Server System Management - UCDP

Setting Up a Kiosk to Display a Custom Site on a Computer

Groep 19
Emiel DEJONGHE, Jasper DESNYDER, Kobbe VEREECKEN, Michiel ANTHIERENS

Abstract

This paper provides a comprehensive guide on how to set up a kiosk on a computer and display a custom site on it, hosted on a web server. The paper focuses on use case driven scenarios, where the kiosk is used in public places, such as museums, libraries, retail stores, or fast food restaurants, to showcase specific information, products, or services.

The paper begins by discussing the background and importance of setting up a kiosk and identifies specific challenges that organizations may face when setting up a kiosk. The paper provides a step-by-step guide to setting up a kiosk on a computer, including configuring the browser settings and setting up the kiosk mode. The paper also explains how to host a custom site on a web server and provides instructions for configuring the kiosk to display the site.

The results of this paper show that setting up a kiosk on a computer and displaying a custom site on it can help organizations engage with their customers, provide a personalized experience, and increase sales or awareness. The paper concludes by summarizing the key points covered in the paper and reiterating the benefits of setting up a kiosk.

Overall, this paper provides organizations with the knowledge and tools necessary to set up a kiosk and display a custom site on it. By following the instructions and guidelines provided in this paper, organizations can create engaging and personalized experiences for their customers, improve their brand awareness, and increase sales or awareness.

1. Introduction

With the ever increasing usages of technology, more and more businesses or shops start implementing a dedicated screen for its customers to use and perform specific actions. A perfect example of this would be the self-order screens at McDonalds or any other fast food restaurant. Image 1 shows a good example of what a kiosk could look like, and how it can be used.

Image 1 - Self-order screens at McDonalds

These screens, also known as kiosks, are designed to allow customers to place orders easily without the need for assistance or human interaction. However, customers are limited to performing actions within the application used for ordering and cannot access other applications or the desktop.

In this paper, we will explore how you can easily set up a kiosk like this yourself, combined with the process of setting up a web server that will host the website you want to display in this kiosk.

2. Proposed solution

Setting up a kiosk can be done in a few different ways, but this paper will only cover two:

- Locally in settings
- Using PowerShell

We chose to only do these 2 options, because they're the most common and most convenient to implement.

Setting up a webserver can be done in a few different ways, but this paper will only cover one: Apache.
We chose to only do this option, because it's one of the most common, and it's most convenient to implement.

We'll use Linux Ubuntu to set up the webserver. We'll install our website on this webserver, after which our kiosk system can connect to this website.

3. Setting up the kiosk

To set up a kiosk in Windows, your Windows version needs to be one of the following:

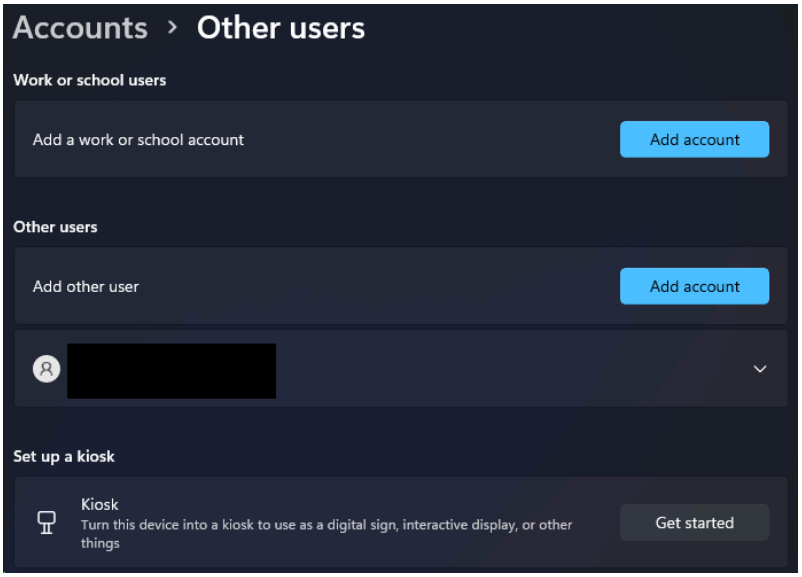
- Windows 10 Professional
- Windows 10 Enterprise
- Windows 10 Education
- Windows 11

Please note: According to Microsoft, all Windows 11 versions should work. However, while testing out different versions, we found that Windows 11 Home doesn't allow setting up a kiosk. To make sure you can set up a kiosk in Windows 11, try to use Windows 11 Professional, Enterprise or Education.

3.1 In local settings

Setting up a kiosk using the settings app should be relatively easy.
(The screenshots all show settings in Windows 11, but the process is the same in Windows 10).

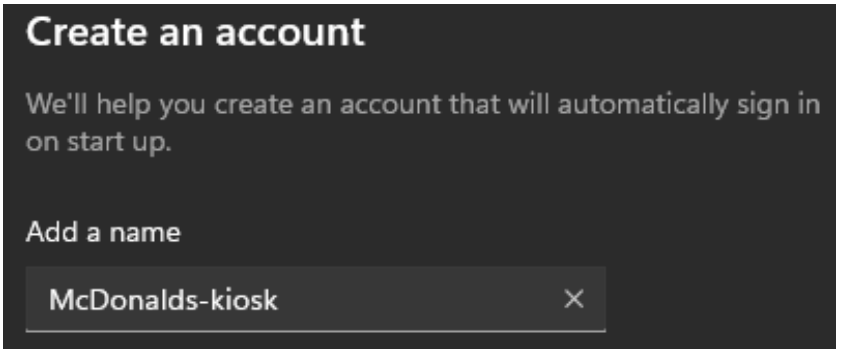
Navigate to the Settings, then proceed to the Accounts section and find the option for Other users.



Within that section, locate the field labeled "Set up a kiosk."

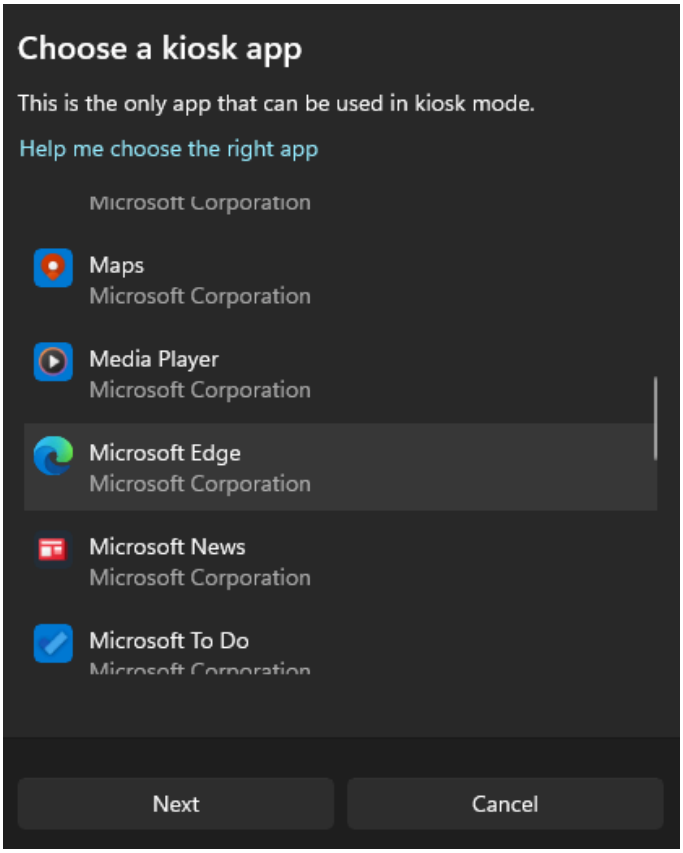
Click on the Get started button to initiate the kiosk setup process.

You will be prompted to assign a name to the kiosk account. Choose a suitable name for the application, such as "McDonalds-kiosk."



Once a name has been selected, click Next.

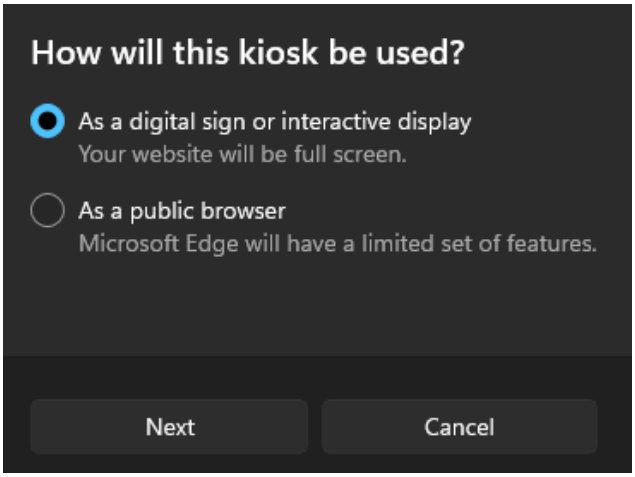
Choose a kiosk app from the options provided. This will determine which application opens when the kiosk user signs in. For example, select Microsoft Edge as the browser application.



Next, there will be two options to specify how the kiosk will be used.

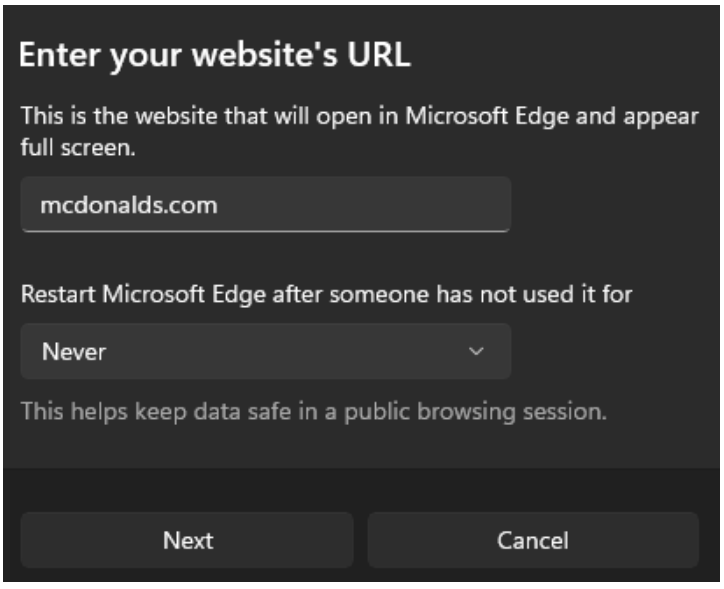
- As a digital sign-in or interactive display
- As a public browser

Choose the option "As a digital sign-in or interactive display" to ensure that users cannot navigate to different websites on the kiosk.



Proceed to the next step and fill in the desired URL that the browser should display. Initially, any URL can be entered, such as "mcdonalds.com." Later, this URL can be modified to match the web server's address.

Additionally, make sure to adjust the setting that determines whether Microsoft Edge restarts after a period of inactivity. Change the option from "5 minutes" to "never" to prevent the browser from reloading.



The kiosk setup is now complete. Sign in using the created kiosk user account to see the results!

To exit the kiosk mode, use the key combination CTRL + ALT + DEL to sign out.

3.2 Using PowerShell

To set up a kiosk in PowerShell, the initial step involves creating a new local user, which will be utilized for signing in as the kiosk.
Execute the following command as an administrator in PowerShell to create a new user:

```
New-LocalUser -Name <Username> -NoPassword
```

Ensure to replace the placeholder "<Username>" with your preferred values.
A functional example would be:

```
New-LocalUser -Name kiosk -NoPassword
```

Alternatively, if you encounter difficulties with the previous method, you can create a new user directly in the local settings. Follow these steps:

1. Go to Settings and select Accounts.
2. Navigate to Other users and click on Add account.
3. You can skip using a Microsoft account by choosing the option "I don't have this person's sign-in information" and proceed to Add a user without a Microsoft account.
4. Provide a username for the new user and leave the password fields blank.

Note: Before proceeding to the next steps, ensure that you have signed in to the new account at least once.

There are multiple methods available for configuring the kiosk settings in Powershell.

1. By AppUserModelID and user name

```
Set-AssignedAccess -AppUserModelID <AUMID> -UserName <username>
```

2. By AppUserModelID and user SID

```
Set-AssignedAccess -AppUserModelID <AUMID> -UserSID <user sid>
```

3. By app name and user name

```
Set-AssignedAccess -AppName <CustomApp> -UserName <username>
```

4. By app name and user SID

```
Set-AssignedAccess -AppName <CustomApp> -UserSID <user sid>
```

Get AppUserModelID

```
Get-StartApps | where name -eq "<app name>"
# To get the AUMID of Microsoft Edge
Get-StartApps | where name -like "Microsoft Edge"
```

Get user SID

```
Get-LocalUser -Name <name> | select sid
```

Get a list of all usernames

```
Get-LocalUser | select name
```

4. Setting up the Linux server

The server is on Digitalocean. You start by creating an account through the GitHub student pack. This gives you 200$ credits to play around with. After the account set-up, create a team and connect a credit card or your Paypal. This is required by Digitalocean to continue. After that you can start setting up a server.

4.1 DigitalOcean create a server (droplet)

Now, go to the homepage of Digitalocean and create a new project. I called it “Kiosk”. After that you go to the “manage” tab. This is on the left of the home page. You choose “Droplets”. Then you see the droplets screen and you click “create Droplet”. We chose the following configurations:

- Region: Frankfurt
- Datacenter: Frankfurt
- VPC Network: Default
- OS: Ubuntu
    - Version 22.04 (most recent LTS)
- Droplet Type: Basic
- CPU: Regular
    - Disk Type: SSD
- Price: 4$
    - 512MB / 1 CPU
    - 10GB SSD
    - 500GB Transfer
- Authentication Method
    - Password
        - “Type your password”
- Options
    - Add improved metrics monitoring and alerting (free)
- Hostname
    - Kiosk-Server
- Project
    - Kiosk

Now we see “$4.00/month” and “create droplet” at the bottom of the page. We can click “create droplet”. Now we have our DigitalOcean Linux server.

4.2 Connecting to the server

Now you can go to your project and you will see that the server is getting created. Once created, click on your droplet and it will open a details page of it. On the details page you search for the ipv4 address (top left). Copy it. Go to your own laptop or computer terminal and SSH towards it. In our case.
ssh root@OurIPV4AddressHere

Then the terminal will ask for a password. This is the password you created in the authentication method. After you filled in the password (and it's correct). You will then see that you are connected to the server through SSH. You are now on your server.

5. Setting up the web server

Apache is one of the most used and respected web servers out there, having a share of more than 20% of all web servers used.

5.1 Preparing the environment

To begin setting everything up on a system, firstly make sure everything is up to date by running the commands shown below:

```
sudo apt update && sudo apt upgrade
```

It's really important that everything is up-to-date when installing new software on a system, because it helps ensure compatibility with other software components and operating systems.

When all updates are finished, everything is ready to install Apache2!

```
sudo apt install apache2
```

Right after installing, the Apache2 web server will be active, and a default page should appear when surfing to the ip of the machine the webserver is running on. Image 2 shows a part of this default page.

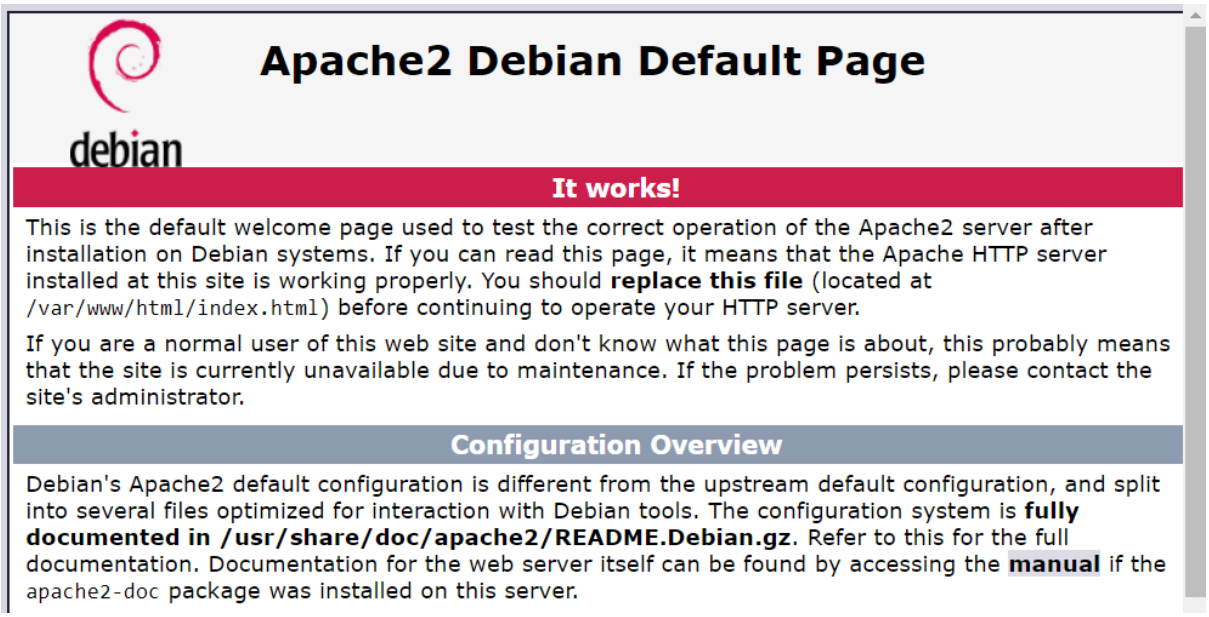


Image 2 - Default page from Apache2

The ip of the machine can be found using:

```
ip a
```

5.2 Troubleshooting

If this doesn't happen or doesn't work, it may be possible that there's another web server already running on the machine. When this happens, both web servers will try to access port 80 (which is the default port for http).

This can be checked using the following command:

```
systemctl status apache2
```

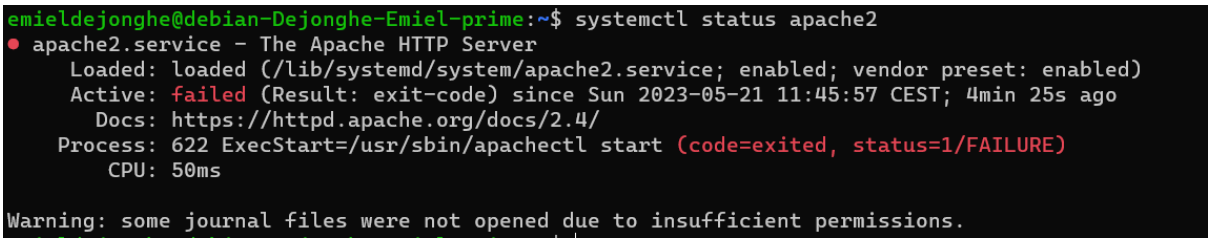


Image 3 - Output when Apache2 failed to start

Use the following command to check which process is using the port 80:

```
sudo lsof -i :80
```

and then kill the process using:

```
sudo kill [PID]
```

5.3 Configuring the custom site

Now, to run a custom page, disable the default site and enable the custom site.

First, add the html, css and javascript to the machine.
Do this by adding a directory into the **/var/www/** directory, which can be done using the following command:

```
mkdir /var/www/example.com
```

After this, create a configuration file for the new site. This is done in the **/etc/apache2/sites-available** directory. It's easier to copy the 000-default.conf file and edit it for the custom site, than creating one from scratch.

```
sudo cp 000-default.conf example.com.conf
```

It's important to change the **DocumentRoot** in this file to the correct location using a text editor (for example nano), in this case **/var/www/example.com**

We now need to disable the default site, and enable the custom site.
This is done as following:

```
sudo a2dissite 000-default.conf
```

```
sudo a2ensite example.com.conf
```

Reload the Apache2 service after this:

```
sudo systemctl reload apache2
```

The new site is now up and running!

5.4 Securing the web server

When creating a web server, it's very important to not forget about security.
It's important to turn off the web server's signature, which provides information about the version. This kind of information doesn't need to be accessible to everyone.

It's possible to turn this off by editing the **/etc/apache2/apache2.conf** config file using:

```
sudo nano /etc/apache2/apache2.conf
```

In this file, change the following line from **On** to **Off (**add this line to the file if it isn't present):

```
ServerSignature Off
```

Once again restart the Apache2 web server using:

```
sudo systemctl restart apache2
```

6. Setting up the web server with Ansible

Another way of setting up an Apache web server is with Ansible. More exactly, Ansible playbooks. Why Ansible? You can create multiple playbooks in multiple scenarios and execute them with one command. A great example is what we will be doing today. Setting up a playbook for installing and setting up Apache on the Linux server. If your server ever gets hacked and wiped, you can set up the whole environment up again with your playbooks in no time. You can use Ansible on Windows by installing and using WSL in Windows.

6.1 Ansible host file

In your WSL environment:

```
sudo nano /etc/ansible/hosts
```

Inside the the file add:
[NameForYourHostHere]
root@YourServerIp

Save the file and exit.

6.2 Ansible connection setup

We are now going to set up the SSH connection between your WSL and your Linux server.

Execute the following command in your WSL environment:

```
ssh-copy-id root@YourServerIp
```

Test the ssh connection with:

```
ssh root@YourServerIp
```

If no password is asked then the SSH connection with your keys works.

Now we can test the Ansible connection:

```
ansible all -m ping
```

If the command outputs a green message with “SUCCESS” then the Ansible connection is ready to go.

6.3 Ansible playbooks

Ansible playbooks are configuration files written in YAML format that define the desired state of a system and the steps to be executed to achieve that state. A playbook runs in order from top to bottom. Within each playbook, tasks also run in order from top to bottom.Each playbook starts with “--- “.

In your WSL, create a folder “ansibleBook”. In that folder make a file called “Configuration.yml”, after that create a folder named tasks and another one named files.

```
kobbe@ASUS-ROG-KV:~/ansibleBook$ ls
Configuration.yml  files  tasks
```

Image 4 - Contents of ansibleBook directory

6.3.1 The configuration.yml file

Inside the Configuration.yml, create something as shown in image 5.

```yaml
---
# Run with: ansible-playbook Configuration.yml -K

- name: Installation and configuration of webserver
  hosts: Kiosk
  become: true

  tasks:
    - include_tasks: tasks/install.yml
    - include_tasks: tasks/config.yml

  handlers:
    - name: Reload Apache
      service:
        name: apache2
        state: reloaded

    - name: Restart Apache
      service:
        name: apache2
        state: restarted
```

Image 5 - Content of the configuration.yml file

The file explained
“Name:” Name of the Playbook
“hosts:” Remember Ansible hosts? The “[NameForYourHostHere]”, that exact name you want to place here without “[]”, only the name.
“become: true” This makes to Playbook execute as root (is needed in this case)

The tasks section, this lists the tasks to be executed:
“tasks/install.yml” Installs the needed software (We will create this file later)
“tasks/config.yml” Configures Apache (We will create this file later)

The handlers section, handlers are tasks that are only triggered when explicitly notified.They are triggered by other tasks using the notify directive.
In this case we have “Reload Apache” and “Restart Apache”, which like the name suggests, when notified will restart or reload Apache.

Once the above is done, open the “tasks” folder and make the files “install.yml” and “config.yml”.

6.3.2 The install.yml file

There are 2 tasks, “Update all packages to their latest version” and “Install Apache”.

The first task
You use apt with name “*” and state: latest. This means that the apt module is set to "*", to indicate that all packages should be updated and the state parameter is set to "latest" to ensure that all packages are updated to their latest available version.

The second task
You use, apt: "name=apache2 update_cache=yes state=latest". This means that the apt module is set to "apache2" to specify that Apache should be installed. The update_cache parameter is set to "yes" to update the package cache before installing Apache. The state parameter is set to "latest" to ensure that Apache is installed with the latest available version.

The file should look something like image 6 shows.

```
---
# First update all packeges then install Apache
# Documentation:

- name: Update all packages to their latest version
  apt:
    name: "*"
    state: latest

- name: Install Apache
  apt: "name=apache2 update_cache=yes state=latest"
```

Image 6 - Content of the install.yml file

6.3.3 The config.yml file

Now after that, open “config.yml” and add the content that image 7 shows to this file.

```
---
# Configure Apache
- name: Set up HTTP virtualHost
  template:
    src: "files/example.com.conf.j2"
    dest: "/etc/apache2/sites-available/example.com.conf"

- name: Enable new site
  shell: /usr/sbin/a2ensite example.com.conf
  notify: Reload Apache

- name: Disable default Apache site
  shell: /usr/sbin/a2dissite 000-default.conf
  notify: Reload Apache
```

Image 7 - Content of the config.yml file

The file explained

There are 3 tasks here, “Set up HTTP virtualHost Client”, “Enable new site” and “Disable default Apache site”.

The first task

The template module is used here, this is used to manage the configuration file for a virtual host in Apache. The template is used with src: "files/example.com.conf.j2" and dest: "/etc/apache2/sites-available/example.com.conf". This grabs the example.com.conf.j2 file that we still need to create in the files folder and places that file in the destination of "/etc/apache2/sites-available/example.com.conf" on the server.

The second task
Here we use the shell module, which is used to execute a shell command on the target hosts.The command we use is “/usr/sbin/a2ensite example.com.conf”, which enables the Apache virtual host configuration file for “example.com.conf”. The notify parameter is used to trigger the handler named "Reload Apache" after this task is completed.

The third task
Here we use shell again, with the command “/usr/sbin/a2dissite 000-default.conf”, which disables the default Apache site configuration file named “000-default.conf”. The notify parameter is used to trigger the handler named "Reload Apache" after this task is completed.

6.3.4 The configuration file for your virtual host

After this, go inside the “files” folder and create the conf file for your virtual host. I called it “example.com.conf.j2”. This is basically the “000-default.conf” file, only here you need to add a “.j2” to it, for it to work with the Ansible playbook. So how to create the conf file? Go to “https://gist.github.com/tjtoml/942d696c868b22a25259” and grab what is inside the “000-default.conf” file. Paste that into your created conf file and adapt it to your needs.You will then have something similar to the content shown in image 8.

```
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerName Kiosk
        ServerAdmin kobbe.vereecken@student.howest.be
        DocumentRoot /var/www/html

        #Redirect / http://ssm-howest.be/
        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
</VirtualHost>

# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Image 8 - Content of the example.com.conf.j2 file

Be advised, I named the file “example.com.conf.j2”, so if you have another name, replace “example.com.conf.j2” with your name in the other files of the Playbook.

And there we go, our playbook is made!

6.4 Execute the Ansible playbook

Go to your “ansibleBook” folder and open it. Now execute the following command:

```
ansible-playbook Configuration.yml -K
```

It will ask for your “Become password:” this is your WSL root password. Fill it in (correctly).
Now you will see something similar like this:

If that is the case, Apache will be installed. You will see that the default Apache website is running and see the service apache2 running as well. From here you can personalize it further to your needs.

7. Example of kiosk client

We created a small example on how the client of a MCDonalds kiosk could look like. The code is publically [available on github](#).
The following images show the most important screens.

Image 9 - The start screen

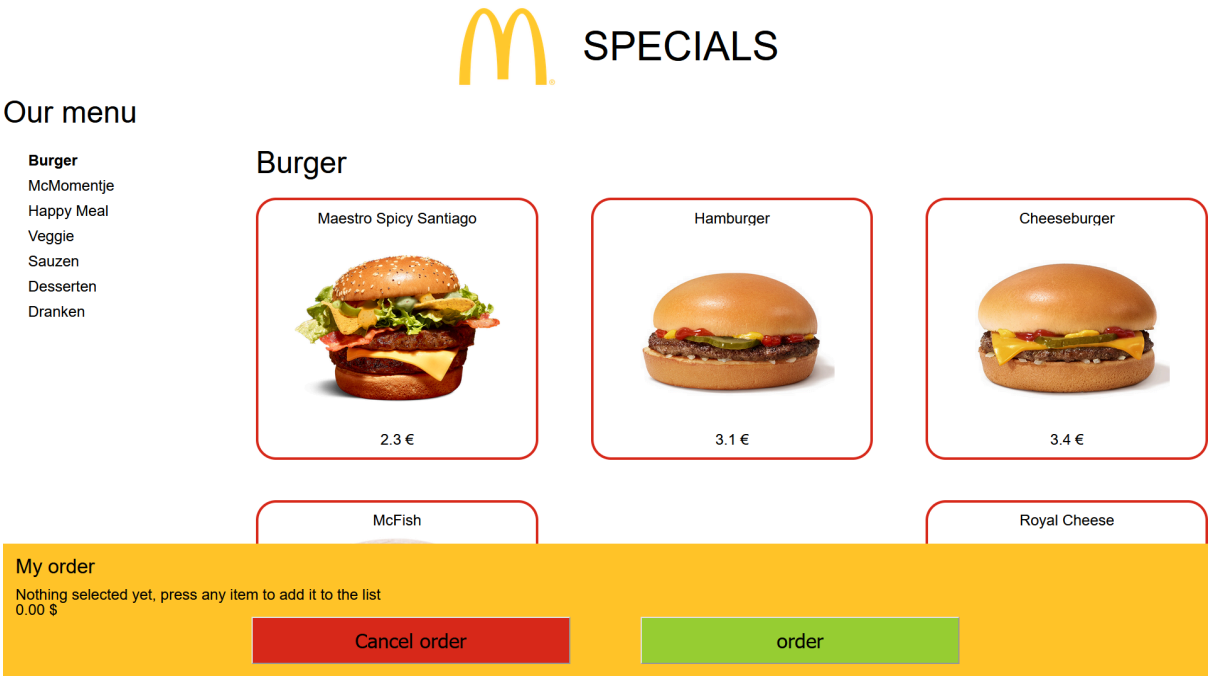


Image 10 - The main screen

8. Sources

1. Lizgt. “Set up a Single-app Kiosk on Windows - Configure Windows.” *Microsoft Learn*, 14 Feb. 2023, https://www.learn.microsoft.com/en-us/windows/configuration/kiosk-single-app#local
2. Dhawan, Rajesh. “Get SID From a User Account Using Powershell.” *TECHEPAGES*, 30 Jan. 2023, https://www.techepages.com/how-to-get-sid-from-the-user-account-in-powershell
3. Lizgt. “Find the Application User Model ID of an Installed App - Configure Windows.” *Microsoft Learn*, 14 Feb. 2023, https://www.learn.microsoft.com/en-us/windows/configuration/find-the-application-user-model-id-of-an-installed-app
4. Koreman, Koen. “Web Technology, Security and Honeypot” *Howest*, 2022-23, https://leho-howest.instructure.com/courses/17169