

Van Mol Gerrit

Professionele Bachelor Elektronica-ICT, afstudeerrichting ICT

Academiejaar 2021/2022

Platform voor automatisatie van firewall configuraties en deployment

Nomios

Excelsiorlaan 89.

1930 Zaventem.

Stagegegevens

Stagiair

Gerrit Van Mol

Opleiding

Elektronica-ICT, afstudeerrichting ICT

Academiejaar

2021/2022

Stageperiode

31/01 - 30/05/2022

Stagebegeleider

Stijn Vanbinnebeeck

Stageplaats

Nomios N.V. Belgium

Excelsiorlaan 89.

1930 Zaventem.

Mentor(en)

Sven Sanders

Automation platform for quick deployment of firewalls

G. Van Mol

The purpose of this project is automating deployment of firewalls. It will simplify the job of a technician that normally installs and deploys these devices. Automating firewall deployment prevents long and difficult procedures that would otherwise take a long time to get up and running with the device.

The first chapter of this report will cover research about the best possible frameworks available, afterwards a comparison is made between frameworks. Finally, one of these frameworks will selected and used to create a platform. Based on functionality requests, the platform is made to perform the most important tasks to make automated deployment of firewalls possible.

In conclusion this platform will provide a workflow that is quick and ease of use. It will eliminate repetitive tasks and long configuration hours. Eventually the platform will be saving time of technicians and improving efficiency of the company.

Keywords: Platform, automation, framework, time saving

Voorwoord

Voor het behalen van een diploma in het derde jaar en Elektronica-ICT, is het maken van een bachelorproef een vereiste. Deze bundel is het schriftelijke verslag ervan.

Deze bachelorproef is het resultaat van de kennis en vaardigheden die ik heb verzameld zowel op school als op mijn stageplaats. Het was een leerrijke uitdaging, waarbij ik ook heel wat nieuwe ervaringen opdeed. Bij de uitwerking van de bachelorproef kreeg ik tips en ondersteuning van mijn docenten, stagebegeleiders, familie en vrienden. In de eerste plaats zou ik graag de directie van de co-hogeschool Odisee willen bedanken voor de kans die ik kreeg er mijn opleiding te volgen.

Daarnaast wil ik alle docenten bedanken voor hun fantastische begeleiding door heen de jaren van mijn opleiding en Sven Sanders voor de fantastische begeleiding tijdens mijn stageperiode. In bijzonder wil ik de heer Johan Donné mijn docent applied-programming, data security en tal van andere vakken bedanken, om onder zijn deskundig toezicht theoretische informatie en praktische ervaring te verschaffen.

Ook wil ik graag de mensen bedanken die niet in mij geloofden voor het versterken van mijn motivatie en doorzettingsvermogen.

Tot slot gaat mijn dank naar mijn ouders Youri en Kathleen Van Mol, Ilona Van Mol en mijn grote zus Laure Van Mol voor hun aanmoediging en ondersteuning.

Met het rapport hoop ik de lezer een boeiend en leerrijk verslag te brengen van mijn bachelorproef.

Gerrit Van Mol

Kapelstraat 57 Lochristi, Juni 2022

Inhoud

Codefragmentlijst	1
Figurenlijst.....	2
Tabellenlijst	3
Afkortingenlijst	4
Begrippenlijst	5
Inleiding	6
1 Analyse opstelling	7
1.1 Huidige werkwijze	7
1.2 Snellere werkwijze.....	8
2 Platform structuur	9
2.1 Platform logica	9
3 Framework.....	12
3.1 Django geschiedenis	12
3.2 Django framework.....	13
3.3 Django architectuur (MVC/MVT)	13
3.3.1 Models	14
3.3.2 Views.....	15
3.3.3 Templates	16
4 Firewall configuratie	19
4.1 FortiManager	19
4.2 Zero Touch provisioning	20
4.2 Conclusie FortiManager	20
4.3 FortiManager scenario	21
5 Ansible	22
5.1 Ansible toepassingen	22
5.1.1 Provisioning	22
5.1.2 Configuratie management.....	22
5.1.3 Applicatie deployment	23
5.2 Continuous delivery	24
5.3 Conclusie Ansible.....	24
6 Ansible scenario	25
7 API-calls	26
7.1 Opstelling.....	27
7.2 Python virtuele omgevingen.....	28
7.3 Voordelen Python virtuele omgevingen.....	28
8 Continuous delivery	30
8.1 Jenkins	30

8.2 Jenkins minimumvoorraarden	32
8.3 Jenkins best practices	33
9 Hosting	33
9.1 Ideale productie opstelling	35
9.2 Volledige projectstructuur	37
Conclusie	38
Nwoord	39
Literatuurlijst	40
Bijlagenoverzicht	43
Bijlage 1: Platform flowchart	44
Bijlage 2: Fortimanager alternatief scenario	45
Bijlage 3: Ansible alternatief scenario	46
Bijlage 4: Basisopstelling	47
Bijlage 5: Development hosting diagram	48
Bijlage 6: Logboek	48
Bijlage 7: Real-world hosting scenario	49
Bijlage 8: Volledige opstelling	50

Codefragmentlijst

Codefragment 1: Django model code voorbeeld [5]	14
Codefragment 2: Voorbeeld inhoud views.py (datetime functie) [7]	15
Codefragment 3: urls.py voorbeeld URL [7]	16
Codefragment 4: views.py met context dictionary [8]	17
Codefragment 5: Template voorbeeld [8]	17
Codefragment 6: Voorbeeld Python functie met subprocess module [17]	26
Codefragment 7: Folder structuur virtuele omgeving.....	29
Codefragment 8: Voorbeeld Gunicorn "bind" comando.....	34
Codefragment 9: Nginx "sites-available" applicatie config voorbeeld	35

Figurenlijst

Figuur 1: Huidig "tijd-consumerend" proces	7
Figuur 2: Nieuwe werkwijze a.d.h.v. platform "grafische interface"	8
Figuur 3: Django platform pagina structuur	10
Figuur 4: Django tijdlijn [18]	12
Figuur 5: Django MVT-werking [25].....	13
Figuur 6: Visuele voorstelling model in databank [5].....	14
Figuur 7: Logica achter view (MVT) [6]	15
Figuur 8: View functie resultaat [7]	16
Figuur 9: Template output voorbeeld	17
Figuur 10: Zero touch provisioning (FortiManager) [10]	20
Figuur 11: FortiManager scenario (alternatief)	21
Figuur 12: Ansible stage levenscyclus [12]	22
Figuur 13: Zero downtime applicatie-update [20]	23
Figuur 14: Ansible scenario (alternatief).....	25
Figuur 15: Basisopstelling (binnen labo).....	27
Figuur 16: Python virtuele omgevingen visueel voorbeeld [9].	28
Figuur 17: Jenkins master-slave architectuur [15].....	31
Figuur 18: Jenkins visuele werking [23]	31
Figuur 19: Werking Django development hosting	33
Figuur 20: Werking services voor hosting in productie.....	34
Figuur 21: Productie opstelling in "real-world" scenario	36
Figuur 22: Overzicht volledige infrastructuur opstelling	37

Tabellenlijst

Tabel 1: Jenkins minimum hardware voorwaarden.....	32
Tabel 2: Jenkins minimumsoftware voorwaarden [17].....	32

Afkortingenlijst

ZTP	Zero Touch Provisioning
DB	Database
MySQL	My Structured Query Language
HTML	Hyper Text Markup Language
UI	User Interface
MVC	Model View Controller
MVT	Model View Template
IDE	Integrated Development Environment
SSL	Secure Sockets Layer
API	Application Programming Interface
JSON	JavaScript Object Notation
ORM	Object Relational Mapper
CRUD	Create, Read, Update, Delete
CSRF	Cross-site Request Forgery
XSS	Cross-site Scripting
IDE	Integrated Development Environment
DVCS	Distribution Version Control System
CI	Continuous Integration
CD	Continuous Development
CLI	Command Line Interface
RAM	Random-access memory
REST	Representational State Transfer

Begrippenlijst

Variabele	Element van bepaald type dat een tijdelijke waarde bijhoudt
Redundant	Herhaling/dubbele gegevens
Property	Eigenschap van een object
Provisioning	Het proces voor opstellen van IT-infrastructuur
Batch processing	Runnen van "jobs" dat geen gebruikers interactie nodig heeft
Staging	Een replica van de productie omgeving om testen uit te voeren

Inleiding

Dit project wordt een platform dat de opzet van firewalls automatiseert. Momenteel werkt de field engineer op een tijd consumerende manier waarbij alle configuraties van elk toestel handmatig gebeuren. Het platform geeft de technieker(s) de mogelijkheid snel en efficiënt meerdere toestellen te plaatsen binnen een bepaald tijdsframe. Aan de hand van zero touch provisioning zullen de techniekers weinig of geen voorkennis moeten hebben om een firewall toestel te plaatsen en operationeel te krijgen. Als een field engineer on-site is en geen basisconfiguratie aanwezig is op het toestel kan deze worden gedownload van het platform naar een USB-stick. Eenmaal de configuratie op de stick staat, kan deze worden overgezet naar de firewall. Met de basisconfiguratie aanwezig kan het platform verdere connectie maken met de firewall, om updates uit te sturen met verdere configuratie.

Er wordt onderzocht met welke technologieën en protocollen het platform best werkt om communicatie met de externe apparaten mogelijk te maken. Of het mogelijk is apparaten van verschillenden vendoren (tegelijk) te configureren. Hoe configuraties het efficiëntst kunnen worden beheerd. Of het mogelijk is meerdere connecties tegelijk tot stand te brengen tussen het platform en toestellen ter plekke. De opdrachtgever vraagt om gebruik te maken van het Django framework, Fortimanager, Ansible en/of alternatieven. Andere frameworks en automatisatie engines worden daarnaast ook onderzocht.

Bij deze opdracht heeft het bedrijf een aantal eisen. Deze omvatten dan vooral een volledig functionele login. Alsook had de opdrachtgever graag een geautomatiseerde basis set-up van een firewall waarbij connectiviteit zoals een ping naar Fortigate-manager functioneert zoals het hoort.

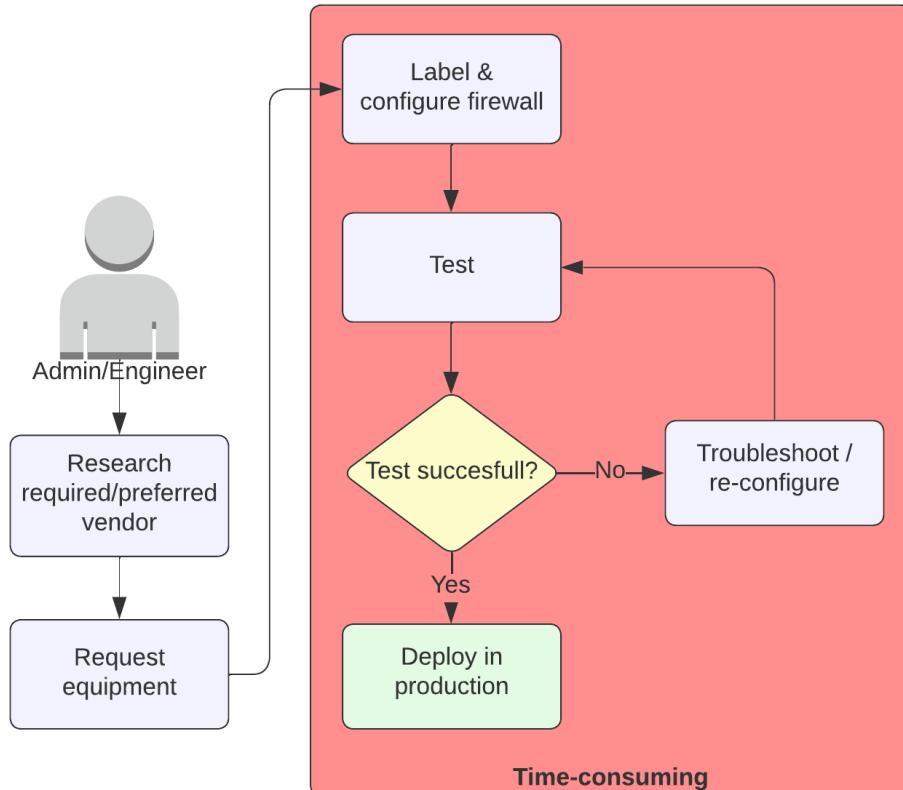
Daarnaast zijn er geen vooropgestelde beperkingen of technische randvoorwaarden waaraan moet worden voldaan.

1 Analyse opstelling

In volgende subhoofdstukken worden processen en relatiediagrammen beschreven die bij aanvang van het project worden gehandhaafd. Met behulp van onderzoek en bevraging is deze flowchart opgebouwd. Om het huidige proces te gaan optimaliseren, zal er een deel worden geautomatiseerd. Ook voor de geoptimaliseerde werkwijze is er een flowchart opgemaakt om het proces visueel te gaan verduidelijken. Deze automatische processen zijn vastgelegd en vooropgesteld door het externe bedrijf (bank) waar deze opdracht voor uitwerkt, wordt. Voor welke bank dit project wordt uitgewerkt zal niet worden beschreven aangezien dit in conflict komt met vertrouwelijke informatie die niet kan/mag worden gedeeld. Op vlak van infrastructuur en hosting van het platform is de volledige vrijheid gegeven. Hosting van het platform wordt later in detail besproken.

1.1 Huidige werkwijze

Tijdens een interview kon er een huidige werkwijze worden vastgesteld voor deployment van firewalls. Waarbij duidelijk werd dat binnen het bedrijf deze werk structuur is vooropgesteld en als standaard wordt aangenomen voor het registreren en uitrollen van firewalls. Daarnaast werd ook duidelijk dat de engineer een minimaal aantal stappen (Figuur 1: Huidig "tijd-consumerend" proces) moet ondernemen, voor een firewall van een labo naar deployment omgeving (on-site) kan worden verhuisd. Deze stappen moeten steeds worden herhaald voor elke individuele firewall, wat heel tijd-consumerend is.

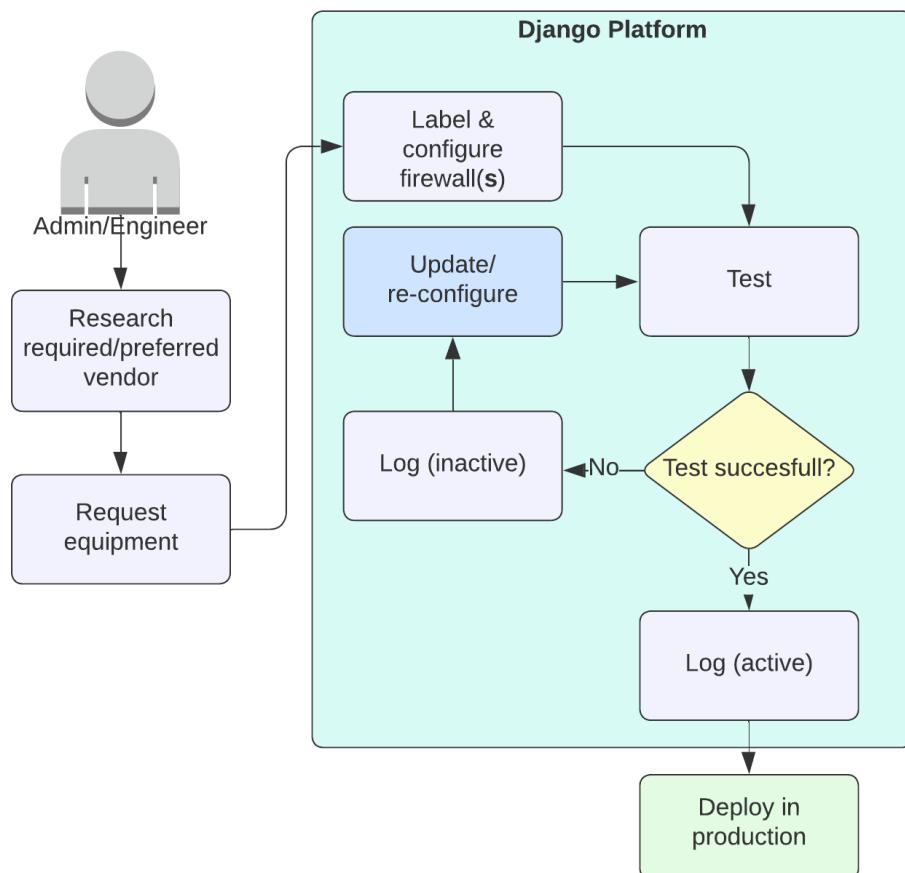


Figuur 1: Huidig "tijd-consumerend" proces

1.2 Snellere werkwijze

Omdat het configureren en testen van firewalls een steeds herhalend proces is, kan er een deel worden geautomatiseerd en versoepeld waardoor dit proces vlotter zal kunnen verlopen. Aan de hand van een platform met een grafische interface, worden handmatige stappen waar origineel steeds een terminal voor nodig was overbodig. Aan de hand van dit platform moet de engineer niet meer wisselen tussen terminal vensters en dergelijke. Dit is veel gebruiksvriendelijker en zorgt voor efficiëntie van gepresteerde werkuren. Alle configuraties of aanpassingen zal een engineer nu kunnen vanuit deze nieuwe interface.

In onderstaande illustratie (Figuur 2: Nieuwe werkwijze a.d.h.v. platform “grafische interface”) zijn de zo goed als alle stappen van het voorgaande proces in het nieuwe platform verwerkt.



Figuur 2: Nieuwe werkwijze a.d.h.v. platform “grafische interface”

2 Platform structuur

Omdat meerdere stappen worden samengesmolten voor het configureren van firewalls, zal aan de hand van het Django framework een grafische interface worden gemaakt om dit mogelijk te maken. Hoe dan ook voor een platform kan worden uitgewerkt moet er een structuur zijn van de pagina's die een gebruiker kan raadplegen, wat hun functie is en hoe ze aan elkaar hangen. In dit hoofdstuk zal elk deel zorgvuldig worden overlopen, met behulp van onderstaande figuur (Figuur 3: Django platform pagina structuur) waarbij de logica start vanaf de gebruiker (bovenaan) tot aan elk eindpunt/pagina.

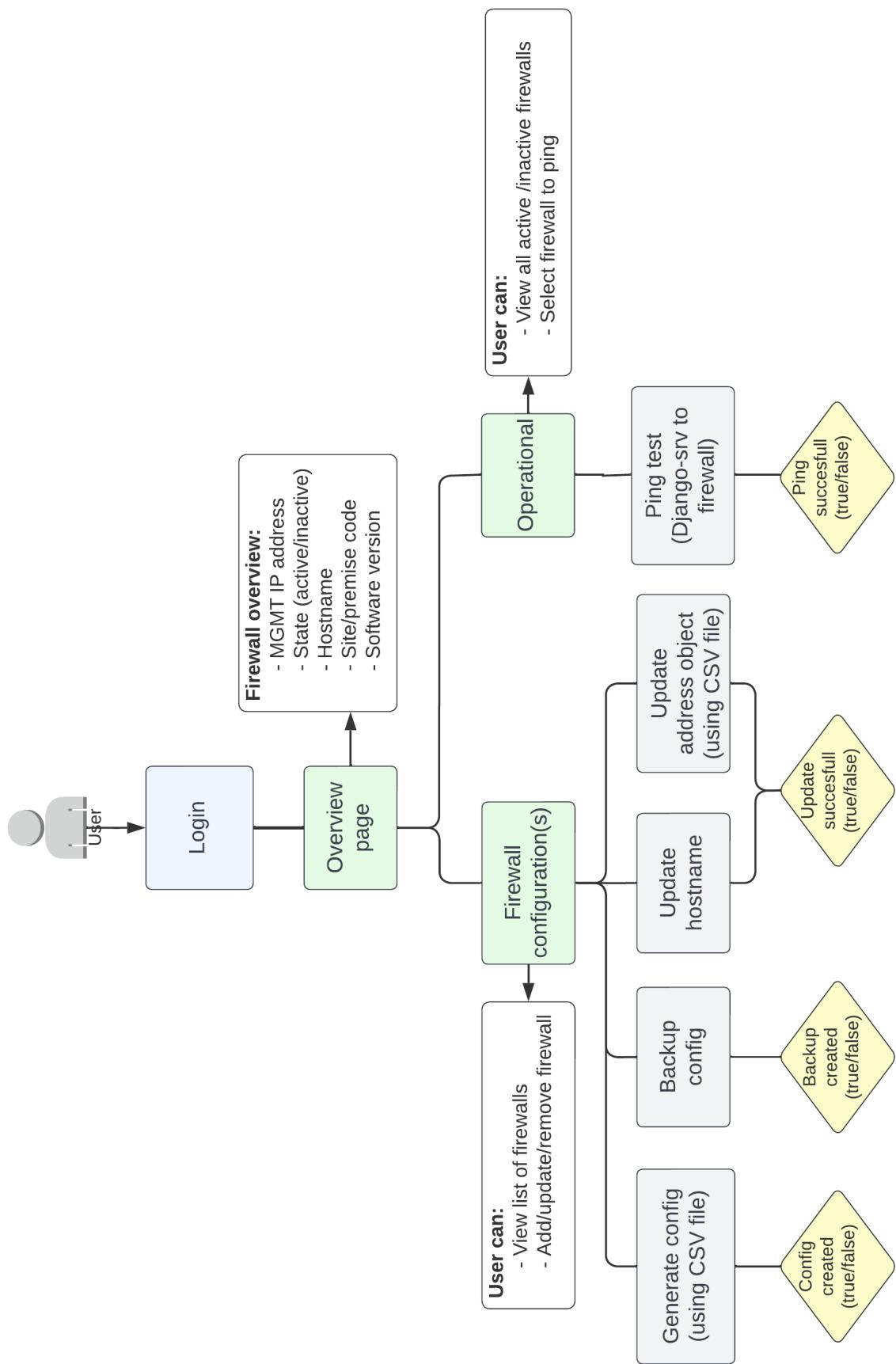
2.1 Platform logica

Voor iemand eventuele data kan raadplegen op een van de beschikbare pagina's moet er eerst worden ingelogd. Accounts worden aangemaakt door de administrator van het platform. Er mag geen mogelijkheid zijn om zomaar te registreren, door de eenvoudige reden dat dit erg onveilig kan zijn voor een publieke applicatie, waarbij iedereen zomaar gevoelige data van een bedrijf kan raadplegen en aanpassen. De administrator van het platform maakt aan de hand van een vooropgestelde lijst aan gekwalificeerde personen een aantal accounts aan. Dit kan de administrator aan de hand van een admin-paneel dat wordt voorzien door het Django framework zelf. Het account van de administrator zelf wordt aangemaakt bij creatie van de databank waar het platform zijn data op zal bewaren.

Eenmaal de gekwalificeerde gebruiker is aangemeld krijgt hij of zij een overzichtspagina te zien waarbij men minimum onderstaande puntjes kan raadplegen.

- MGMT Interface/IP
- State (active/inactive)
- Hostname
- Site/premise code
- Software version
- Vendor/type

De vernoemde puntjes geven een snel overzicht van alle beschikbare firewalls en hun basisconfiguratie. Als er een aanpassing/update moet gebeuren op een van de reeds beschikbare firewalls op de overzichtspagina, kan er worden verder gegaan door op een firewall te klikken. De gebruiker zal worden omgeleid naar de firewallconfiguratie pagina. Daar kan de gebruiker de gekozen firewall aanpassen en/of verwijderen. Indien er nog geen firewalls beschikbaar zijn in het overzicht kan er via dezelfde pagina (firewall configuratie) een firewall worden toegevoegd. Aan de hand van een vooropgestelde CSV/Excel bestand worden firewall configuratie parameters ingeladen om een basisconfiguratie te genereren voor de specifieke firewall. De nieuw gegenereerde firewall configuratie wordt opgeslagen op de server van het platform. Later kan de gebruiker kiezen om het bestand te downloaden en manueel te uploaden naar de firewall, of indien mogelijk automatisch de nieuwe configuratie te pushen naar de firewall(s) waarvoor de specifieke configuratie werd gecreëerd. Wanneer een configuratie wordt aangemaakt krijgt de gebruiker een succes status terug, zo weet de gebruiker zeker dat het genereren van de configuratie succesvol was of niet.



Figuur 3: Django platform pagina structuur

Ook op de firewall configuratie pagina kan de gebruiker een back-upconfiguratie van de gekozen firewall(s) opvragen. Hierbij worden de op dat moment actieve firewall configuraties opgevraagd en terug opgeslagen op de server van het platform. Indien er reeds een back-up is gecreëerd van een specifieke firewall zal deze worden overschreven. Opnieuw, om de gebruiker op de hoogte te brengen of (wanneer) de back-up succesvol is aangemaakt, wordt er terug een succes status bijgehouden en weergegeven.

Soms is het mogelijk dat een firewall moet her-labelt/verplaatst worden naar een andere site. Daarom is er de mogelijkheid om firewall hostnames en adres objecten aan te passen. Een hostname kan rechtstreeks worden aangepast op de interface van het platform. Adres objecten daarentegen moeten terug worden voorzien aan de hand van een CSV/Excel bestand. Dit bestand wordt geüpload en opgeslagen op de server van het platform. De gedefinieerde waarden worden uit het CSV-bestand opgehaald en omgevormd in een JSON-bestand zodat het via een API-call of alternatief kan worden geüpload naar de firewall. Na een hostnaam of adres object update, krijgt de gebruiker opnieuw een succes statusmelding. Op die manier weet de gebruiker met zekerheid dat de aanpassing is doorgevoerd.

Wanneer een firewall voor het eerst wordt geconfigureerd en toegevoegd aan het platform moet de engineer zeker zijn dat het apparaat operationeel is, daarom is er een pagina voorzien waar dit kan worden gecontroleerd. Op de "operational" pagina wordt er een overzicht gecreëerd van de actieve en inactieve firewalls. De staat van een firewall wordt bepaald afhankelijk van een ping (ICMP-protocol). Wanneer er een antwoord is op de ping zal op het platform worden weergegeven dat het toestel actief, ook wanneer geen reactie is op de verzonden ping zal er inactief worden weergegeven. Een firewall dat op inactief staat kan twee dingen betekenen;

1. De firewall is net toegevoegd en dus nog niet getest
2. De firewall heeft mogelijks een fout en kan niet (meer) worden getest

Wanneer een test wordt uitgevoerd krijgt de gebruiker een melding op de grafische interface die aangeeft of de ping is uitgestuurd of niet. Als dan de status van die specifieke firewall op inactief staat en zijn status niet update weet de engineer dat het toestel moet worden gecontroleerd op configuratiefouten of andere oorzaken.

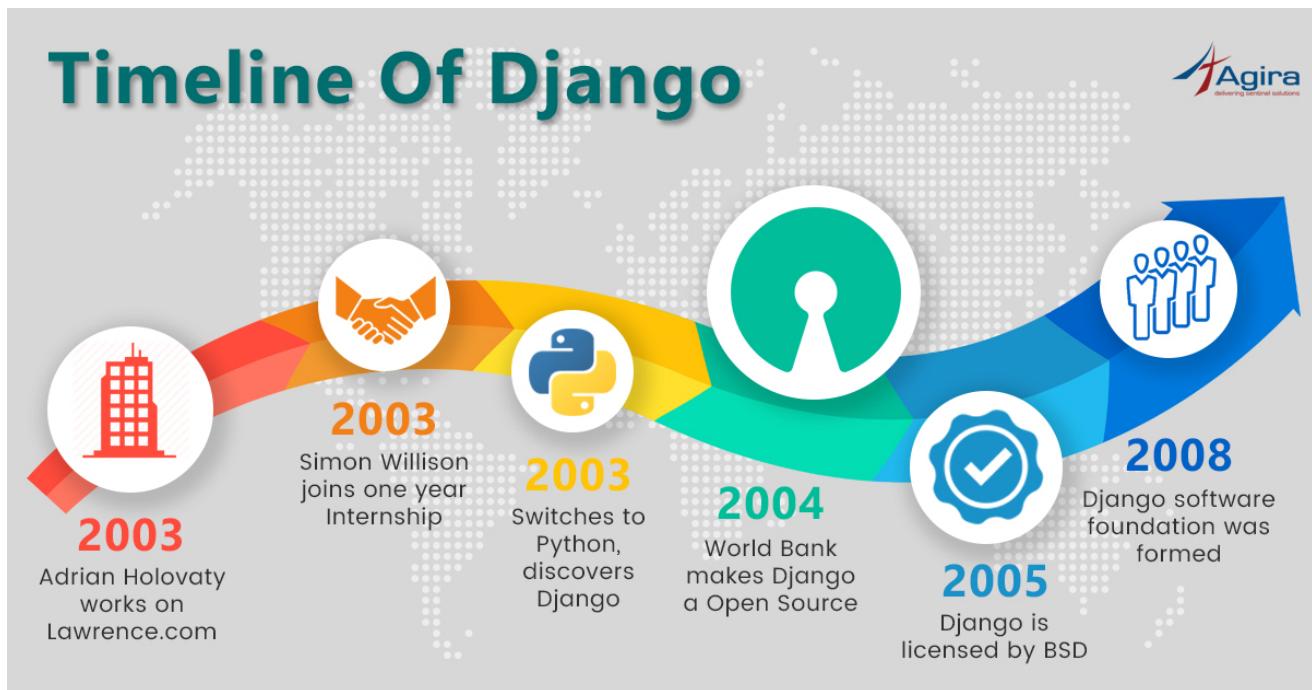
3 Framework

Om het platform uit te werken is er een framework nodig dat kan worden gebruikt als fundamenteel middelpunt om verder op te bouwen. Er werd voor dit project het Django framework opgelegd door de opdrachtgever waarbij een potentiële overweging voor eventuele andere frameworks wordt uitgesloten.

Hoe dan ook zal er in dit hoofdstuk worden terug gekeken op wat de geschiedenis is van Django, waar het vandaan komt en hoe het is ontstaan. Waarna wordt uitgelegd wat MVT is en wat het verschil is met MVC.

3.1 Django geschiedenis

Origineel is het Django framework ontstaan en ontworpen door "Lawrence journal world" in 2003 [1]. In 2004 werd het een open-source project en kreeg het de naam Django. De naam wordt uitgesproken als "Jang-oh" waarbij de "D" stil is [2]. Het is vernoemd naar de jazzgitarist Django Reinhardt. Een web team dat nieuwswebsites beheerde, maakte gebruik van veel repetitieve code en ontwerpen. Deze herhaalde code werd uiteindelijk uitgewerkt als een algemeen web development framework. Het framework werd steeds meer gekend en verbeterd waarbij de eerste officiële versie (v0.90) werd uitgerold in november 2005. Ondertussen zitten ze aan versie v3.2.12 sinds december 2021. Bij elke versie worden mogelijke problemen weggewerkt en features toegevoegd zoals nieuwe templates, database typen, etc. Django is ondertussen een veelzijdig framework dat de mogelijkheid biedt meerdere soorten websites te creëren.



Figuur 4: Django tijdlijn [18]

3.2 Django framework

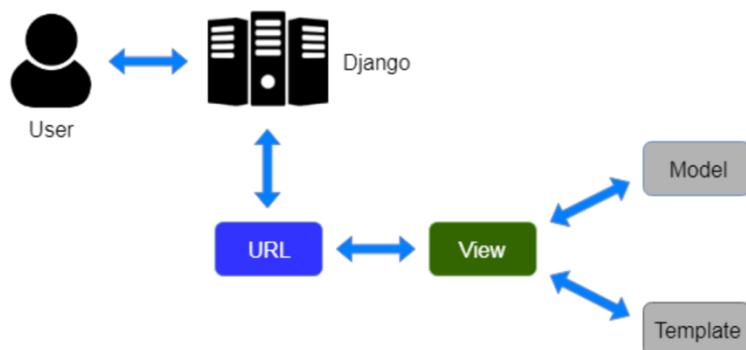
Django is een gratis open-source framework, gemaakt om te focussen op het ontwikkelen van onderhoudbare websites. Sinds 2005 geeft het de mogelijkheid om snel maar ook veilige en schaalbare websites te ontwikkelen in de Python programmeertaal. Het framework heeft een zeer actieve gemeenschap waar elke ontwikkelaar steeds terecht kan voor vragen. Daarnaast is de documentatie ook zeer uitgebreid en goed onderhouden. Het grootste voordeel van dit framework is dat het over een rijke feature set beschikt van meer dan 10 000 packages. Handig ingebouwde packages zijn bijvoorbeeld, API's, CMS, UA, CAPTCHA-ondersteuning en form validatie. Django is gekend voor de "batteries-included" filosofie. Het principe achter "batteries-included" is dat de algemene functionaliteit voor het bouwen van een web-applicatie zou moeten meegeleverd worden in het framework, in plaats van aparte bibliotheken die deze functionaliteit leveren. Daarnaast wordt ontwikkeling ondersteund door een non-profit organisatie, wat de ontwikkeling vooruithelpt en bedrijven/individuen zekerheid geeft over de levensduur en vooruitzichten van het framework [3] [4].

3.3 Django architectuur (MVC/MVT)

Het makkelijkste aan Django is dat alle achterliggende delen al inbegrepen zijn, zo is het niet nodig om een backend te maken aan de hand van API's, JavaScript, etc. Omdat Django gebruik maakt van het MVC/MVT-software design is het niet meteen nodig om gebruik te maken van alternatieve backend opties. Dit softwaredesign is een collectie van drie belangrijke componenten Model, View en Template (controller). In andere programmeertalen is de "Model View Controller" (MVC) architectuur een bekende standaard. Het Django framework daarentegen beschikt over de "Model View Template" architectuur. Het grootste verschil zit in de "view" deel van beide architecturen.

Bij Django wordt de view van MVC gezien als de data logica en beschrijft de data die wordt gepresenteerd aan de gebruiker. In tegenstelling tot hoe iets wordt gepresenteerd naar de gebruiker. De view beschrijft dus welke data wordt vertoond en niet hoe de data kan worden vertoond aan de gebruiker.

Het is dus logisch om de inhoud te scheiden van de presentatie, dat is waar templates een rol spelen. In Django beschrijft een "view" welke gegevens worden gepresenteerd, maar een view draagt hier dus data over naar een template, die op zijn beurt beschrijft hoe de gegevens worden gepresenteerd. De controller in Django is het framework op zich [2]. In onderstaande afbeelding (Figuur 5: Django MVT-werking) kan men een visuele weergave terugvinden van hoe de MVT-structuur in zijn werk gaat.



Figuur 5: Django MVT-werking [25]

3.3.1 Models

Aan de slag gaan met een gewone databank is vrij complex en omslachtig, omdat de ontwikkelaar moet weten welke vele query's nodig zijn voor het aanmaken, verwijderen of aanpassen van tabellen en/of andere database-gerelateerde zaken. Modellen brengt de complexiteit van SQL (Structured Query Language) databanken naar beneden door taken en tabellen te structureren in modellen.

Een model is een soort interface voor de data en is verantwoordelijk voor het onderhouden van de data. Het zorgt voor een logische structuur achter de volledige applicatie en wordt gerepresenteerd door de databank (relationele databanken zoals MySQL of Postgres). Een model in Django wordt gebruikt voor het maken van tabellen, data velden en eventuele extra parameters. Algemeen zal elk model in Django een enkele tabel voorstellen in de databank.

Aan de hand van het meegeleverde admin-paneel van Django, kunnen gemakkelijk datavelden worden aangemaakt, aangepast, verwijderd of opgehaald van een model samen met vele bijkomende operaties. Django modellen bieden simpelheid, consistentie en version-control. De basis eigenschappen van een model zijn als volgt;

- Elk model is een Python klasse dat subklassen creëert van het "django.db.models.Model"
- Elk attribuut van een model represeneert een database veld
- Geeft een automatisch gegenereerd/toegang tot de database

Een Django model voorbeeld kan in onderstaand codefragment worden geraadpleegd.

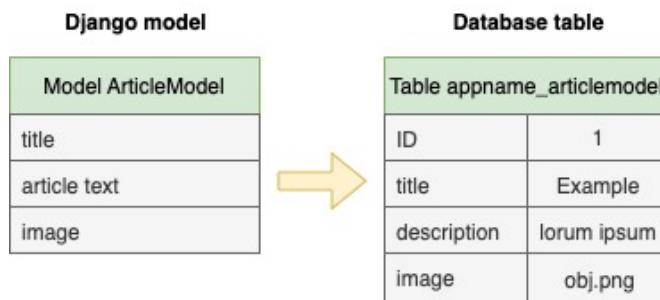
```
from django.db import models

# Create your models here.

class ExampleModel(models.Model):
    title = models.CharField(max_length = 200)
    description = models.TextField()
```

Codefragment 1: Django model code voorbeeld [5]

Wanneer een model is aangemaakt in Django moet de applicatie voor opstart worden gemigreerd. Wat betekent dat alle modellen die nog niet aanwezig waren als tabel in de databank op dat moment worden aangemaakt. In onderstaande afbeelding (Figuur 6: Visuele voorstelling model in databank) is een visuele voorstelling van wat er gebeurt als een model wordt gemigreerd naar de databank [5].

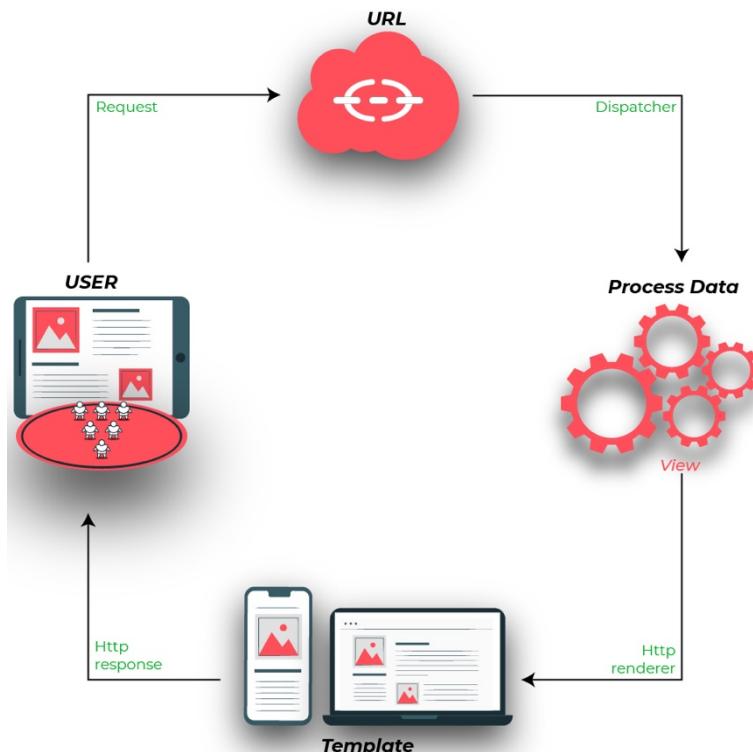


Figuur 6: Visuele voorstelling model in databank [5]

3.3.2 Views

Django views zijn één van de meest belangrijke onderdelen van de MVT-structuur. Zoals eerder al werd vermeld, is een view een methode dat web requests verwerkt en een antwoord teruggeeft. Dit antwoord kan HTML-inhoud, redirect, 404 error of eender wat zijn dat een webbrowser kan afbeelden.

Django views zijn een deel van de interface en renderen de HTML/CSS/JavaScript in een template om uiteindelijk weer te geven op een webpagina [6]. Onderstaande afbeelding (Figuur 7: Logica achter view (MVT)) geeft visueel beeld van hoe de logica van een view in elkaar zit.



Figuur 7: Logica achter view (MVT) [6]

Wanneer er een project is aangemaakt, zijn er een aantal bestanden die worden meegeleverd. Eén van die bestanden is de "views.py", daarin worden methoden gedefinieerd die de binnenkomende web requests verwerken. In onderstaand codefragment wordt de inhoud van een voorbeeld "views.py" bestand weergegeven.

```
# import HttpResponse from django
from django.http import HttpResponseRedirect
# get datetime library
import datetime

# create a function
def CurrentDate_view(request):
    # fetch date and time
    now = datetime.datetime.now()
    # convert to string
    html = "Time is {}".format(now)
    # return response
    return HttpResponseRedirect(html)
```

Codefragment 2: Voorbeeld inhoud views.py (datetime functie) [7]

Om de gedefinieerde view functie weer te geven op een webpagina moet de functie worden opgeroepen in het “urls.py” bestand samen met een passende URL. Wanneer de gebruiker dan die specifieke URL bezoekt, krijgt hij of zij data terug gedefinieerd in de voorgaande functie. Onderstaand codefragment roept de functie “CurrentDate_view” op uit het “views.py” bestand (Codefragment 2: Voorbeeld inhoud views.py (datetime functie)).

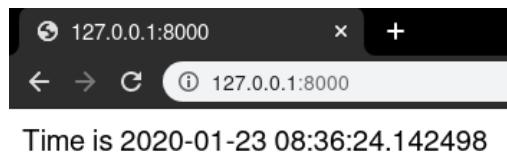
```
from django.urls import path

# importing views
from app_name import views

urlpatterns = [
    # default path calls function 'CurrentDate_view' from views.py
    path('', views.CurrentDate_view),
]
```

Codefragment 3: urls.py voorbeeld URL [7]

Wanneer de thuispagina (“#”) wordt bezocht door de gebruiker, zal de voorgaand vermelde functie (“CurrentDate_view”) worden opgeroepen en gelijkaardig resultaat als onderstaande afbeelding zal worden weergegeven.



Figuur 8: View functie resultaat [7]

3.3.3 Templates

Een template bestaat uit statische inhoud zoals standaard-HTML/CSS en JavaScript in een HTML-bestand. Naast de standaard-HTML en CSS bestaat een template ook speciale syntax, die beschrijft hoe dynamisch inhoud wordt geïmplementeerd. Het Django framework genereert en behandelt dynamische webpagina's zichtbaar voor de eindgebruiker. Aan de hand van de Django backend (models/views) wordt er data opgehaald en doorgegeven aan een template. De data wordt dan door behulp van een template mooi afgebeeld naar de gebruiker toe. Templates kunnen van elkaar overerven, bijvoorbeeld een header template kan worden verwerkt in een standaard webpagina template waarbij ze samen een geheel vormen voor de eindgebruiker [8].

Om data van de view (views.py) in een template te refereren moet er een (dictionary) context worden gecreëerd in een bepaalde view functie, in onderstaand codefragment (Codefragment 4: views.py met context dictionary) is een voorbeeld zichtbaar.

```

# import HttpResponse from django
from django.shortcuts import render

def GetData_view(request):
    # create a dictionary to pass
    # data to the template
    context ={
        "data":"my data here",
        "example":":Hello there",
    }
    # return response with template and context
    return render(request, "base.html", context)

```

Codefragment 4: views.py met context dictionary [8]

Nu er een context is gecreerd in de view functie en is gekoppeld aan een template (base.html) in de “return”, kan de gedefinieerde data worden opgehaald in het template. In onderstaand codefragment worden de variabelen “data” en “example” opgeroepen, waarna de tekst achter deze variabelen zal worden afgebeeld.

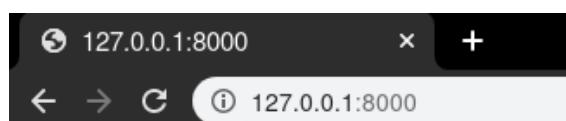
```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Homepage</title>
</head>
<body>
    <p> Data is {{ data }}</p>
    <p> General Kenobi says {{ example }}</p>
</body>
</html>

```

Codefragment 5: Template voorbeeld [8]

Volgende figuur (Figuur 9: Template output voorbeeld) wordt als resultaat weergegeven aan de gebruiker;



Data is my data here
General Kenobi says Hello there

Figuur 9: Template output voorbeeld

Nu er een beter beeld is gecreëerd van wat de MVT-architectuur is, kan er met zekerheid worden gezegd dat het een cruciaal en onmisbaar deel van Django is waardoor vele taken eenvoudig en zeer dynamisch worden.

Als er wordt teruggekeken op voorgaande afbeelding (Figuur 5: Django MVT-werking) kan de MVT-architectuur als volgt worden samengevat;

Het Model helpt met beheer van de databank en is de laag die zorgt voor toegang tot de data. De View wordt gebruikt als logica laag die zorgt voor interactie met de modellen om data over te dragen, deze data wordt later weergegeven in een Template. De Template is de presentatie laag die de gebruikersinterface volledig behandeld.

Er is geen aparte controller en de complete applicatie is gebaseerd op het Model View Template design, de applicatie wordt dus volledig behandeld door het framework zelf [25]. Als een model is aangemaakt in Django kan het worden gemigreerd in de databank waarna het meteen in gebruik kan worden gesteld. Deze opstelling maakt Django een zeer krachtig framework waardoor er kan worden gefocust op de ontwikkeling van de applicatie zelf en er is minder werk vereist van de ontwikkelaar waardoor hij of zij van scratch, zou moeten beginnen. Andere features zoals user management, software administratie features en anderen zijn steeds integreerbaar via packages [3] [4].

Django is voorbestemd voor snelle webontwikkeling. Deze snelheid is één van de beste eigenschappen van Django en de kracht dat het de ontwikkelaar biedt zonder de functionaliteit en beveiliging naar beneden te brengen. Dat alles in één van de eenvoudigste en populairste talen ter wereld gekend als Python.

Wanneer de ontwikkelaar alle vrijheid wil in aanpassingen en wat er toegepast wordt op de applicatie is Django minder aangeraden. In dergelijke gevallen kunnen alternatieven voor Django worden gebruikt, zoals Flask, CherryPy, etc. Dit zijn web frameworks die kunnen worden gebruikt voor meer eenvoudig flexibele websites. Daarnaast zijn er nog enkele niet-Python frameworks zoals Node.JS, Laravel, React die ook een goed alternatief zijn voor Django. Maar dan is kennis van deze andere programmeertalen een vereiste [5].

Hoe dan ook, Django is een ideaal web framework voor het ontwikkelen van schaalbare en complexe applicaties.

De gemeenschap ontwikkelt ook voortdurend nieuwe en interessante functionaliteiten om toe te voegen aan het reeds fantastisch framework.

Voor dit project zal het Django framework dus een centraal punt zijn waar niet alleen de gebruiker maar ook de te onderhouden toestellen zullen samen komen. In een volgend hoofdstuk zal er worden onderzocht hoe er kan gecommuniceerd worden met deze toestellen, op welke manier en met behulp van welke technologie/protocol dat zal zijn.

4 Firewall configuratie

Om verschillende firewalls van andere fabrikanten te kunnen configureren/updaten moet er een manier zijn om deze aan te spreken. Omdat er tussen fabrikanten geen gedefinieerde standaard is van hoe er met firewalls kan worden gecommuniceerd (buiten SSH), is er een automatisatie engine, API-calls of ander alternatief nodig om dit te kunnen realiseren. Het is zo dat een gebruiker bij het toevoegen van een firewall op het Django platform, moet kunnen aanduiden van welke vendor het product is. Op basis daarvan kan er achterliggend een andere connectie of basisconfiguratie worden opgesteld.

De reden dat er een mogelijkheid moet zijn om meerdere firewall vendors te kunnen configureren is als volgt: Indien de gebruiker ooit van vendor wisselt, is het platform nog steeds bruikbaar. Daarnaast hangt de gebruiker ook niet vast om te kiezen voor een specifieke vendor, omdat het platform enkel die vendor ondersteunt. Er werd bij aanvang van het project voorgesteld om het platform samen te laten werken met Fortimanager. Hoe dan ook als Fortimanager in gebruik wordt genomen komt men in de voorgaand vernoemde situatie terecht waarbij de gebruiker vast hangt aan een vendor (Fortinet in dit geval).

Daarom zijn er een aantal alternatieven die eventueel een oplossing kunnen bieden, zoals Ansible of rechtstreekse API-calls vanaf het Django platform. Hoe dan ook is belangrijk te weten wat de verschillende technologieën zijn en waar ze toe instaat zijn.

4.1 FortiManager

FortiManager is een platform gemaakt om gecentraliseerd en automatisch beheer te kunnen voorzien voor meerdere FortiGate toestellen. Het platform voorziet controle, segmentatie en algemeen consistente bescherming voor toestellen, applicaties en gebruikers. Dit platform is een voorkeur van klant omdat het product specifiek is (Fortinet) en meer functionaliteit integreert dan andere alternatieven zoals IBM Security Guardium en WatchGuard Dimension. Het FortiManager platform voorziet ondersteuning om één tot honderdduizend toestellen te configureren vanuit één centrale console.

FortiManager beheert indirect de FortiAP, FortiSwitches en andere Secure Access toestellen via de FortiGate. Om zero touch-configuraties te ondersteunen, maakt FortiManager gebruik van de functie "Add Model Device" waarmee gebruikers een toestel kunnen toevoegen. Eenmaal een FortiGate met een overeenkomende ID is geregistreerd bij de FortiManager wordt er automatisch de configuratie toegepast op dat specifieke toestel. De klant heeft een voorkeur voor FortiManager omdat er tijdens aanvang van het project wordt samengewerkt met Fortinet als leverancier.

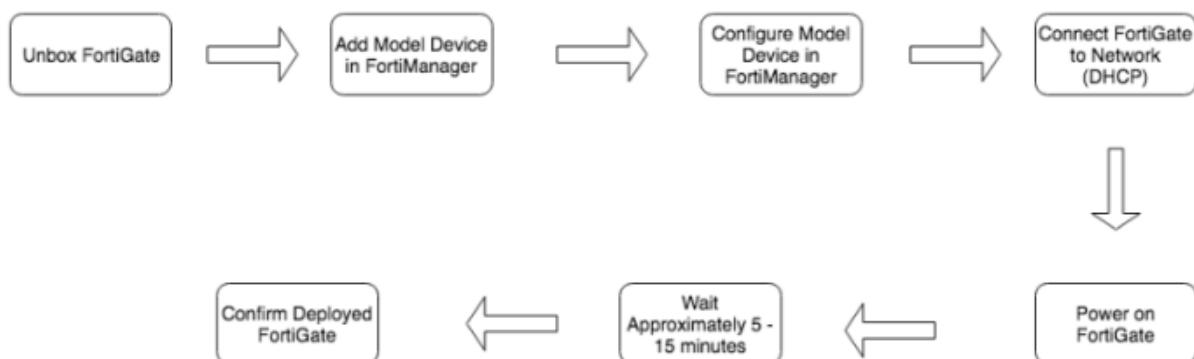
4.2 Zero Touch provisioning

Zero touch provisioning betekent dat een configuratie aan een apparaat wordt toegewezen zonder tussenkomst van de gebruiker, bij voorkeur voordat er toegang is tot de fysieke hardware. Er zijn meerdere methoden om dit soort functionaliteit te bereiken.

Zero touch provisioning is bedoeld om een interessant probleem op te lossen. Het probeert de "menselijke fout" te verminderen bij het configureren van een toestel voor/tijdens productie. Of nu een fout wordt veroorzaakt door stress en tijdsdruk, er is altijd kans dat de persoon die de configuratie uitrolt een kleine fout of storing van twee minuten kan veranderen in een storing van twee uur.

Het hebben van een niet-stressvolle, lagedrukomgeving om vooraf een configuratie goed te plannen en toe te passen, vergroot de kansen op een succesvolle implementatie enorm. In zekere zin geeft deze functie een kort moment tijdens de implementatie waar de engineer de "handen van het stuur" kan halen terwijl de configuratie naar het apparaat wordt geüpload.

Daarnaast is zero touch provisioning belangrijk omdat het de beheerder in staat stelt een configuratie voor meerdere apparaten in te stellen voordat er toegang wordt gegeven tot het fysieke of virtuele toestel. Hierdoor kan de beheerder offline tools gebruiken om snel apparaten in bulk te configureren en implementeren. Fortinet biedt een aantal tools om dit te bereiken bij het implementeren van FortiGate-firewalls [10].



Figuur 10: Zero touch provisioning (FortiManager) [10]

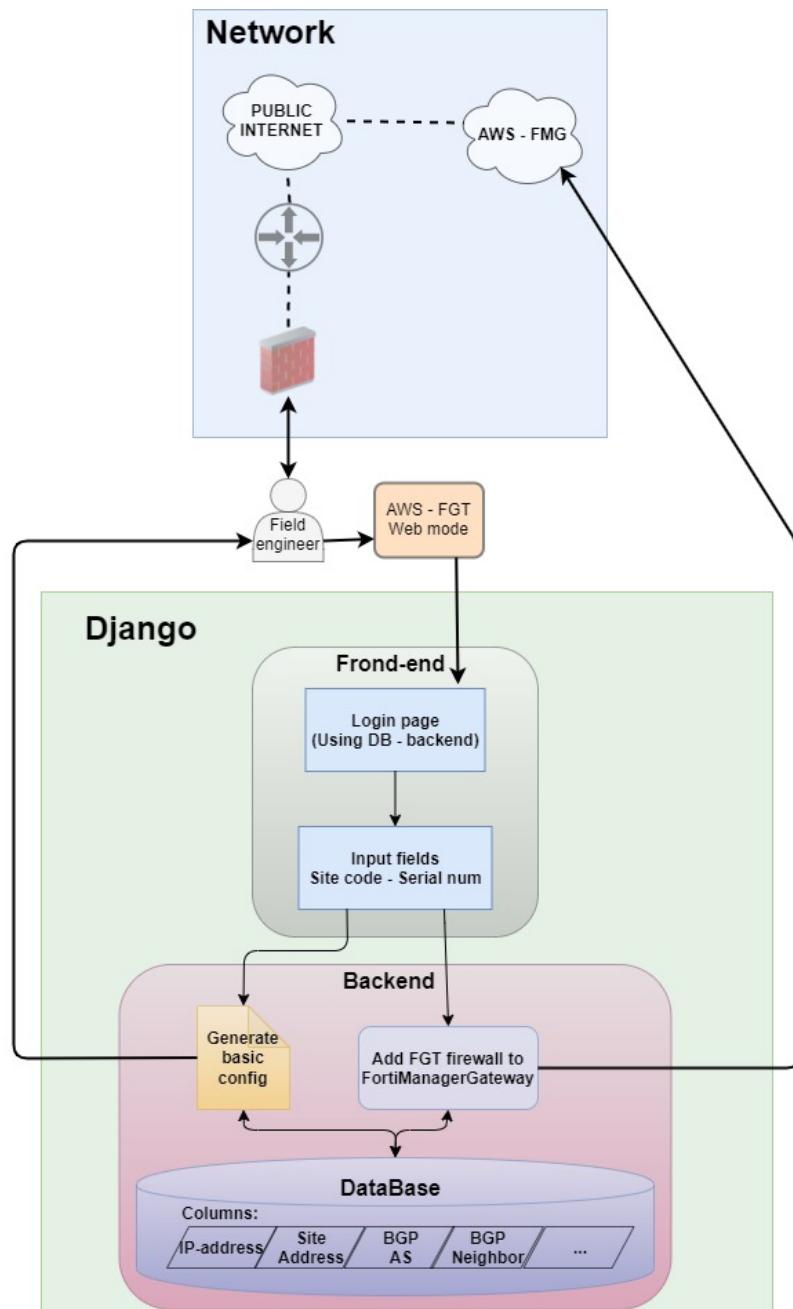
4.2 Conclusie FortiManager

Er kan met zekerheid worden gezegd dat FortiManager veel mogelijkheden en moeiteloosheid naar voren brengt. Hoe dan ook is er opnieuw het probleem van de compatibiliteit met andere toestellen/vendoren. Wanneer er bijvoorbeeld een contract werd getekend door de klant, om een minimumaantal jaar bij Fortinet te blijven als leverancier is er geen twijfel om voor Fortimanager te kiezen. Maar indien de klant steeds de vrijheid heeft om te wisselen van leverancier, is FortiManager een limiterende optie op vlak van compatibiliteit en uitbreidbaarheid naar toekomstige firewall installaties.

4.3 FortiManager scenario

Ondanks dat FortiManager niet wordt gebruikt bij aanvang van het project, is het interessant om het scenario waarbij het eventueel wel wordt gebruikt voor te stellen. Wanneer FortiManager wel wordt gebruikt zal deze als een soort tussenpion spelen en dirigent zijn van alle firewalls.

Het Django platform zou nog steeds worden gebruikt, maar nu enkel voor het aanmaken van de basisconfiguratie en het testen van de firewalls. Eenmaal de configuratie is gemaakt wordt de firewall toegevoegd op de FortiManager-gateway en de configuratie gepusht. Aanpassingen en updates worden na configuratie vanaf dat moment ook gedaan vanuit de FortiManager-gateway. Onderstaande figuur (Figuur 11: FortiManager scenario (alternatief)) geeft een beeld van de potentiële opstelling wanneer dit scenario zou worden uitgewerkt. In het onderstaande schema wordt verondersteld dat de FortiManager-gateway wordt gehost in de cloud van AWS (Amazon Web Services).



Figuur 11: FortiManager scenario (alternatief)

5 Ansible

Naast Fortimanager is er Ansible. Dit is een open-source automatisatie engine die repetitieve taken zoals configureren van servers of uitrol van applicaties kan gaan automatiseren. Aan de hand van CI/CD kunnen snel vernieuwde versies van een applicatie online gebracht worden zonder enige downtime. Ansible is vooral gekend in de tak van “infrastructure as code” en is vergelijkbaar met het zero touch provisioning principe van FortiManager. Het verschil met FortiManager en Ansible is dan dat Ansible zo goed als alle soorten toestellen kan configureren, waarbij FortiManager specifiek is voor Fortinet producten. Ansible brengt meer consistentie, betrouwbaarheid en schaalbaarheid naar eender welke IT-omgeving. Met de YAML-taal is het ook gemakkelijker om scripts/configuraties te lezen en schrijven als standaard Engels.

Gebruikmakende van een enkele controleserver kunnen meerdere machines worden aangestuurd. Alle machines die onder de controleserver vallen worden aangesproken via SSH. Met SSH kan de controleserver updates en configuraties uitvoeren op andere machines. Het is mogelijk om naast SSH-authenticatie gebruik te maken van Kerberos, LDAP of andere authenticatie managementsystemen [11].

Ansible kan dus worden gebruikt door meerdere applicaties zoals provisioning van virtuele machines, configuratie management, applicatie uitrol, beveiliging automatisatie en meer. Om de Django applicatie of firewalls vlot uit te rollen en online te krijgen kan de Ansible engine een belangrijke rol spelen. Mogelijks in samenwerking met andere services of achterliggende Linux-machines.



Figuur 12: Ansible stage levenscyclus [12]

5.1 Ansible toepassingen

5.1.1 Provisioning

Bare-metal servers hebben af en toe nood aan provisioning. Met Ansible kan iedereen veel zaken integreren, zoals datacenterbeheer-tools om servers aan te spreken en een provisioning te starten. Bij gebruik van verschillende hypervisor(s), virtuele opslag/netwerken kan Ansible een diverse selectie van modules en omgevingen ondersteunen. Door gebruik van sommige modules is het mogelijk om gemakkelijker te gaan schalen.

Naast bare-metal servers en hypervisor(s) is het ook mogelijk om netwerkinfrastructuur en cloudoplossingen te leveren. De netwerk-automatiseringsmogelijkheden zorgen voor validatie en continue werking van fysieke netwerkapparaten.

Wanneer Ansible een rol speelt voor een (publieke of private) Cloud omgeving is het mogelijk om niet alleen de clouddiensten te leveren, maar ook de onderliggende infrastructuur en applicaties binnen de Cloud te onderhouden [12].

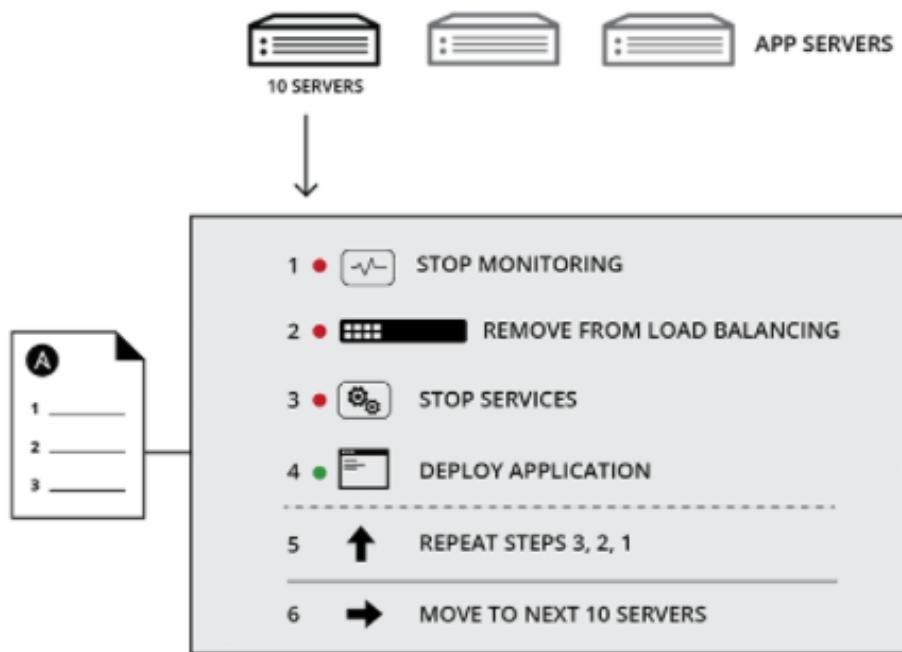
5.1.2 Configuratie management

Ansible is een state-driven resourcemodel, wat inhoudt dat het de toestand van de systemen en services beschrijft. De huidige status van de doelmachine is niet relevant, aangezien Ansible de status van de machine naar wens zal transformeren. Dit leidt tot minder potentiële fouten in vergelijking met script-gebaseerde oplossingen, omdat deze vaak onomkeerbare acties veroorzaken. Ansible heeft ook de mogelijkheid om de configuratie te valideren op mogelijke syntaxfouten [13].

5.1.3 Applicatie deployment

Zoals eerder al werd vermeld is het mogelijk om te beschrijven wat de gewenste staat is van een machine. De gewenste staat van een machine is gedefinieerd in een Ansible "Playbook". Playbooks worden uitgevoerd tegen een groep/enkele host waarbij de gewenste staat voor elke host hetzelfde is. Op die manier kan elke ontwikkelaar configuratiefouten vermijden en is de configuratie met zekerheid op elke machine hetzelfde. Op die manier maakt Ansible herhalende opdrachten zoals jaarlijks onderhoud en andere configuraties betrouwbaarder.

Playbooks worden verzonden via SSH naar de gedefinieerde doelhost met de OS-compatibele modules. Toestel specifieke modules zijn vergelijkbaar met opdrachten in de CLI voor het besturingssysteem van de doelmane. Zodra de machine een module heeft ontvangen, worden ze ter plaatse uitgevoerd, eenmaal de taak is voltooid (staat geüpdatet) wordt de module verwijderd van de doelhost [13].



Figuur 13: Zero downtime applicatie-update [20]

Zoals te zien is in bovenstaande afbeelding (Figuur 13: Zero downtime applicatie-update) is het mogelijk om applicaties en updates uit te rollen, zonder dat de gebruiker enige downtime opmerkt. De gebruiker merkt geen downtime omdat elke server of servergroep apart worden geüpdatet met behulp van de load-balancer.

Wanneer de ontwikkelaar gebruik maakt van een Distributed Version Control System is het mogelijk om downloadbare artifacts te maken van actieve applicaties. Hoe dan ook, het is ook mogelijk om gebruik te maken van REST API's waarbij een aantal variabelen worden aangepast op een andere service. Als er eenmaal aanpassingen zijn of de uitrol is van een nieuwe update, kunnen mail services of een pushnotificatie worden opgesteld en verstuurd. Op die manier is de ontwikkelaar steeds up-to-date en weet hij of zij dat de aanpassing is doorgevoerd of niet.

5.2 Continuous delivery

Met de stage- testmethode kunnen Ansible- "plays" worden gevalideerd en getest. In het inventory bestand is het mogelijk om omgevingen in te delen in verschillende groepen van machines. Dit maakt het mogelijk om test en productiemachines te definiëren. Het is echter aangeraden om productie en testmachines te definiëren in aparte inventory bestanden, op die manier worden enige verrassingen voorkomen wanneer een configuratie wordt uitgerold. Wanneer een "play" klaar is, kan het worden getest op de staging machines. Als de test slaagt, kan ervoor gekozen worden om ook automatisch de "play" op productiemachines uit te voeren waarbij het eindproduct beschikbaar wordt voor gebruikers.

Weet dat een "play" bestaat uit een set aan geschreven instructies in de YAML-taal om een gewenste machineconfiguratie in te tellen. Deze machines kunnen bestaan uit monitoringsystemen, netwerk toestellen, load-balancers, webservers en andere soorten toestellen. Een veelvoorkomende manier om Ansible te gebruiken is met een CI-systeem waarbij succesvolle builds van een play worden gepubliceerd [14].

Taken die plaatsvinden bij het testen:

- De CI zorgt ervoor dat Ansible een playbook uitvoert om een staging omgeving te implementeren met de nieuwe apparaat configuratie of toepassing
- Wanneer de test slaagt, kan er een melding worden verzonden om de uitrol te bevestigen naar productieomgeving
- Na publicatie kan de configuratie of applicatieversie worden opgeslagen in een artefact op de CI-server voor latere raadpleging of andere doeleinden.

5.3 Conclusie Ansible

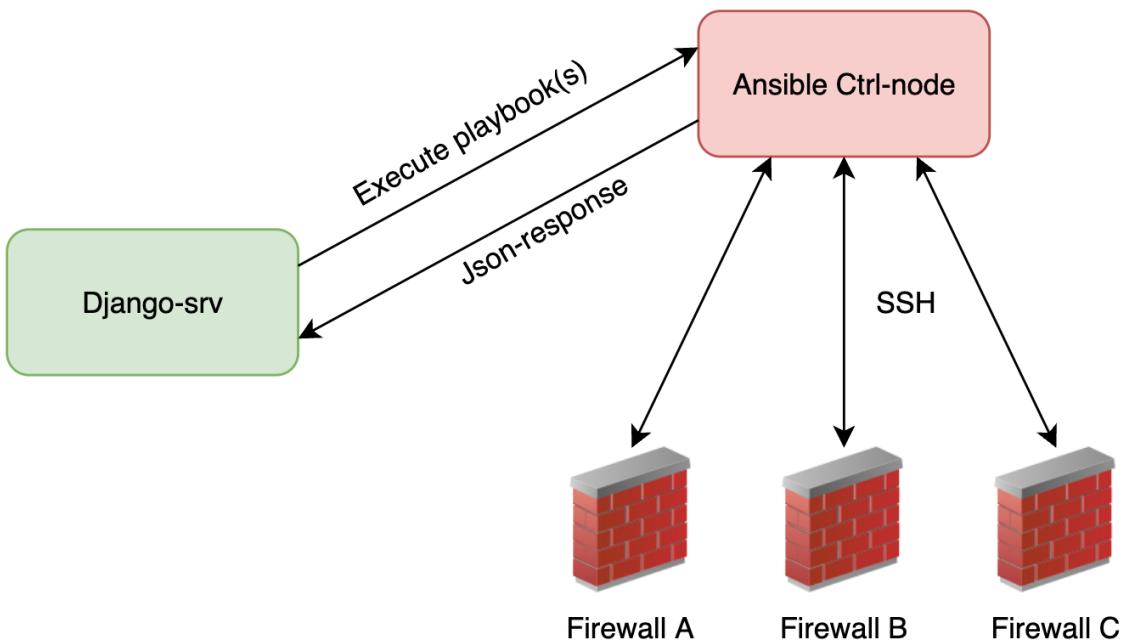
Nu er een beter zicht is gecreëerd over de functionaliteiten van Ansible, kan er ook hier met zekerheid gezegd dat Ansible een groter spectrum aan mogelijkheden naar boven brengt vergeleken met FortiManager. Ansible kan ingezet worden voor het configureren/onderhouden van virtuele en fysieke firewalls maar ook voor applicatie deployment.

Bij aanvang van het project zal Ansible niet meteen een rol spelen. De reden dat Ansible niet wordt gebruikt bij aanvang van het project is om complexiteit van het project te minimaliseren en ontwikkeling van de applicatie zelf te bevorderen. Indien het wordt gebruikt zal het eerder voor firewall deployment zijn dan voor CI/CD-toepassingen. Anderzijds kan Ansible ook worden ingezet voor snelle deployment van de infrastructuur zoals het creeren van een Django server waarop de applicatie draait. Hierdoor kan er snel op meerdere locaties een Django server worden opgezet waardoor in samenwerking met een load-balancer, redundantie wordt opgebouwd. Met behulp van een load-balancer is er steeds een Django server online indien een server faalt.

Deployment van het Django platform zal mogelijks via een ander CI/CD-automation-tool worden afgehandeld. Hoe dan ook Ansible is een waarde kandidaat voor eender welk project dat iemand wil gaan automatiseren. De mogelijkheden en ondersteuning van alle verschillende modules maken de beschikbare automatisatie opties eindeloos.

6 Ansible scenario

Ondanks dat Ansible net als FortiManager niet zullen gebruikt worden, is het opnieuw interessant en belangrijk om te weten wat het scenario zou zijn indien het wel gebruikt wordt. De verschillende werkingen en mogelijkheden kunnen altijd later eventueel worden gebruikt voor het uitwerken van dit, of een ander project. Vergelijkbaar met het FortiManager scenario (Figuur 11: FortiManager scenario (alternatief)) zal Ansible een gelijkaardige rol spelen (tussen pion). Het enige verschil met Ansible is dat in theorie, nu alle soorten toestellen geconfigureerd/beheerd kunnen worden. FortiManager kon dit ook, maar enkel voor Fortinet toestellen.



Figuur 14: Ansible scenario (alternatief)

In bovenstaande figuur (Figuur 14: Ansible scenario (alternatief)) is het alternatieve scenario met Ansible uitgetekend. Hier wordt duidelijk dat Django geen API-calls stuurt naar een FortiManager of aparte toestellen, maar wordt achterliggend via SSH verbonden met de Ansible-control node. De achterliggende SSH-connectie komt vanuit de Django "view", daar wordt een methode uitgevoerd die gebruik maakt van de module "[subprocess](#)". Dit is een Python module die commando's kan doorsturen naar externe machines. De Python functie waarin deze module "subprocess" wordt gebruikt kan dan Ansible playbook(s) aanroepen met eventueel toegevoegde variabelen. De火walls gedefinieerd in bovenstaande figuur kunnen virtuele, maar ook fysieke machines zijn.

Om een beeld te geven van hoe zo een Python functie met deze module er uitziet, is er een code voorbeeld opgemaakt (Codefragment 6: Voorbeeld Python functie met subprocess module). In het voorbeeld wordt er een ping uitgevoerd voor het controleren van actieve toestellen. Het scenario geeft Django-template waarbij de gebruiker moet kiezen uit een lijst van toestellen waar hij of zij een ping naar wil uitsturen. De beschikbare lijst met toestellen wordt aangeleverd bij het laden van de pagina en achterliggend door de functie "index" gedefinieerd in het "views.py" bestand.

```

from django.shortcuts import render
import subprocess
import json
from django.http import JsonResponse

def index(request):
    FirewallList = [
        ('FW1 - Fortinet os', 'FW1'),
        ('FW2 - Palo Alto os', 'FW2')
    ]
    return render(request, 'example_app/index.html',
                  {'FirewallList': FirewallList})

def ping_test(request):
    firewall = request.POST.get('fw_name')
    result = subprocess.Popen(['ansible-playbook', 'ping-playbook.yml',
                              '--extra-vars', 'host=' + firewall],
                             stdout=subprocess.PIPE)
    (output, err) = result.communicate()
    return JsonResponse(json.loads(output))

```

Codefragment 6: Voorbeeld Python functie met subprocess module [17]

De Python functie “ping_test” gedefinieerd in bovenstaand codefragment zal dus aan de hand van een module “subprocess”, Ansible commando’s gaan doorsturen naar een externe Ansible server. De zogezegde ping wordt dan verstuurd vanaf de Ansible server, waarna een response wordt teruggestuurd naar de Django server. Wanneer de Django server een response heeft ontvangen zal dit worden weergegeven aan de gebruiker.

7 API-calls

Bij aanvang en voor het uitwerken van dit project, zal gebruik gemaakt worden van API-calls voor het configureren van firewalls. Door gebruik te maken van API-calls kunnen toestellen van verschillende vendor ook worden aangesproken. Hiervoor moeten er weinig tot geen extra packages worden geïnstalleerd op de Django server om succesvolle API-requests werkende te krijgen. Aan de hand van Python-functies in de “view” kan het Django platform rechtstreeks API-calls uitsturen naar één of meerdere firewalls.

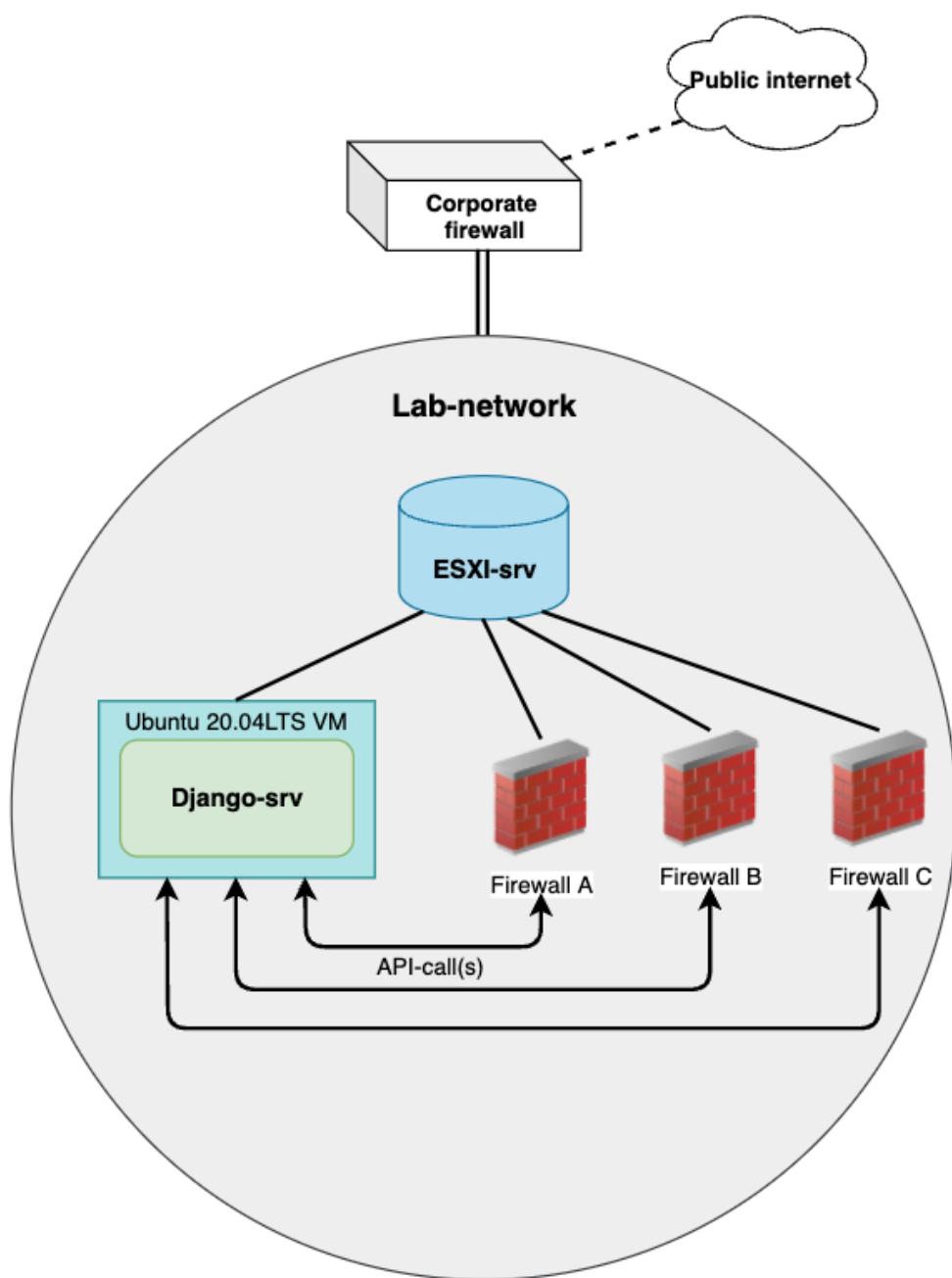
Fortimanager of Ansible worden niet verkozen voor de basisopstelling om twee eenvoudige redenen;

Zoals eerder vermeld is Fortimanager vendor specifiek, waardoor compatibiliteit met andere firewalls beperkt is of gewoonweg niet bestaat. Daarnaast beperkt het de gebruiker opnieuw toekomstgericht, wanneer er eventueel gewisseld wordt van vendor/leverancier.

Ansible is ook een uitstekende kandidaat, maar wordt niet in gebruik genomen gewoonweg omdat het de opstelling enorm complex zou maken, waardoor de tijd van ontwikkeling en ingebruikname van het platform enorm zou achteruitgeschoven worden. Eenmaal het platform operationeel is kan er eventueel worden uitgebreid met Fortimanager, Ansible of een ander alternatief om extra functionaliteit toe te voegen.

7.1 Opstelling

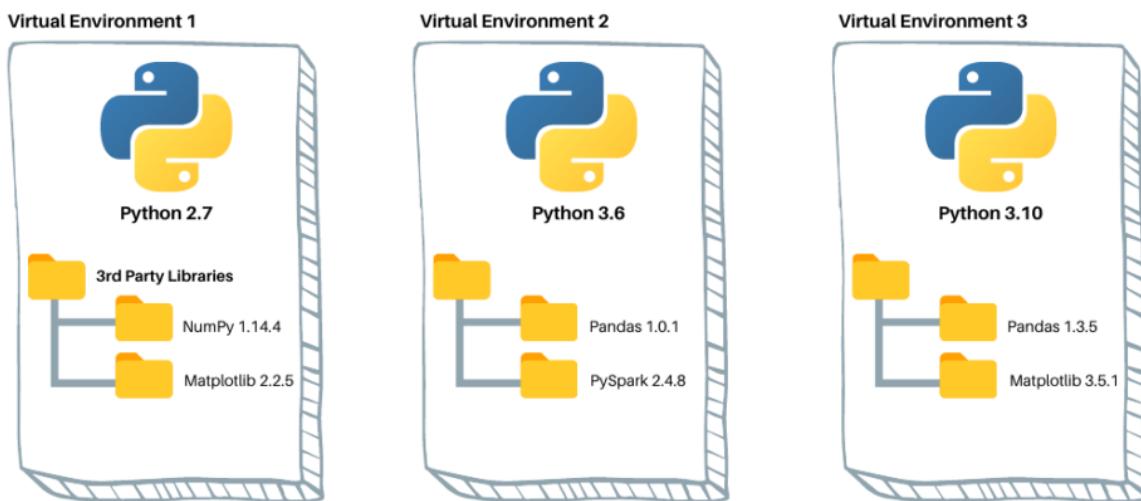
Onderstaande figuur (Figuur 15: Basisopstelling (binnen labo)) geeft een illustratie van de vereenvoudigde basis opstelling in het labo, voor ontwikkeling en testen van het Django platform. Vanop de ESXI-server worden minimum vier virtuele machines opgesteld, die machines bestaan uit een Ubuntu server en drie firewalls. De Ubuntu server zal het Django platform hosten en de firewalls met API-calls aanspreken. Zoals eerder vermeld (Figuur 3: Django platform pagina structuur) zal het aan de hand van API-calls mogelijk zijn om firewalls te updaten en configureren. De firewalls in onderstaande diagram kunnen fysieke maar ook virtuele firewalls zijn. Het toevoegen en testen van een fysieke firewall zal voor dit project afhankelijk zijn volgens beschikbaarheid (magazijn stock).



Figuur 15: Basisopstelling (binnen labo)

7.2 Python virtuele omgevingen

Voor ontwikkeling van het Django platform wordt gebruik gemaakt van een Python virtuele omgeving. In een volgend hoofdstuk wordt duidelijk waarom virtuele omgevingen handig kunnen zijn in een productieomgeving. Een virtuele Python omgeving bestaat uit twee componenten. De Python interpreter waarop de virtuele omgeving draait en een map met bibliotheken van derden die in de virtuele omgeving zijn geïnstalleerd. Deze virtuele omgevingen zijn gescheiden van de andere virtuele omgevingen, waardoor eventuele wijzigingen of geïnstalleerde packages aangebracht in die virtuele omgeving geen invloed hebben op de andere virtuele omgevingen of de systeembibliotheken. Er kunnen dus meerdere virtuele omgevingen worden gecreëerd met verschillende Python versies, bibliotheken of dezelfde bibliotheken in verschillende versies omdat elke virtuele omgeving onafhankelijk is van elkaar.



Figuur 16: Python virtuele omgevingen visueel voorbeeld [9]

In bovenstaande afbeelding (Figuur 16: Python virtuele omgevingen visueel voorbeeld) is een voorbeeld van wat er op een systeem staat wanneer meerdere virtuele Python-omgevingen zijn gemaakt. Eigenlijk is een virtuele omgeving een afgezonderde mappenboom die een specifieke Python-versie, bibliotheken en andere scripts bevat. Er zijn geen beperking op het aantal virtuele omgevingen op een systeem, omdat het slechts mappen zijn die enkele bestanden bevatten [9].

7.3 Voordelen Python virtuele omgevingen

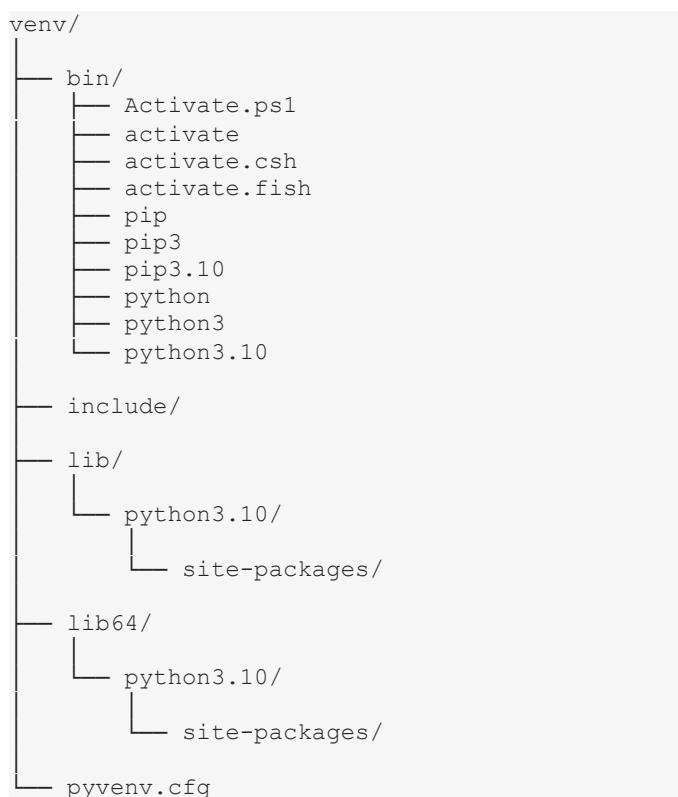
Het voordeel van virtuele Python omgevingen wordt duidelijk wanneer verschillende Python projecten op een dezelfde machine worden ontwikkeld. Sommige projecten zijn afhankelijk van verschillende versies of pakketten. Bijvoorbeeld; een ontwikkelaar werkt aan twee verschillende Django applicaties, het eerste project maakt gebruik van een ander front-end framework pakket dan het tweede project. Daarnaast zal het eerste project met Python versie 2.X en de andere met versie 3.X worden uitgerold. Dit zou leiden tot compatibiliteitsproblemen omdat Python niet tegelijkertijd meerdere versies van hetzelfde pakket kan gebruiken. Het andere voordeel dat het gebruik van virtuele Python omgevingen vergroot, is wanneer er wordt gewerkt op beheerde servers of productieomgevingen waar algemene systeem pakketten worden vereist en niet kunnen worden gewijzigd.

Virtuele Python omgevingen creëren afgezonderde contexten, om afhankelijkheden die nodig zijn voor verschillende projecten gescheiden te houden. Op die manier worden conflicten met andere projecten of systeem pakketten gecreëerd. Het komt erop neer dat het opzetten van virtuele omgevingen de beste manier is om verschillende Python projecten te isoleren. Vooral wanneer deze projecten verschillende en tegenstrijdige afhankelijkheden hebben. Het is aangeraden aan elke ontwikkelaar om altijd een virtuele omgeving te creëren voor elk Python project, waarbij installatie van alle vereiste pakketten daarin terechtkomen [9].

Wanneer een project van machine moet migreren of dergelijke situatie is het minder lastig om de nodige pakketten over te zetten, dan een machine te klonen of pakketten individueel over zetten. In dit project worden ook virtuele omgevingen gebruikt, op die manier kan de Django omgeving en zijn packages later eventueel worden uitgerold aan de hand van Docker containers.

Het maken van een virtuele omgeving gebeurt met behulp van de module “venv”. Hierdoor wordt een folderstructuur nagemaakt van een standaard Python installatie op een systeem. Tegelijk worden symlinks gekopieerd in de folderstructuur naar de Python executable.

De folderstructuur van een virtuele omgeving wordt in onderstaand fragment weergegeven (**Error! Reference source not found.**).



Codefragment 7: Folder structuur virtuele omgeving

8 Continuous delivery

Wanneer een product in productie wordt gebracht is het belangrijk dat deze snel en flexibel uitgerold kunnen worden. Daarom wordt er gebruik gemaakt van Jenkins en virtuele omgevingen. Jenkins zorgt er voor dat updates meteen doorgerold kunnen worden naar de productieomgeving. De virtuele omgevingen maken het mogelijk om producten/applicaties met zekerheid op verschillende machines of sites operationeel te maken. Dit is door het "requirements.txt" bestand dat vaak standaard bij elk project wordt meegeleverd, waardoor de nodige packages meteen zijn geïnstalleerd op de omgeving en de applicatie geen compatibiliteitsproblemen of dergelijke heeft bij opstart. De virtuele omgeving en installatie van de nodige packages in het "requirements.txt" bestand kan worden opgezet met behulp van Jenkins. In volgend subhoofdstuk wordt besproken wat Jenkins precies is en waarvoor het kan gebruikt worden.

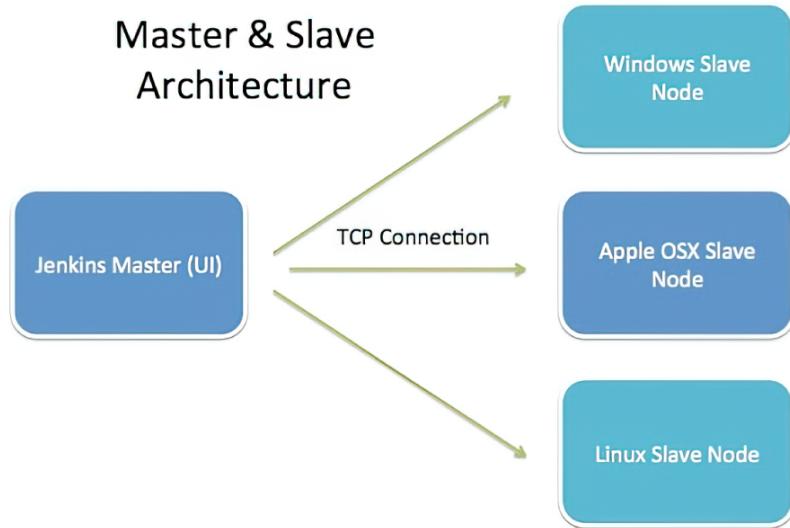
8.1 Jenkins

Het enige constante in technologie is verandering. Het hele proces begint opnieuw zodra er een release, nieuwe functie of update is waarna veranderingen uiteindelijk in productie worden gesteld. Daarom is er zoiets als CI/CD. Als er wordt gedacht om CI/CD te implementeren zijn er meerdere oplossingen die dit kunnen bieden. Daaronder vallen automatisatie tools zoals Jenkins, GitLab CI/CD, Ansible, TeamCity, etc. Het doel achter continuous delivery is dat projecten en ideeën uiteindelijk bruikbaar blijven voor de eindgebruiker. Wanneer er een update gebeurt is het dus de bedoeling dat de gebruiker deze niet merkt aan de beschikbaarheid van het product.

Opnieuw maakt het externe gebruikers niet uit hoe een product of idee werd afgeleverd, zolang ze eenmaal toegang hebben tot de functie of applicatie. Voor het interne team zoals een DevOps-engineer, is het wel belangrijk hoe een product wordt geleverd, omdat het proces van functielevering rechtstreeks invloed heeft op het team. Een product dat niet op de juiste manier wordt afgeleverd, kan later voor heel wat zorgen en problemen zorgen die kunnen worden vermeden, daarom maakt men best gebruik van CI/CD.

Zoals eerder vermeld is Ansible ook in staat om CI/CD te voorzien. Hoe dan ook is dit niet het kerndoel van Ansible, daarom zal gebruik gemaakt worden van Jenkins. Enige tekortkomingen of andere zaken dat Jenkins en/of Ansible hebben, kunnen worden opgevangen door ze eventueel te laten samenwerken. De samenwerking van deze twee technologieën is een vaak voorkomend fenomeen.

Jenkins is een open source platform en heeft net als Django een sterke gemeenschap achter zich staan, voor enige bijstand die een ontwikkelaar of DevOps-engineer zou nodig hebben kan worden gerekend op publieke forums. Opnieuw omdat het een zeer populair open source engine is, worden steeds meer en betere functies naar Jenkins gebracht na controle van meerdere ontwikkelaars, waardoor ook de nieuwste veiligheidsstandaarden worden toegepast. Omdat Jenkins een gedistribueerde werking heeft is het voldoende om slechts één server te voorzien, zelfs voor complexe projecten.

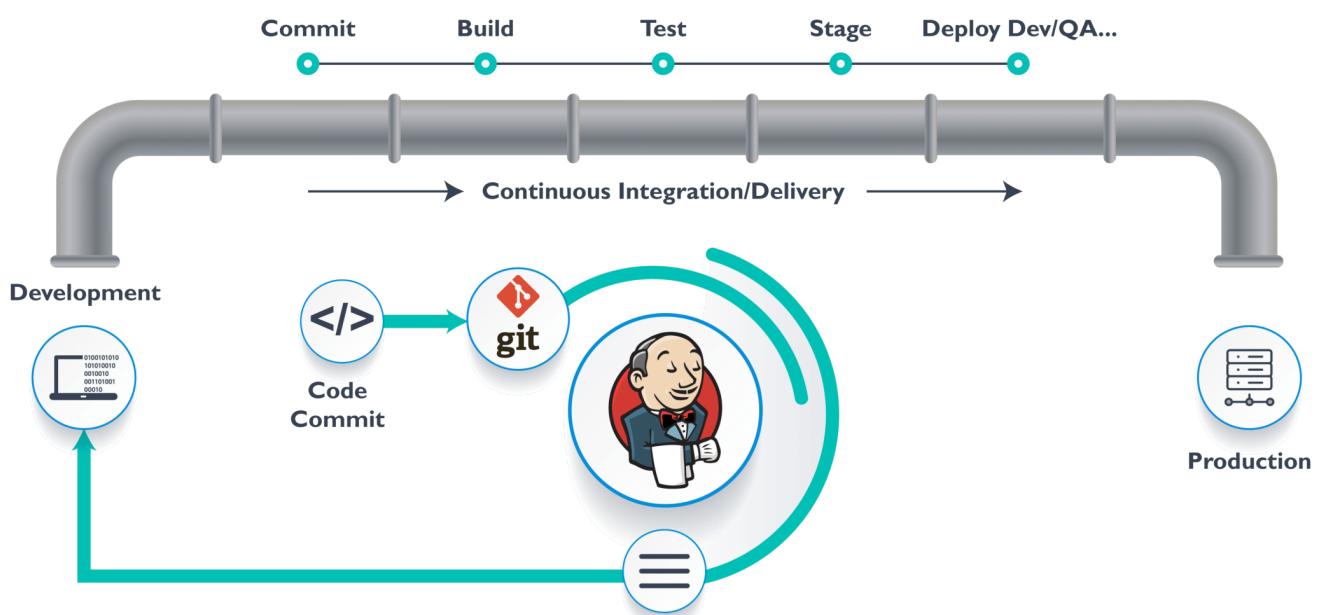


Figuur 17: Jenkins master-slave architectuur [15]

Wanneer iemand bijvoorbeeld de compatibiliteit van een applicatie wil testen, kan Jenkins gebruikmaken van de Master-Slave architectuur. Daarbij beschikken de "slaves" (volgens configuratie) over een ander besturingssysteem.

De Jenkins server werkt dan als "Master". Afhankelijk van hoe Jenkins is geconfigureerd is het mogelijk om op deze "slave" servers meerdere build/test omgevingen tegelijkertijd te laten lopen (Figuur 17: Jenkins master-slave architectuur) [16].

De reden achter het gebruik van Jenkins in plaats van de ingebouwde CI/CD-service van GitLab, is dat er later nog steeds zonder problemen kan worden gewisseld van "Distributed Version Control System" (DVCS) service of platform. De enige aanpassingen die moeten gebeuren in Jenkins om pipeline(s) operationeel/actief te houden is de repository bron URL-aanpassen.



Figuur 18: Jenkins visuele werking [23]

8.2 Jenkins minimumvoorwaarden

De Jenkins minimumvoorwaarden zijn belangrijk voor het opstellen en configureren van de virtuele machine in de labo-omgeving bij aanvang van het project. Wanneer de volledige configuratie uiteindelijk wordt opgesteld op andere locaties, moeten de machines ook capabel zijn om Jenkins te ondersteunen.

Voor Jenkins operationeel kan zijn op een machine zijn er een aantal minimumvoorwaarden die moeten voldoen. Onder deze voorwaarden vallen dan vooral hardware maar ook software voorwaarden. Een belangrijke factor op vlak van hardware is het aantal beschikbare tijdelijk geheugen (RAM) en harde schijf opslagcapaciteit in Gigabytes (GB).

Type hardware	Minimumvoorwaarde
RAM	256MB
Schijf opslag	1GB (10GB bij gebruik docker container)

Tabel 1: Jenkins minimum hardware voorwaarden

De hardware voorwaarden van bovenstaande tabel (**Error! Reference source not found.**) zijn vaak niet genoeg wanneer meerdere jobs tegelijk moeten kunnen draaien. Ook wanneer wordt gewerkt met een team van mensen zijn bovenstaande minimumaantallen niet voldoende. Voor een klein team wordt al sneller één Gigabyte aan RAM en vijftig Gigabytes aan opslag aangeraden. Hoe dan ook verschillen de minimumvoorwaarden van team tot team en van project tot project.

Naast hardware voorwaarden is het software gedeelte waarbij wordt verwacht dat een aantal packages zijn geïnstalleerd en compatibel zijn met het systeem waarop ze worden geïnstalleerd. In onderstaande tabel (Tabel 2: Jenkins minimumsoftware voorwaarden) is een oplijsting van de verschillende minimumvoorwaarden op valk van software [17].

Software	Beschrijving
Java	Java Development Kit (JDK) of (JRE)
Webbrowser (Enkel voor systeem met GUI)	Google chrome, Firefox, Safari, etc.
Besturingssysteem	Windows, macOS, Linux (Ubuntu, etc)

Tabel 2: Jenkins minimumsoftware voorwaarden [17]

In bovenstaande tabel kan iedereen de nodige software/packages raadplegen wanneer er van Jenkins wordt gebruik gemaakt. Omdat niet alle machines een GUI ondersteunen is een webbrowser niet echt een vereiste. Zolang de eindgebruiker de Jenkins interface maar kan raadplegen is er geen probleem. Er kan bijvoorbeeld aan de hand van een andere (externe) machine met een webbrowser naar het IP-adres van de Jenkins server worden gegaan.

8.3 Jenkins best practices

De gebruikersinterface van Jenkins is niet zo heel gebruiksvriendelijk als sommige moderne alternatieven. Sommige geavanceerdere configuraties kunnen lastig zijn om in te configureren voor DevOps-engineers zonder veel ervaring. Hoewel de community support sterk is om hierbij te helpen. In eender welke situatie is er zeker en vast een plug-in ontwikkeld die kan helpen.

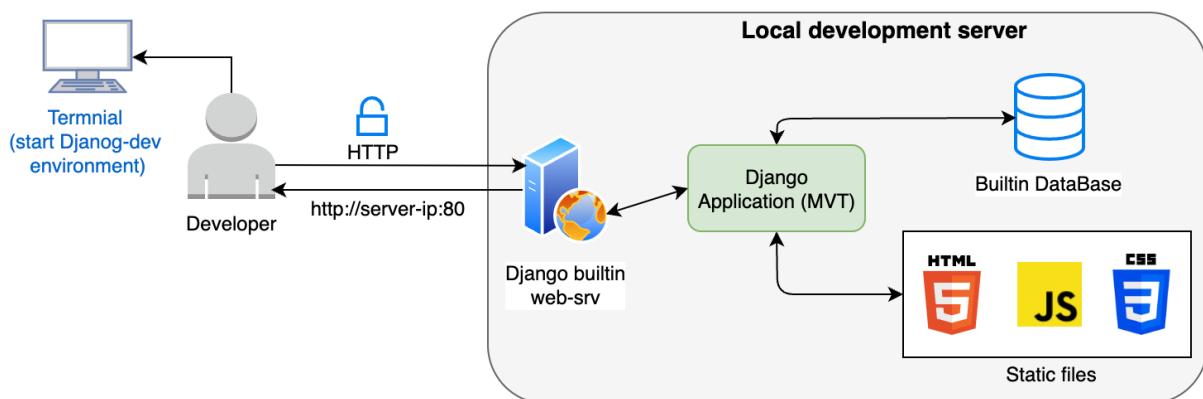
Om ervoor te zorgen de engineer het beste uit Jenkins haalt, zijn er toch een aantal zaken die de engineer best opvolgt;

- Houd het veilig: zoals bij elke belangrijke softwaretool, is het van belang om Jenkins-server(s) veilig te houden met robuuste gebruikersauthenticatieprocessen. Daarnaast is een back-up van de map 'Jenkins Home' steeds aangeraden, deze bevat alle configuratie- en loggegevens die een engineer zeker niet wil verliezen.
- Wees selectief met plug-ins: het aantal beschikbare plugins is groot, daarom kan het verleidelijk zijn om er te veel te proberen. Dit zou processen kunnen vertragen, waardoor het hoofddoel van een snelle levering teniet wordt gedaan.
- Vermijd complexe Groovy-code in pijplijnen: Groovy-code wordt altijd uitgevoerd op de master/controller, dus als het te ingewikkeld is, zal dit een aanslag zijn op de computerbronnen. Onthoud dat de pipeline is ontworpen om het team te helpen, sneller te bouwen en dus het ontwerp best zo eenvoudig mogelijk houdt [16].

9 Hosting

Het hosten van een applicatie in een productieomgeving is heel verschillend van een ontwikkelingsomgeving. Dit gaat zowel over de projectstructuur als de services die beschikbaar worden gemaakt naar de eindgebruiker(s) toe. Bij het ontwikkelen van een Django applicatie kan er lokaal een ontwikkelings-server worden geactiveerd. Op die manier kan de ontwikkelaar zijn of haar gecodeerde resultaat gaan bekijken en testen. Om deze ontwikkelings-server op te starten moeten er steeds een reeks aan commando's worden uitgevoerd, daarnaast is het ook zo dat als de ontwikkelaar zijn of haar terminal afsluit de server ook afsluit. Dit soort hosting is puur en enkel voor ontwikkeling en zou zeer volatiel zijn wanneer een applicatie op die manier in productie wordt aangeboden.

Onderstaande figuur (Figuur 19: Werking Django development hosting) geeft een beeld van hoe Django applicatie(s) lokaal worden beschikbaar gemaakt voor ontwikkeling. De afbeelding maakt duidelijk dat de ontwikkelaar eerst een terminal moet openen om



Figuur 19: Werking Django development hosting

daar de Django ingebouwde web-server te activeren. Eenmaal de lokale web-server is opgestart kan de ontwikkelaar via het IP-adres van de server onbeveiligd (met HTTP) naar de web-server surfen.

Een opstelling als deze in productieomgevingen is onrealistisch aangezien het niet schaalbaar, beveiligd en consistent beschikbaar is. Daarom is er een productie opstelling gecreëerd, door samenwerking van verschillende services zal het platform uiteindelijk consistent beschikbaar blijven voor de eindgebruiker.

Wanneer terug wordt gekeken naar voorgaande figuur (Figuur 15: Basisopstelling (binnen labo)) is er een Ubuntu server waarop de Django applicatie beschikbaar wordt gemaakt. Op deze server zijn er een aantal services extra aangemaakt voor in de productie. Onderstaande lijst geeft weer welke services aanwezig zijn op deze server;

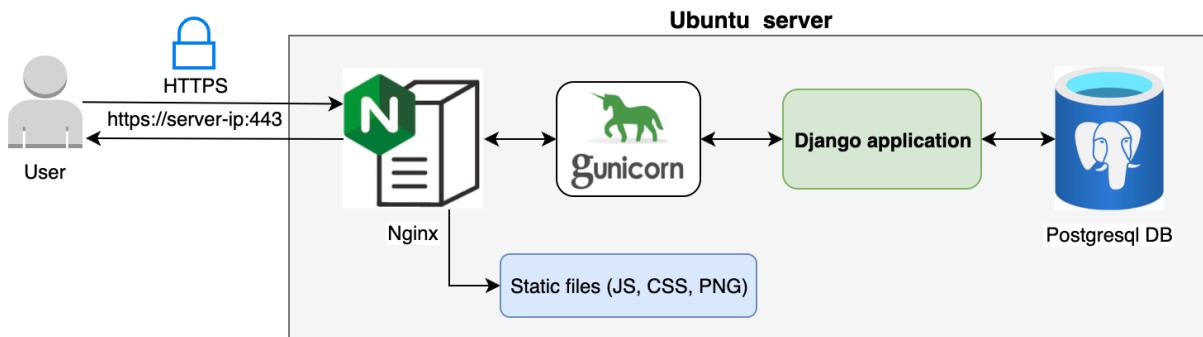
- Django
- [Gunicorn](#)
- [Nginx](#)
- [Postgresql](#)

Omdat het een Django applicatie is dat wordt gehost, is het vrij logisch dat Django op de server aanwezig is. Hoe dan ook, het actief beschikbaar maken van de applicatie gebeurt door Gunicorn en Nginx. Beide technologieën spelen een grote rol en gaan hand in hand voor deze situatie. Gunicorn neemt de taak van het standaard meegeleverde Django WSGI-service over en zal de applicatie ten alle tijden draaiende houden. Kortgezegd is de Gunicorn een deployment platform voor Python webservers en applicaties [22]. Het enige verschil is dat de server minder wordt belast met Gunicorn dan met de standaard WSGI-service omdat het meer geoptimaliseerd is voor Django. Gunicorn wordt met een specifiek commando gekoppeld aan de applicatie-WSGI;

```
gunicorn --bind 0.0.0.0:8080 application.wsgi
```

Codefragment 8: Voorbeeld Gunicorn "bind" commando

Naast Gunicorn is Nginx, die speelt als webserver/proxy en verwijst requests door naar de juiste (Django) applicatie afhankelijk van configuratie. Om modellen te maken en data van de Django applicatie weg te schrijven is er een database vereist. De database die daarvoor gebruikt wordt is een Postgres database, gecreëerd op dezelfde machine. Vaak wordt een database op een aparte machine gehost voor redundantie om geen "single-point of failure" te creëren. Maar om de complexiteit en configuratie tijd te besparen, wordt de database bij uitwerken van dit project niet extern. Alsook wordt de latency tussen de applicatie zelf en database verlaagd, wat algemene snelheid van het platform zou moeten verbeteren.



Figuur 20: Werking services voor hosting in productie

Voorgaande afbeelding (Figuur 20: Werking services voor hosting in productie) is een visuele weergave van de services op een Ubuntu machine. Hierdoor wordt duidelijk welke services een request van de gebruiker verwerken. De figuur toont aan dat wanneer de gebruiker naar het IP-adres van de server surft een request wordt verwerkt door Nginx. Nginx stuurt op zijn beurt de request door naar Gunicorn, waarna de Django applicatie een request binnenkrijgt. De request activeert een functie in de view van Django MVT-model en zal bij gevolg een Django template updaten om aan de gebruiker weer te geven. Statische bestanden en CSS-eigenschappen worden gedefinieerd in map "static" van Django applicatie. De map "static" wordt ook naar verwezen in de Nginx configuratie, zodanig dat Nginx deze kan weergeven naar de gebruiker. Een voorbeeld van de Nginx configuratie voor dit platform kan in onderstaand codefragment worden geraadpleegd.

```
server {
    listen 443;
    server_name 10.0.89.147;

    location = /favicon.ico { access_log off; log_not_found off; }
    location /static/ {
        root /home/deployment-user/automationplatform/siteplatform;
    }

    location / {
        include proxy_params;
        proxy_pass http://unix:/run/gunicorn.sock;
    }
}
```

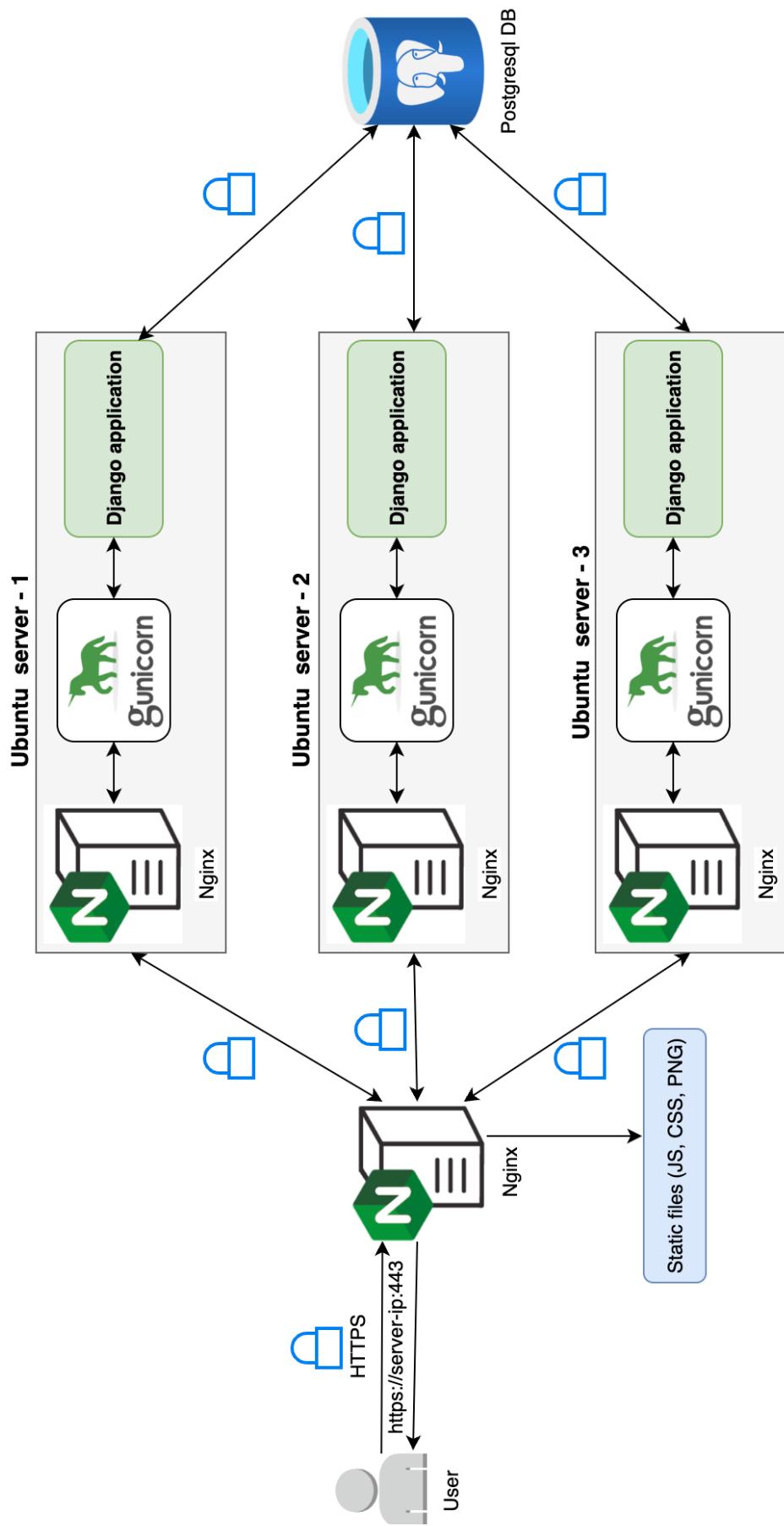
Codefragment 9: Nginx "sites-available" applicatie config voorbeeld

Bovenstaande configuratie zorgt er ook voor dat Django zelf niet meer zal proberen om een web interface weer te geven, maar wordt deze taak doorgegeven aan Nginx en Gunicorn. Als er later een request wordt gestuurd naar de app-server, zal op poort 443 de applicatie beschikbaar zijn. Achterliggend zijn nog heel wat meer configuraties nodig voor Gunicorn maar deze worden niet in dit verslag gedocumenteerd om herhaaldeleijke en grote blokken code te vermeiden. Voor meer configuratie details kan het logboek worden geraadpleegd via [deze link](#) of de referentie in de bijlagen (bijlage 6).

9.1 Ideale productie opstelling

Wanneer de last voor één applicatie-server te groot wordt door de vele requests, kan deze last tussen meerdere applicatie-servers worden verdeeld. Dit gebeurt aan de hand van load-balancing. Opnieuw kan Nginx hier een oplossing bieden. De onderstaande figuur (Figuur 21: Productie opstelling in "real-world" scenario) beschikt over een centrale Nginx-server, deze heeft drie belangrijke taken te vervullen.

Allereerst worden bestanden uit de "static" folder van de applicatie nu geserveerd door de centrale Nginx server. Hierdoor is het een soort webserver op zich, daarnaast zal deze Nginx-server ook spelen als een proxy, omdat inkomende requests zullen doorgestuurd worden naar de achterliggende applicatie-servers. Als laatst is de centrale Nginx-server ook een soort load-balancer, omdat afhankelijk van het aantal requests de server zal bepalen naar welke backend-server een request wordt doorgestuurd om te gaan afhandelen. Deze opstelling brengt vooral redundantie met zicht mee. Wanneer nu een server niet actief is, zal dat voor de eindgebruiker zo goed als niet merkbaar zijn omdat de applicatie nog steeds beschikbaar is op de andere backend servers. Soortelijke opstellingen zijn ideaal voor een productieomgeving. Vaak kan men gehoste applicaties op deze of gelijkaardige manier terugvinden in de algemene industrie.

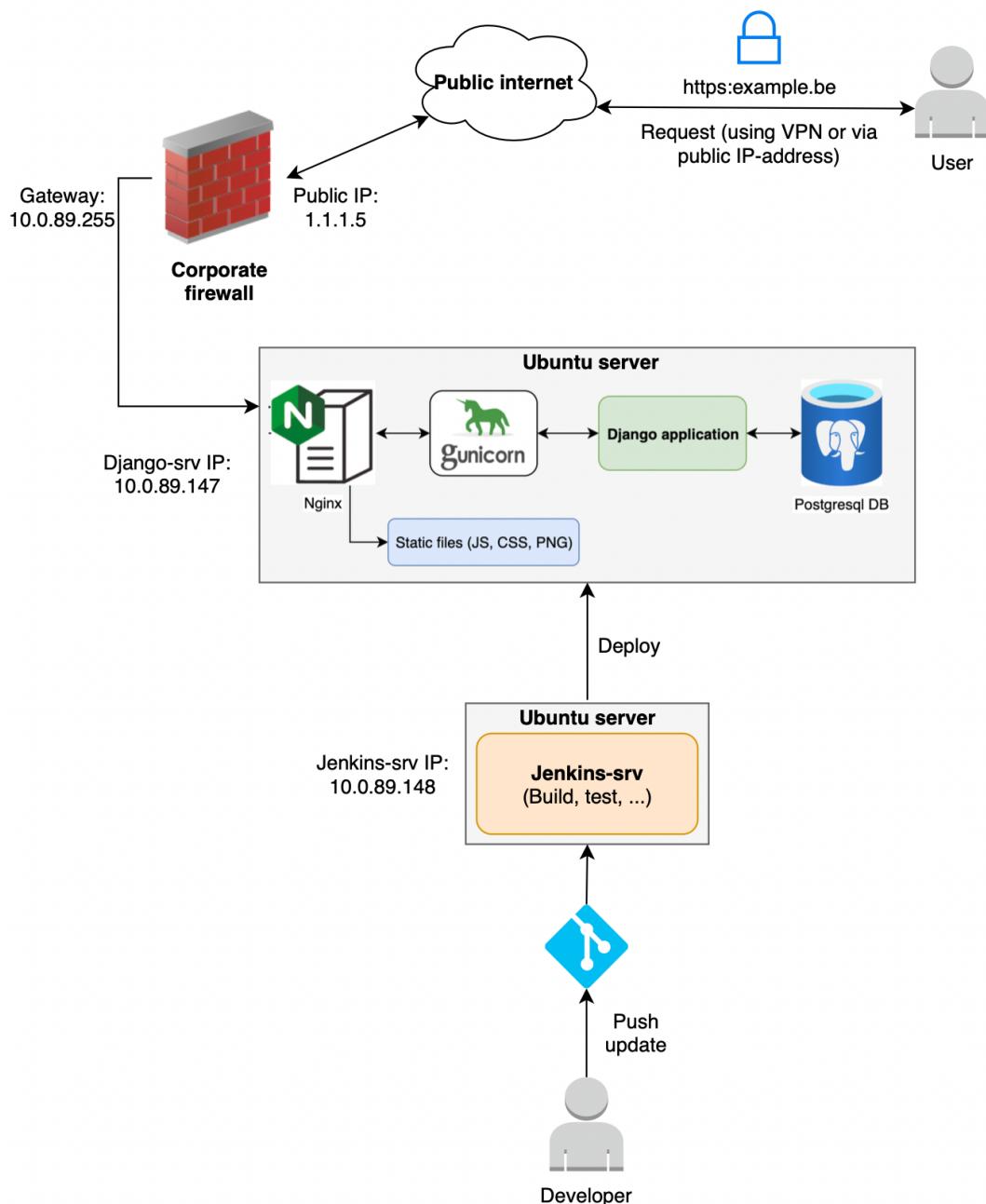


Figuur 21: Productie opstelling in "real-world" scenario

9.2 Volledige projectstructuur

In voorgaand subhoofdstuk werd duidelijk wat de ideale opstelling is voor het hosten van applicatie(s) in een productieomgeving.

Voor dit project wordt het ideale scenario ook nagestreefd. Enkel de redundante servers worden niet opgezet om complexiteit te reduceren. In onderstaande figuur wordt duidelijk dat Jenkins alle ontwikkelaars de mogelijkheid geeft om de applicatie steeds te verbeteren aan de hand van de vooropgestelde CI/CD-configuratie. De applicatie zelf wordt beschikbaar op een Ubuntu machine en dit wordt mogelijk gemaakt door de voorgaand vermelde services (Nginx, Gunicorn, etc). Uiteindelijk zal de applicatie bereikbaar zijn via een VPN, indien het enkel voor werknemers bereikbaar mag zijn. Later kan dit ook steeds publiek bereikbaar gemaakt worden door de juiste NAT-regels toe te passen op de corporate firewall. Wanneer de applicatie publiek bereikbaar is kan het IP-adres met bijhorende poort aan een DNS-record worden gelinkt. Op die manier is het platform vlot bereikbaar.



Figuur 22: Overzicht volledige infrastructuur opstelling

Conclusie

Uit de studie is duidelijk wat de beperkingen zijn van het handmatig configureren van netwerk toestellen. In dit geval is dat vooral tijdverlies. Ook is een zicht gecreëerd van de mogelijk alternatieve technologieën die gebruikt kunnen worden als fundamentele bouwsteen voor het opbouwen van dergelijk project. De alternatieve technologieën kunnen uiteindelijk steeds worden gebruikt als uitbreiding van het platform. Op die manier kan de schaal voor het uitrollen van firewalls eventueel worden vergroot, of de compatibiliteit verschillen tussen firewall vendors worden verkleind.

Omdat het platform nog niet beschikbaar is gemaakt naar engineers binnen het bedrijf en/of eindgebruikers, is feedback beperkt tot onbestaand. Door die reden is het nog niet duidelijk wat de eventuele werkpunten zijn om het platform te optimaliseren en bij te schaven.

De kennis die werd verworven tijdens het onderzoek voor dit project kan eventueel in toekomstige projecten gunstig zijn en tijd besparen. Vooral in de ontwikkelingsfase en uitwerken van een projectstructuur kan dit een voordelig zijn.

Op lange termijn is regelmatige feedback van engineers en reviews van gemachtigde externen zeer waardevol. Zo kan het platform steeds worden verbeterd en geoptimaliseerd. Dit zorgt ervoor dat het platform uiteindelijk optimaal presteert en eventueel meer eenvoud of functionaliteit biedt dan origineel werd voorgesteld.

Nawoord

Het project op zich was zeer leerrijk en interessant, waarbij de onderzoeksraag begon met een relatieve scope van vendorspecifieke toestellen tot multivendor compatibiliteit. Aangezien het project op praktisch vlak nog niet is vervolledigd, blijft het lastig om te reflecteren op bevindingen uit de praktijk. Hoe dan ook is de theoretische uitwerking één van de belangrijkste en fundamentele delen voor het uitwerken van eender welk project. Het platform en software defined networking oplossingen als deze creëren innovatie en vernieuwing naar de wereld en technische-werkvloer.

Mijn kennis voor het hosten van applicaties en van de MVT/MVC-architectuur is vergroot door het Django framework te leren kennen. Daarnaast heeft het framework en project op zich mij meer inzicht gegeven in creëren en hosten van webapplicaties. Maar ook persoonlijk geïnspireerd om meer soortelijke projecten te gaan onderzoeken en uitwerken.

Graag wil ik daarom mijn stagegever nogmaals bedanken voor de opportunitet van een stageplaats en dit project.

Literatuurlijst

- [JavaPoint, „Django history - javapoint,” [Online]. Available:
1 <https://www.javatpoint.com/django-tutorial>. [Geopend 22 03 2022].
]
- [Django docs (community), „FAQ Django docs - History and pronounce,” Django community, [Online]. Available:
2 <https://docs.djangoproject.com/en/4.0/faq/general/#:~:text=Django%20is%20pronounced%20JANG%2Doh,audio%20clip%20of%20the%20pronunciation..> [Geopend 17 05 2022].
- [I. Education, „What is Django - IBM,” 29 03 2021. [Online]. Available:
3 <https://www.ibm.com/cloud/learn/django-explained>. [Geopend 21 03 2022].
]
- [„Full Stack Python - Django framework,” [Online]. Available:
4 <https://www.fullstackpython.com/django.html>. [Geopend 21 05 2022].
]
- [Javapoint, „Django MVT - javatpoint,” [Online]. Available:
5 <https://www.javatpoint.com/django-mvt>. [Geopend 17 05 2022].
]
- [GeeksforGeeks, „GeeksforGeeks - Django Models,” 11 04 2022. [Online]. Available:
6 <https://www.geeksforgeeks.org/django-models/>. [Geopend 22 05 2022].
]
- [GeeksforGeeks, „GeeksforGeeks - Views In Django,” 16 09 2021. [Online]. Available:
7 <https://www.geeksforgeeks.org/views-in-django-python/>. [Geopend 22 05 2022].
]
- [GeeksforGeeks, „GeeksforGeeks - Django Templates,” 27 09 2021. [Online]. Available:
8 Available: <https://www.geeksforgeeks.org/django-templates/>. [Geopend 23 05 2022].
]
- [„Django introduction,” MDN contributors, 18 02 2022. [Online]. Available:
9 <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>.
] [Geopend 20 03 2022].
- [J. Patel, „Django Alternatives: Top 5 Python Frameworks To Use in 2022,” 02 03 2022. [Online]. Available: <https://www.monocubed.com/blog/django-alternatives/>.
0 [Geopend 22 03 2022].
]
- [J. Torian, „Zero(ish) Touch Provisioning with FortiManager – Explained,” 04 02 2019.
1 [Online]. Available: <https://www.historiantech.com/zeroish-touch-provisioning-with-fortimanager-explained/>. [Geopend 01 05 2022].
]
- [R.H Ansible, „About Ansible - Ansible documentation,” [Online]. Available:
1 <https://docs.ansible.com/ansible/latest/index.html>. [Geopend 03 11 2021].

2
]

[R.H Ansible, „Red Hat Ansible use case - Provisioning,” [Online]. Available:
1 <https://www.ansible.com/use-cases/provisioning>. [Geopend 03 11 2021].

3
]

[R.H. Ansible, „Ansible for Configuration management,” [Online]. Available:
1 <https://www.ansible.com/use-cases/configuration-management>. [Geopend 03 11
4 2021].

]

[R.H. Ansible, „Ansible for application deployment,” [Online]. Available:
1 <https://www.ansible.com/use-cases/application-deployment>. [Geopend 03 11 2021].
5
]

[R.H. Ansible, „Ansible For Continuous Delivery,” [Online]. Available:
1 <https://www.ansible.com/use-cases/continuous-delivery>. [Geopend 03 11 2021].
6
]

[M. Ahmed, „Medium - Ansible for Networking with Django-based GUI,” 12 12 2021.
1 [Online]. Available: <https://medium.com/@mahmoud.maghni8/ansible-for-networking-with-django-based-gui-simple-use-case-d558d3ec99b6>. [Geopend 27 05
] 2022].

[M. Lotfinejad, „Dataquest - A Complete Guide to Python Virtual Environments,” 09 03
1 2022. [Online]. Available: <https://www.dataquest.io/blog/a-complete-guide-to-python-virtual-environments/>. [Geopend 14 05 2022].
]

[J. McAllister, „packtpub - Mastering Jenkins,” [Online]. Available:
1 <https://subscription.packtpub.com/book/application-testing/9781784390891/2/ch02lvl1sec15/understanding-the-master-and-slave-architecture>. [Geopend 15 05 2022].

[S. Mino, „Jobsity - What is Jenkins and Why Should You Use It?,” [Online]. Available:
2 <https://www.jobsity.com/blog/what-is-jenkins-and-why-should-you-use-it>. [Geopend
0 v 05 2022].
]

[J., „TOOLSQA - Jenkins What are the pre- requisites and procedure,” 14 05 2022.
2 [Online]. Available: <https://www.toolsqa.com/jenkins/install-jenkins/>. [Geopend 14
1 05 2022].
]

[„Django documentation - How to deploy with WSGI,” [Online]. Available:
2 <https://docs.djangoproject.com/en/4.0/howto/deployment/wsgi/#:~:text=Django's%20primary%20deployment%20platform%20is,compliant%20application%20server%20to%20use..> [Geopend 25 05 2022].

- [A. Technologies, „Django timeline – Skillsets, Budget and Benefits,” 05 01 2021.
- 2 [Online]. Available: <https://www.agiratech.com/django-web-development-skillsets-budget-and-benefits>. [Geopend 21 03 2022].
-]

- [A. Martin, „Django logo,” 03 08 2021. [Online]. Available: <https://njkhanh.com/flask-django-or-pyramid-choose-the-right-python-framework-for-your-project-p5f33363134>. [Geopend 24 03 2022].
-]

- [Wikipedia contributors, „Django (web framework),” 05 04 2022. [Online]. Available: [https://en.wikipedia.org/wiki/Django_\(web_framework\)](https://en.wikipedia.org/wiki/Django_(web_framework)). [Geopend 13 04 2022].
- 5
-]

- [„Web2py Python Framework (Youtube),” 01 08 2014. [Online]. Available: <https://www.youtube.com/watch?v=yKQNmutC0WY>. [Geopend 28 04 2022].
- 6
-]

- [P. ReviewDesk, „CherryPy - Python Web Framework,” 13 04 2022. [Online]. Available: <https://www.predictiveanalyticstoday.com/cherrypy/>. [Geopend 28 04 2022].
- 7
-]

- [W. Semik, „Flask vs. Django - STXNEXT,” 29 11 2021. [Online]. Available: <https://www.stxnext.com/blog/flask-vs-django-comparison/>. [Geopend 10 05 2022].
- 8
-]

- [„Web2Py pros and cons - Choosing a better framework,” [Online]. Available: https://www.tutorialspoint.com/python_web_development_libraries/python_web_development_libraries_choosing_a_better_framework.htm. [Geopend 10 05 2022].
- 9
-]

- [„Compare CherryPy vs web2py,” 13 02 2021. [Online]. Available: <https://codeahoy.com/compare/cherrypy-vs-web2py>. [Geopend 10 05 2022].
- 0
-]

- [„Edureka - Afbeelding Jenkins werking,” [Online]. Available: <https://www.edureka.co/blog/content/ver.1531719070/uploads/2018/07/Asset-36-11.png>. [Geopend 15 05 2022].
- 1
-]

- [„Python 3.10.4 documentation - subprocess,” [Online]. Available: <https://docs.python.org/3/library/subprocess.html>. [Geopend 24 05 2022].
- 2
-]

Bijlagenoverzicht

Bijlage 1: Platform flowchart

Bijlage 2: FortiManager alternatief scenario

Bijlage 3: Ansible alternatief scenario

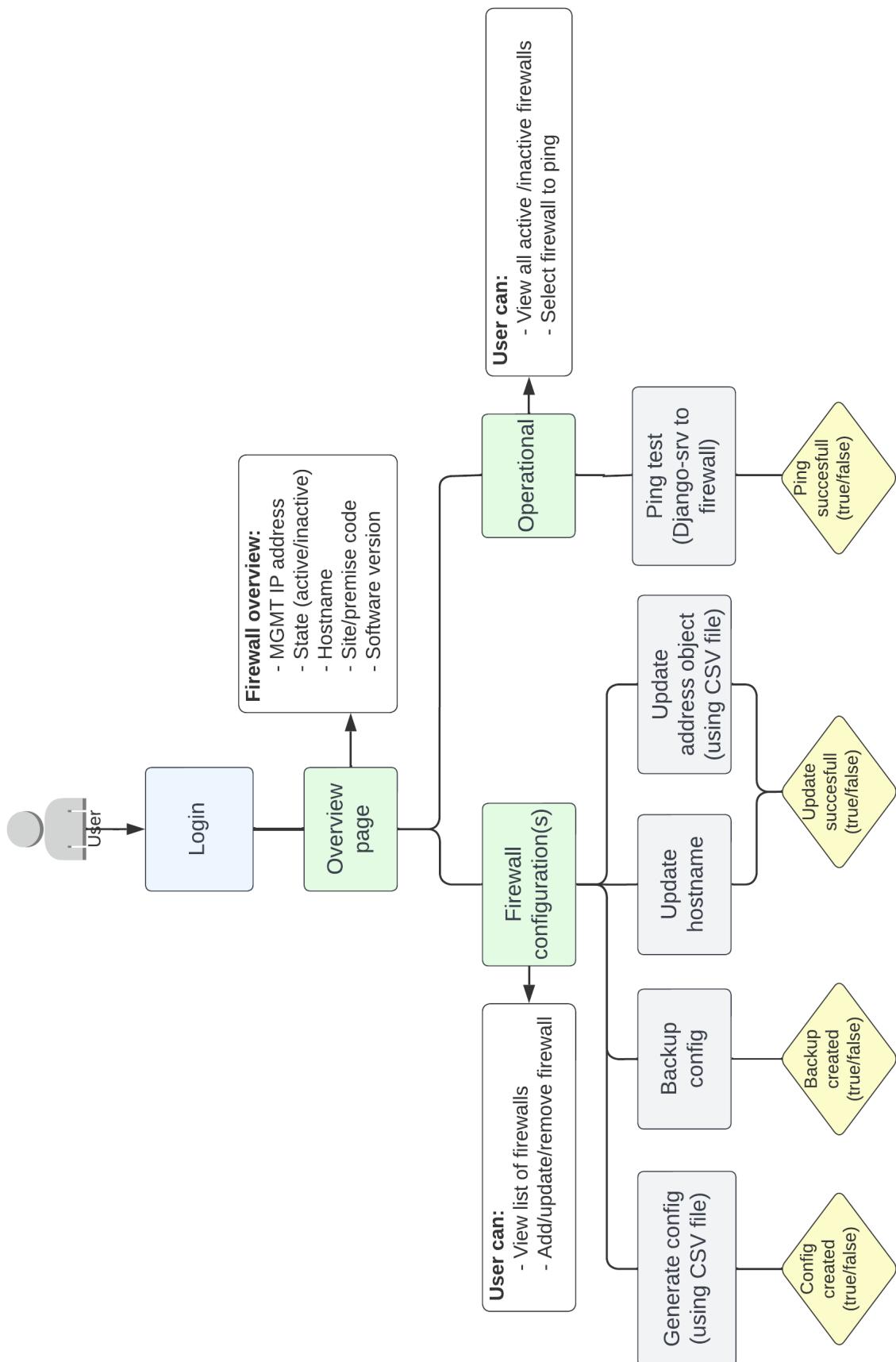
Bijlage 4: Basis labo opstelling

Bijlage 5: Netwerk/hosting diagram

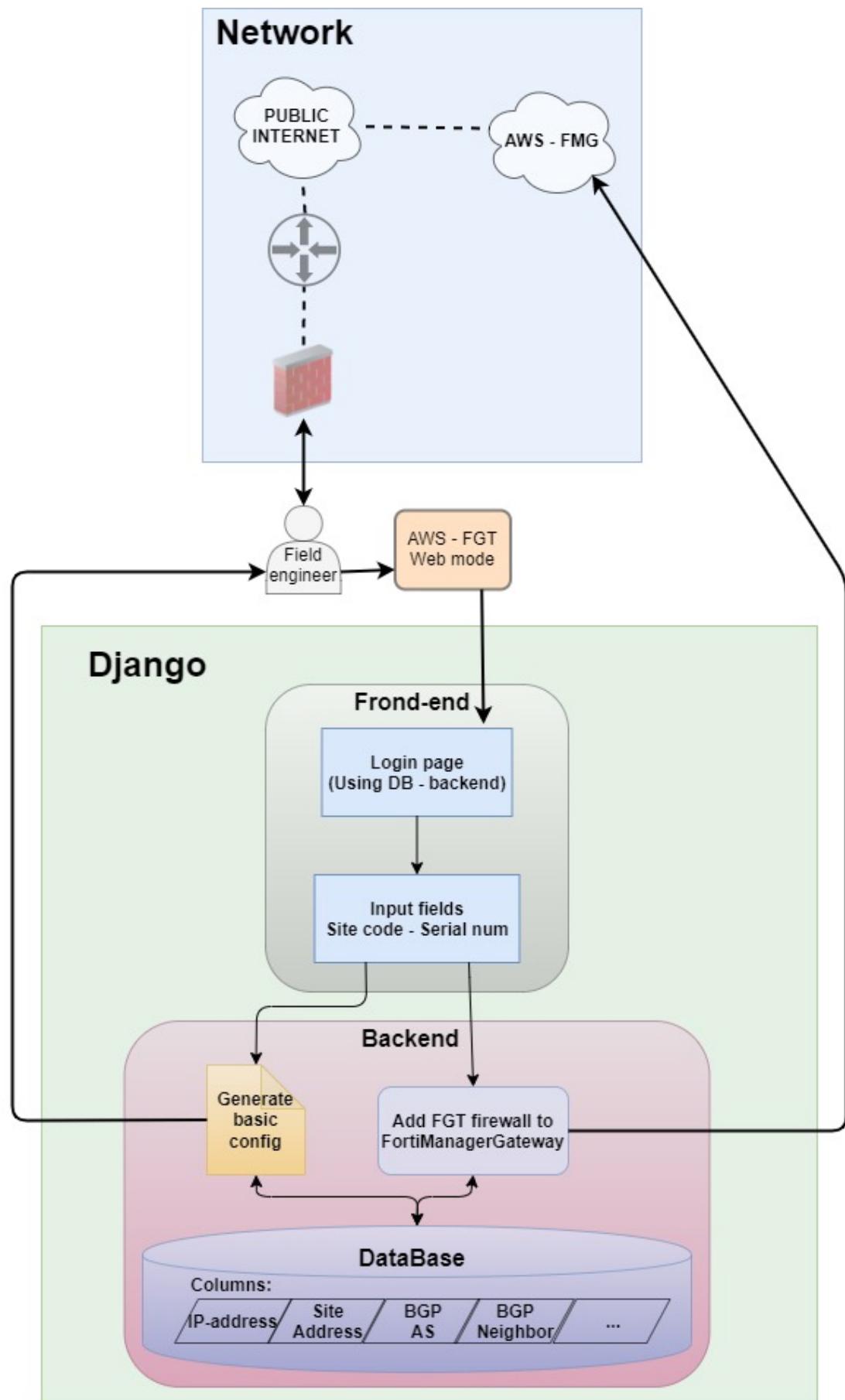
Bijlage 6: Logboek

Bijlage 7: Real-world hosting scenario

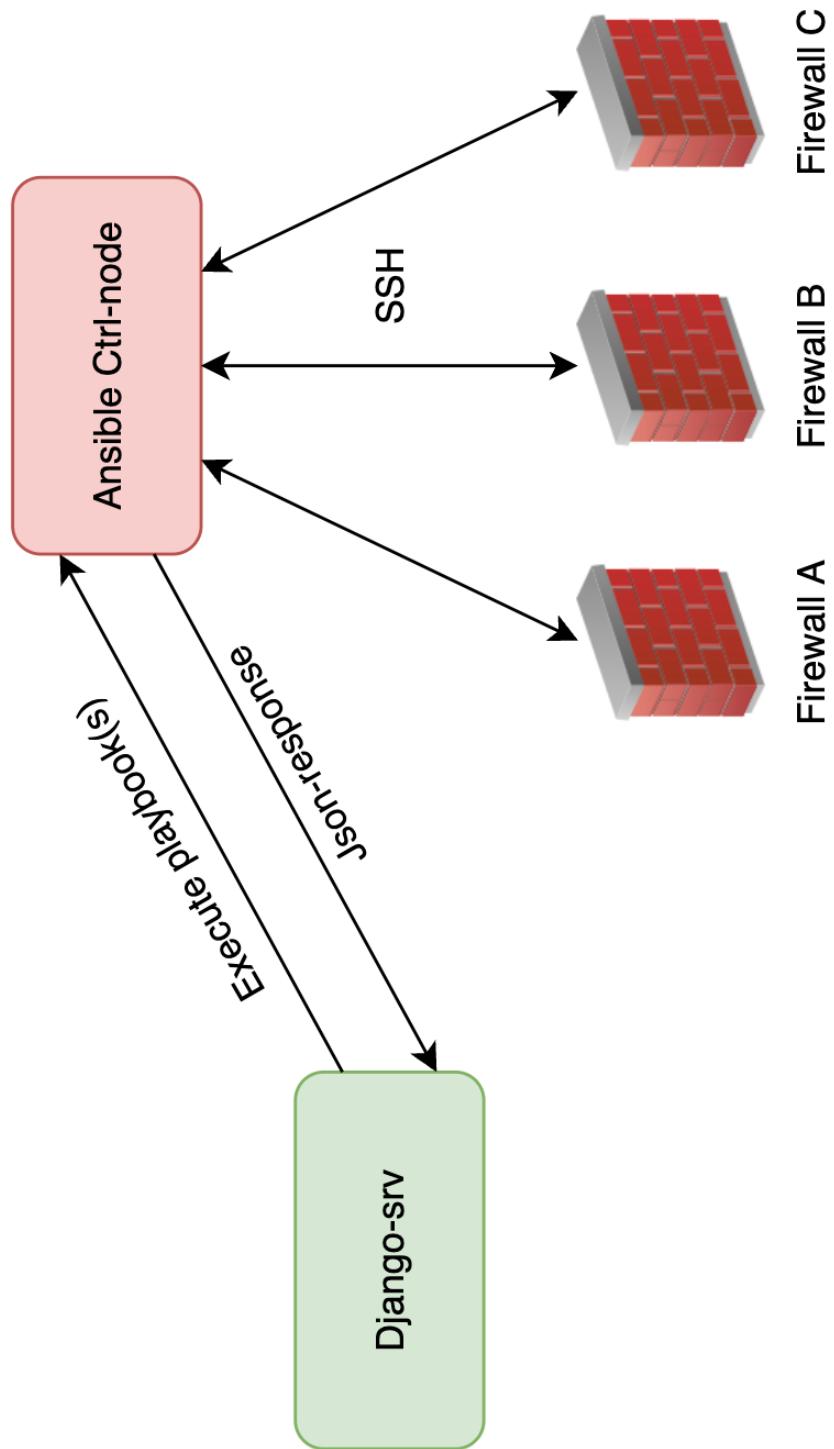
Bijlage 1: Platform flowchart



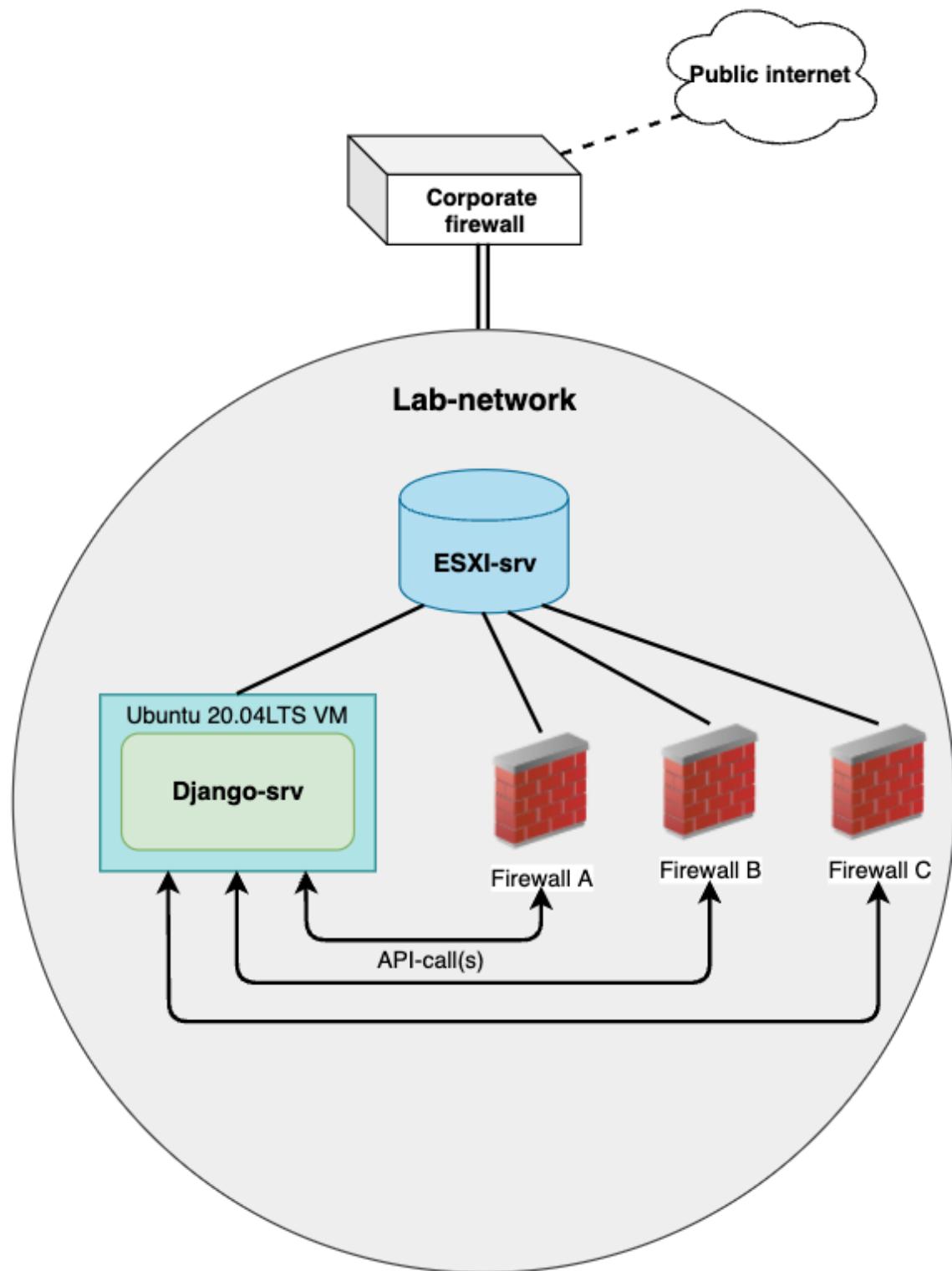
Bijlage 2: Fortimanager alternatief scenario



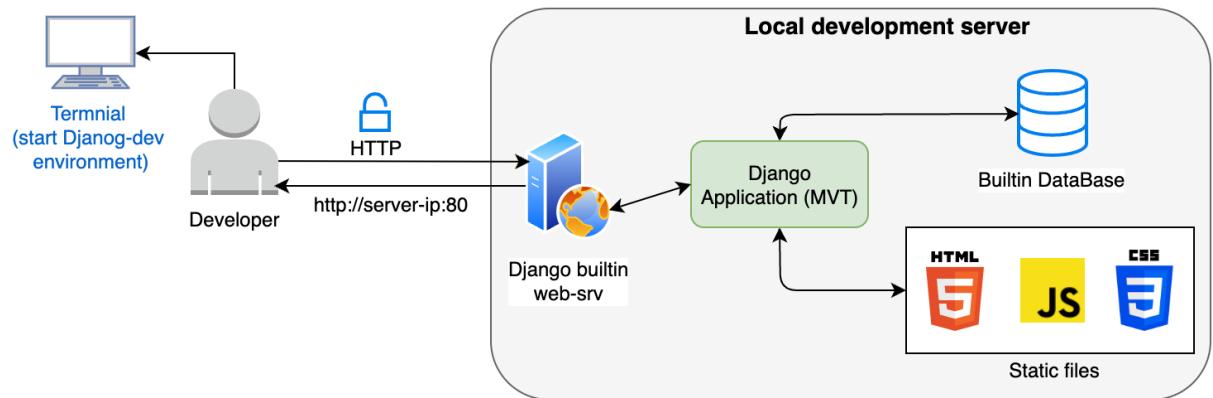
Bijlage 3: Ansible alternatief scenario



Bijlage 4: Basisopstelling



Bijlage 5: Development hosting diagram

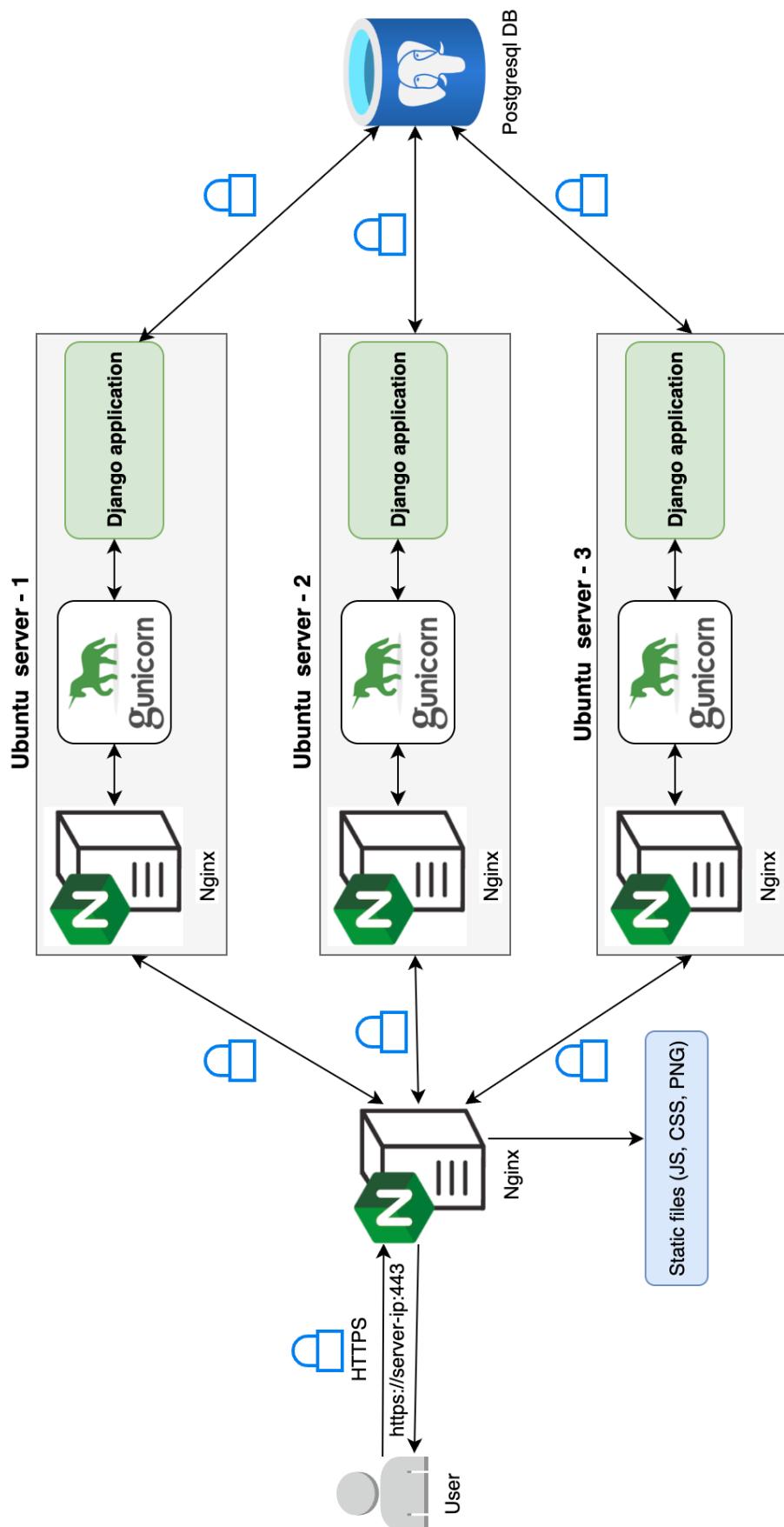


Bijlage 6: Logboek

Volledige link naar logboek van project:

https://hackmd.io/@DSL4hE_5RQy8PVEMU7XOp/BJqQFr-Ic

Bijlage 7: Real-world hosting scenario



Bijlage 8: Volledige opstelling

