the website itself such that it can send the according messages to the main process. Instead of taking this rather complicated and bug prone approach it is also possible to use the functionality directly from the website itself, eliminating the additional step of using the IPCMain process. This makes it possible to put all the needed functionality into its own JavaScript files that can be imported in the views where the functionality is needed e.g. a Git module that handles all Git operations. This makes it also easier to maintain and extend the codebase in the future. Finally, using this approach it was possible to reduce the procedure described before into the click of a single button by starting CORE directly with the corresponding setup for the chosen lab.

3.5 Feedback channels

The last feature that is required to integrate the VCC into iLabX and to fulfill (R1) is the implementation of a feedback channel. This channel should give students the options to provide feedback to the instructors directly from within the vLab. There are two different types of feedback planned. The first one is textual feedback that can be given through the edX forums. To integrate this forum, a button has been added to the VCC from which a new window is opened. This window opens the edX forums page such that students can ask questions or report issues there. The second type of feedback is more complicated. In [3] a snapshot functionality has been added to the vLab. This is a feature that makes periodic snapshots of the running CORE session such that students can revert to that earlier stage of the vLab in case they do something wrong. This feature should also be used to report bugs of the vLab to the instructors. Therefore, next to a view where all the local snapshots are provided, the VCC needs to be capable of uploading selected files to the provided backend such that the instructors can analyze

and fix the issue. Since the snapshot functionality was not fully implemented at the time of finishing this project, the corresponding views have been implemented but not been integrated into the VCC. This needs to be adapted after the feature is fully functional to fully integrate this feature in the VCC.

3.6 Evaluation of the User Interface

One of the main goals of this IDP was to make the vLab more usable. Most of this improvements revolve around the VCC. In order to evaluate the design and usability of the user interface which has been implemented in the vLab Control Center the 10 heuristics established by Jakob Nielsen to evaluate usability are used. Even though these were introduced in 1995 [8], they are still used today to evaluate user interfaces of all kinds. The ten heuristics are:

1. Visibility of System Status
2. Match Between System and the Real World
3. User Control and Freedom
4. Consistency and Standards
5. Error Prevention
6. Recognition Rather than Recall
7. Flexibility and Efficiency of Use
8. Aesthetic and Minimalist Design
9. Help Users Recognize, Diagnose, and Recover from Errors
10. Help and Documentation

In the rest of this subsection, the meaning of these heuristics is explained. In addition to that, the VCC is evaluated on how well the user design has been implemented in the VCC or if it has violated some of these heuristics. Finally, a user survey has been conducted to confirm this evaluation which is discussed in the next section.

The **Visibility of System Status** heuristic is supposed to tell if the user is kept informed about what is going on in the system via feedback that is shown in a reasonable time. An example for this could be a specific symbol that is shown when data is processed. In the VCC there are not many processes that trigger actions that run in the background. The only process that is not directly visible to the user is the download

of data onto the system via Git. This process is made visible by an icon that indicates that the VCC is using Git to download the data. This icon appears while the data is being synchronized and disappears when no download is happening. All of the other processes that are started appear immediately in sight of the user, e.g. when a lab is started, CORE appears on top of the VCC. Overall it can be said that this heuristic is not violated at any point by the design of the VCC. The second heuristic in the list is the **Match Between System and the Real World**. This means that the system, in this case the VCC, should not use any advanced technical terms that are not known by the users but language and terminology that is easy to understand. In addition to that the order in that items appear should be logical and follow common conventions. The VCC therefore needs to be analyzed with regard to those two aspects. The language used throughout the application is the same language that is used within the MOOC. All buttons and descriptions are labelled accordingly and there is no additional technical language that is introduced in the VCC that is not used in the course. In addition to that the icons that are used in the navigation on the right side were chosen such that they represent symbols that are used for similar purposes. The only icon that can be argued about would be the earth symbol that is used to navigate to the lab view. A better symbol to represent the Lab could be a test tube that represents this view. On the other hand, it can also be argued that the globe that is representing the Internet which is the subject of the whole course. The problem with introducing the test tube as new icon would be, that it has to be designed in a way to fit it into the design of the other font awesome icons that are used. To make things easier, the globe, which was already available is used to represent this view. The order of the icons that are located in the side bar are the following: home, news, labs, help. This is a common order of these categories. Another option to order the icons would be to put the lab before the news as it is used more often. This order would be different from the order that other applications use. The next heuristic is called **User Control and Freedom**. This refers to the fact that users could unintentionally use functionality and therefore need an "emergency exit" from a state they can not continue further from. In the VCC every action can be undone by a simple click on the icon of the view. In addition to that, every new window that opens up can be closed without affecting some other part of the system. Therefore the emergency is implicitly given to the user, even though it may not be clearly marked. Since this functionality is included implicitly it may not be obvious to the users and they may not be aware of this. Finally, there is no explicit button for an "emergency exit" since the interface is designed in such a way that users can reach all other functionality even when they may have accessed some functionality by accident before. Therefore, the heuristic is partially fulfilled given that there is an implicit emergency exit at all times that may not be marked clearly enough at some

times. The **Consistency and Standards** heuristic is an indicator if an application follows conventions that are set by the platform. This includes the meaning of symbols and the placement of these within the application. The VCC uses the same windows that are displayed for all applications that are running in the vLab. This makes it consistent with other windows that are displayed within the system. In addition to that the VCC has its options on the right hand side of the screen similar to the task bar on the right hand side of the vLab. The icons that are used are, as mentioned before, consistent with the icons and symbols that are used by other applications for similar functionality. Therefore the VCC does not violate this heuristic. Very closely related to this is also the **Recognition Rather than Recall** heuristic. It means a user should not be overwhelmed with things that he has to remember but that he can recognize what to do in order to use certain functionality. This heuristic is complied with in the VCC as the emphasis for navigating and using functionality is on the use of icons. Therefore the only thing a user has to remember is behind which icon he can find the given functionality. This is to make it easier for the user to use the VCC through recognition instead of having to remember what to find where since the symbols are already an indicator for the functionality. In some cases the user may encounter errors which is measured **Error Prevention** heuristic. It says that the design should be in such a way that errors are prevented before they occur. In addition to that actions that can cause errors need to be confirmed by the user before they are executed. In the VCC, the only action that is error-prone is pulling data from the Internet. Even though user has no influence on whether this action executed or not, she needs to be informed that the latest update has failed and the system may not be using the latest data. In order to achieve this, the user gets a warning in case the operation has failed. Another heuristic to look at is the **Flexibility and Efficiency of Use**, which is about the integration of accelerators for expert users to speed up the usage and the possibility for users to tailor the work flow for their needs. This heuristic is violated by the VCC as no such possibilities exist and may need to be added in the future. One possibility may be to choose a different view than the landing page as the default view, e.g. the lab view such that the lab environment can be started even faster. Since the interface of the VCC is rather simple there are not many options for speed up the interaction as it is limited to four views where all features can be accessed from directly. This simple design also plays into the **Aesthetic and Minimalist Design** heuristic which says that the information that is displayed should be limited to the relevant bits instead of overflowing the user with too much input to process. The text on the buttons and in the views is reduced to the needed amount of information. The lab view gives a quick idea of what the lab is about and descriptions in other views are as concise as possible. At the moment, users are not supported in the **Recognition, Diagnosis and Recovery**

Heuristic	Violated
Visibility of System Status	no
Match Between System and the Real World	no
User Control and Freedom	partially
Consistency and Standards	no
Error Prevention	partially
Recognition Rather than Recall	no
Flexibility and Efficiency of Use	yes
Aesthetic and Minimalist Design	no
Help Users Recognize, Diagnose, and Recover from Errors	yes
Help and Documentation	no

Table 3.2: Overview of which Nielsen's usability heuristics are violated by the VCC

from Errors that might occur which is also a violation of this heuristic. Even though this is a problem, this heuristic is according to [9] not one of the most important ones to measure a good usability. Finally, the **Help and Documentation** of the VCC are located on edX where users can find answers in a dedicated section to this as well as a dedicated section for questions on the edX forum. This has the advantage that users do not need to remember different places for where they can find answers and solutions to problems they might encounter. Table 3.2 gives a quick overview of which heuristics are violated by the VCC.

3.7 User survey

In order to confirm the evaluation of the previous subsection with users, a small user survey with 13 students was conducted after they used the VCC (and the vLab overall) as part of a test run of the iLabX course. The survey was composed of 10 questions aiming at the heuristics discussed above, the questions itself can be found in the Appendix. All heuristics presented in Section 3.6 but *Flexibility and Efficiency of Use* were covered by the survey as it is violated completely by the VCC. The results of the survey are discussed in the following. Like the heuristics, the questions can be divided into four different categories [6]. The first category describes if the interface meets the users expectations. It includes the *Match the real world*, *Consistency and Standards* and the *Help and Documentation* heuristics. 92.3% of the participants felt that the VCC was only using language that was also used throughout the MOOC. In contrary to that only on participant of the course thought that the VCC consisted mostly of terms that were never used before. This implicates that the VCC uses language that is known to the users and does not introduce new terminology. In addition to that 46% of the users

said that the VCC is very similar to other applications they use and 31% indicated that the VCC has still some similarities. The remaining 23% stated that the VCC has few similarities with applications they use. Therefore it can be concluded that the VCC follows common standards that are also used throughout other applications. Finally, 77% of the participants of the survey stated that they knew where to find the needed documentation to solve problems they encountered. The other 23% reported that they did not know where to find information that can help solve their problem. No student indicated that she did not know where to find the documentation at all. This gives insight that students were aware of where to find information but does not give insight into the quality of the documentation that exists. As mentioned before, the amount of existing documentation is not sufficient at the moment, making it necessary to extend it in the future.

The next category is about how much the user is kept informed and how much he can influence the actions that are executed by the VCC. The heuristics that are in this category are *User Control and Freedom*, *Visibility of System Status* and *Flexibility and Efficiency of Use*. For the *User Control and Freedom*, users were asked if they were aware that they could always or most of the time redo actions if they did something wrong. 12 out of the 13 students reported they could always redo things in case they clicked something wrong with one student reporting that this options was totally missing. It therefore can be said that to most users it seemed to be clear how to get out of a stale state even when the explicit "emergency exit" was missing. In order to improve on this, an explicit "back" button could be added in the future to the VCC. Using the heuristic *Visibility of System Status*, 53.8% of the students stated they were aware of what actions of the VCC were executed all of the time while 38.5% indicated they were aware of ongoing processes most of the time. Only 7.7% of the students stated they were not aware of the processes that were going on most of the time while no user was not aware of anything that was executed by the VCC. As mentioned before, there was no question regarding to *Flexibility and Efficiency of Use* since there are no options to customize the user experience within the VCC in the current version.

The third category revolves around how error handling of the VCC. The three heuristics that can be put into this domain: *Error Prevention*, *Recognition Rather than Recall*, and *Help Users Recognize, Diagnose, and Recover from Errors*. This topic is covered by two questions. The first one asked if the amount of information after an error occurred was sufficient. This was confirmed by 6 out of the 13 students. Two students said they were presented with too much information while the remaining 5 students indicated that they were missing crucial information to understand what the nature of the error was. This is supported by the second question in this field that asked if students were

told what went wrong in case of an error. Only 38.5% of the users knew what caused the error and another 46.2% were partially aware of what happened. This indicates, that the VCC's capabilities of reporting errors can be improved in the future which was also found by the evaluation done in Section 3.6.

The final domain is about how simple the design of the VCC is which is represented by the *Aesthetic and Minimalist Design* heuristic. In order to gain more insight over how satisfied the users were with the design of the interface they were asked two questions. The first one was about how well they could get used to the VCC in order to be able to work with it. 76.9% of the students stated that it was very easy to get used to the VCC while the other 23.1% said it was fairly easy to use the VCC. No student said it was hard to interact with the system. Finally, students were also satisfied with the amount of information that was displayed within the VCC with 11 out of the 13 students stating it contained the perfect amount of information and two of them indicating they were presented with too much information. Overall the survey reflected most of the points that were mentioned before and are listed in Table 3.2. The only difference is that users were more satisfied with the documentation and with regards to the *User Control and Freedom* heuristics. Finally, it can be said that the participants were satisfied with the interface of the VCC. This concludes how the vLab Control Center was implemented as part of this project.

Chapter 4

Hardening the vLab

A minor, but also important part of the project was the hardening of the vLab with regards to stability in order to fulfill (R2) described in Chapter 2. This is required to make it possible to work through tasks that are part of iLabX. Since the vLab is tested as part of [3], the main focus lies on making it possible to do the tasks given in iLabX. Most of the errors encountered in this aspect was with regards to the CORE emulator. The first major issue that occurred was that it was not possible to open a shell on hosts within a running CORE setup when using SSH to connect to a container. This is necessary as students might want to connect from one container to an other container directly via SSH and not via the vLab as host system. This problem was caused by the *devpts* virtual file system that was not mounted in the CORE containers. This file system is needed to spawn "pseudo terminals" (also short *pty*) that enable a Linux environment to open terminals that are not directly connected with hardware like a keyboard or mouse to provide input. To be able to open a shell via SSH on the CORE containers it was necessary to add a corresponding entry in the file system in the *fstab* file that then needs to be mounted by every container in the CORE setup in order to enable SSH on them with the command *mount -a*. This error occurred since each CORE container does not use a so called initialization system like *systemd* in order to initialize such file systems to make such operations possible. Therefore devpts had to be mounted manually.

Another critical point that needed to be fixed before the MOOC could start was the possibility to run Firefox reliably via X forwarding. Firefox crashed regularly when opened with the *ssh -X <host> "firefox"* command. The reason for that is related to the Xorg display server as there are issues with the connection to the correct *DISPLAY* environment that is used within the vLab. Most of these crashes could be prevented

by explicitly setting the DISPLAY variable in the command resulting in the following: *ssh -X <host> "env DISPLAY=:0 firefox"*. This spawns a new Firefox session on that host that is be displayed in the vLab. The next problem that occurs is that if there is a second instance of Firefox that is already running in the vLab, both sessions are connected. This behavior is not intended as it has some side effects on the IP addresses that are used within CORE when websites that are located on other containers are accessed. In order to prevent this it is possible to spawn an isolated Firefox session one has to use the "no-remote" argument. Therefore, the resulting command is *ssh -X <host> "env DISPLAY=:0 firefox -no-remote"*. On a few occasions it can still happen that Firefox crashes on the first start up but restarting the browser solves this problem. To be able to correctly render websites that are hosted on other CORE containers, some Firefox-settings needs to be pre-configured. This can be done by placing corresponding files in the vLab file system when building the image. The corresponding file is located in the "/usr/lib/firefox-esr" where the multiprocesses need to be disabled in order to display the websites correctly.

In addition to these special configurations some packages were removed in order to make the resulting image of the vLab smaller. This included the removal of a second browser and other software such as the Libre Office Suite that is not needed throughout iLabX. On the other hand self-implemented software was added to the system. These additions include the VCC and some custom scripts that are capable of generating specially crafted packets that can be sent to other hosts. Moreover, we composed the corresponding man pages for these self-implemented programs and added them to the man database. Finally, in order to make the setup file for the final part of the MOOC smaller, the contents for web server were added into the vLab.

Overall it can be said that the environment by [16] was already very stable and there were not many bugs encountered that had to be fixed as part of this project.

Chapter 5

Tutorials

Since the vLab is used as part of a Massive Open Online Course it is also used by an audience that is not familiar with the system. In order to make all features of the vLab accessible they need to be documented. The documentation that is provided to the users of the vLab is currently hosted on edX. This is reasonable, since most users that are using the vLab are also participants of the course iLabX. In order to make the information more accessible to those users, it therefore does make sense to integrate the documentation of the vLab into the platform of the MOOC itself instead of providing another source to host the content. The documentation contains both information about how to set up the vLab as well as how to use the functionality delivered by the vLab. The first section of the documentation is about how to install the environment with VirtualBox. This is then followed by a short introduction about what functionality is provided by the environment. In order to integrate the MOOC into the vLab, the VCC has been added to the system. Its features are explained in the next section of edX. It contains brief information about how to choose the correct branch and how each view of the vLab works. The last two sections are about CORE and the Linux environment itself. The first section about CORE contains information how to access the containers in a running setup and how to change the appearance of the setup by displaying interfaces or IP addresses. The section about Linux contains a brief tutorial on how to get started in a Linux environment.

Overall, it can be said that the documentation needs to be extended in the future. At the moment it only contains information about how to properly utilize features provided by the environment but not how to resolve errors that may occur. In addition to that, the documentation should be extended by some tutorial videos to make the system even more accessible to new users. Moreover, some reference to the documentation of the

Debian distribution should be added since the vLab-OS is based on this distribution. Therefore most errors that occur with regards to the Linux environment can be solved with the extensive documentation that is provided for Debian.

Chapter 6

Conclusion and Future Work

In the project that is has been realized in this IDP, the virtual iLab isle has been extended and improved such that it can be used in the newly developed Massive Open Online Course "iLabX". In order to realize this, the requirements that were described in Section 2.2 needed to be fulfilled. To integrate the MOOC into the vLab, the so called vLab Control Center has been developed and integrated into the existing image of the virtual machine. The VCC downloads and keeps the latest data that is needed up to date when it is connected to the Internet. In addition to that, this data is also kept and cached accordingly in the local system to make it available to the users even when there is no connection to the Internet. The downloaded data is also integrated into the VCC making it available to the users within a single application. In addition to that, the procedure that was required to set up the CORE environment was eliminated. Instead, the correct environment can be set up with the click of a single button in the VCC. The VCC adds further channels that enable students to provide feedback to the instructors from iLabX making it possible to report bugs or suggest improvements for the system from within the vLab itself.

The image of the vLab itself has decreased in size as unneeded software was removed. Besides that other software has also been added to make it possible for students of the course to solve the given tasks without having to install additional software. In addition to that, the vLab and installed programs have been customized even more to fit the needs of its users.

The current version of the vLab can be used to successfully run the course iLabX as it has been shown by a test run that took place at the TUM in summer 2018. Students of the course were satisfied with the given environment and only known issues occurred

during the run. In the future, these issues need to be fixed. Another shortcoming is the documentation which only consists of tutorials of how to use features of the vLab, but does not contain solutions to problems that may be encountered by the user.

Chapter 7

Appendix

Questions of the conducted user survey.

I always knew what was going on in the vLab and VCC. I never felt the system was doing things I was not aware of.

	1	2	3	4	
I agree	○	○	○	○	I disagree

The system was using language I could easily understand

	1	2	3	4	
The same terminology that was also used in iLabX was used.	○	○	○	○	No there were terms that I have never seen before

I could always redo things when I did something wrong (e.g. clicked on the wrong part in the VCC)

	1	2	3	4	
There was always a way to redo things	○	○	○	○	There was no way to redo things

The Layout of the vLab and VCC are similar to other systems and applications that I use

	1	2	3	4	
Very similar	○	○	○	○	Not similar at all

When an error occurred, the error message I got was helpful

	1	2	3	4	5	
No information given	○	○	○	○	○	Too much information given

I was able to get used to the vLab and VCC very easily

	1	2	3	4	
I agree	○	○	○	○	I disagree

The VCC did contain information that was irrelevant to iLabX

	1	2	3	4	5	
It was missing relevant information	○	○	○	○	○	It contained too much information

Whenever an error occurred, I was told what went wrong

	1	2	3	4	
I agree	○	○	○	○	I disagree

I knew where to look when I needed help, e.g. the documentation

	1	2	3	4	
I agree	○	○	○	○	I disagree

Do you have suggestions on how to improve the vLab and VCC?

Long answer text

Bibliography

[1] "Comparison of CORE Network Emulation Platforms". In: *Proceedings of IEEE MILCOM Conference* (2010), pp. 864–869.

[2] *Core Open Research Emulator (CORE)*. `https://www.nrl.navy.mil/itd/ncs/products/core`. Accessed: 2018-08-17.

[3] Yaroslav Dushko. "Testing of Virtual iLab Isle". Masters Thesis. Technische Universität München, 2018.

[4] *Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS*. `http://electronjs.org/`. Accessed: 2018-08-18.

[5] *GitHub*. `https://github.com/`. Accessed: 2018-08-22.

[6] Stephan Krusche. *Lecture notes in Project Organization and Management*. 2017.

[7] Haewoon Kwak et al. "What is Twitter, a social network or a news media?" In: *Proceedings of the 19th international conference on World wide web*. AcM. 2010, pp. 591–600.

[8] Jakob Nielsen. "10 usability heuristics for user interface design". In: *Nielsen Norman Group* 1.1 (1995).

[9] Jakob Nielsen. "Enhancing the explanatory power of usability heuristics". In: *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM. 1994, pp. 152–158.

[10] *Online Master of Science in Computer Science*. `http://www.omscs.gatech.edu/`. Accessed: 2018-08-22.

[11] *Python*. `https://www.python.org/`. Accessed: 2018-08-18.

[12] *Python in Debian*. `https://wiki.debian.org/Python`. Accessed: 2018-08-18.

[13] *RSS 2.0 Specification*. `http://cyber.harvard.edu/rss/rss.html`. Accessed: 2018-08-19.

[14] *Service Workers: an Introduction*. `https://developers.google.com/web/fundamentals/primers/service-workers/`. Accessed: 2018-08-14.

[15] *ServiceWorker Cookbook*. `https://serviceworke.rs/`. Accessed: 2018-08-14.

[16] Moritz Sichert. “Virtual iLab Isle”. Interdisciplinary Project. Technische Universität München, 2017.

[17] *StackOverflow Developer Survey 2018*. `https://insights.stackoverflow.com/survey/2018/#most-popular-technologies`. Accessed: 2018-08-18.

[18] *VirtualBox End-user documentation*. `https://www.virtualbox.org/wiki/End-user_documentation`. Accessed: 2018-08-22.

[19] Jochen Wulf et al. “Massive open online courses”. In: *Business & Information Systems Engineering* 6.2 (2014), pp. 111–114.