Institut Universitaire de Technologie,
Aix-Marseille Université

ANNEXES
RAPPORT DE STAGE de fin de deuxième année
Bachelor Universitaire de Technologie
Spécialité Réseaux et Télécommunications
parcours cybersécurité

Modélisation et Gestion de l'infrastructure physique d'un Campus

Rayan SABQI

DIRNUM Luminy

Responsable entreprise : Pascal MOURET

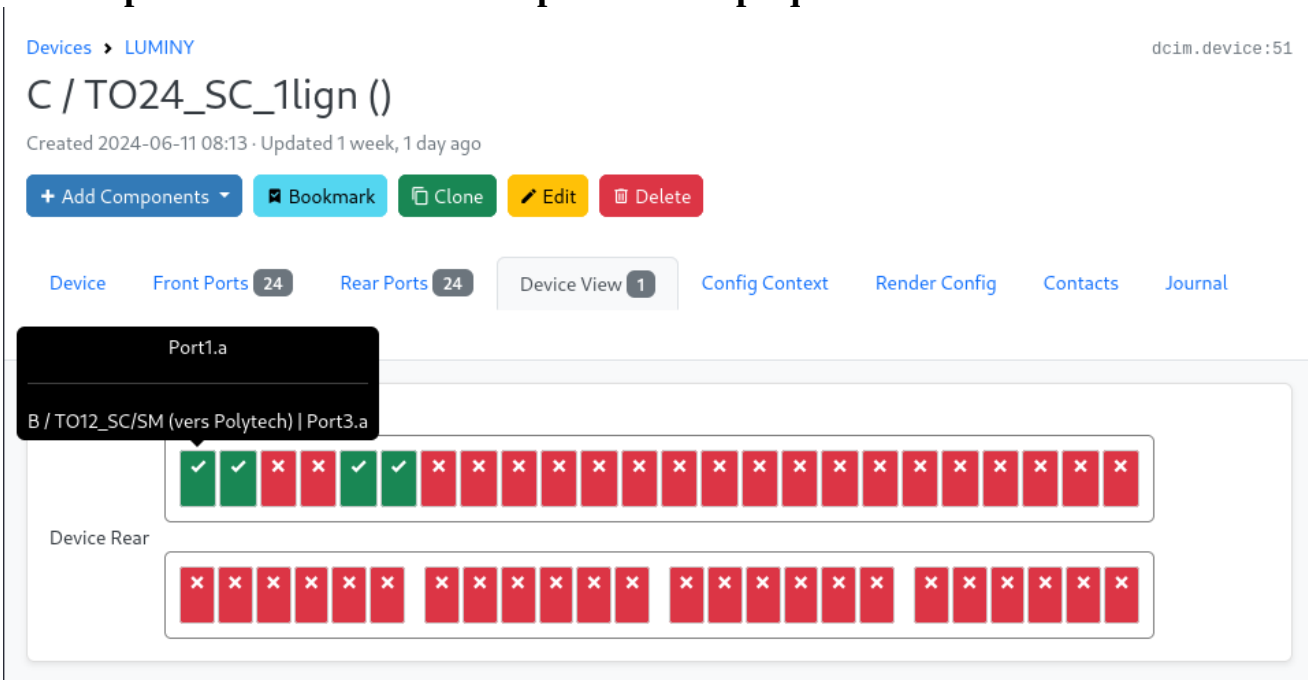
Responsable académique : Tin NGUYEN

2024

Table des matières

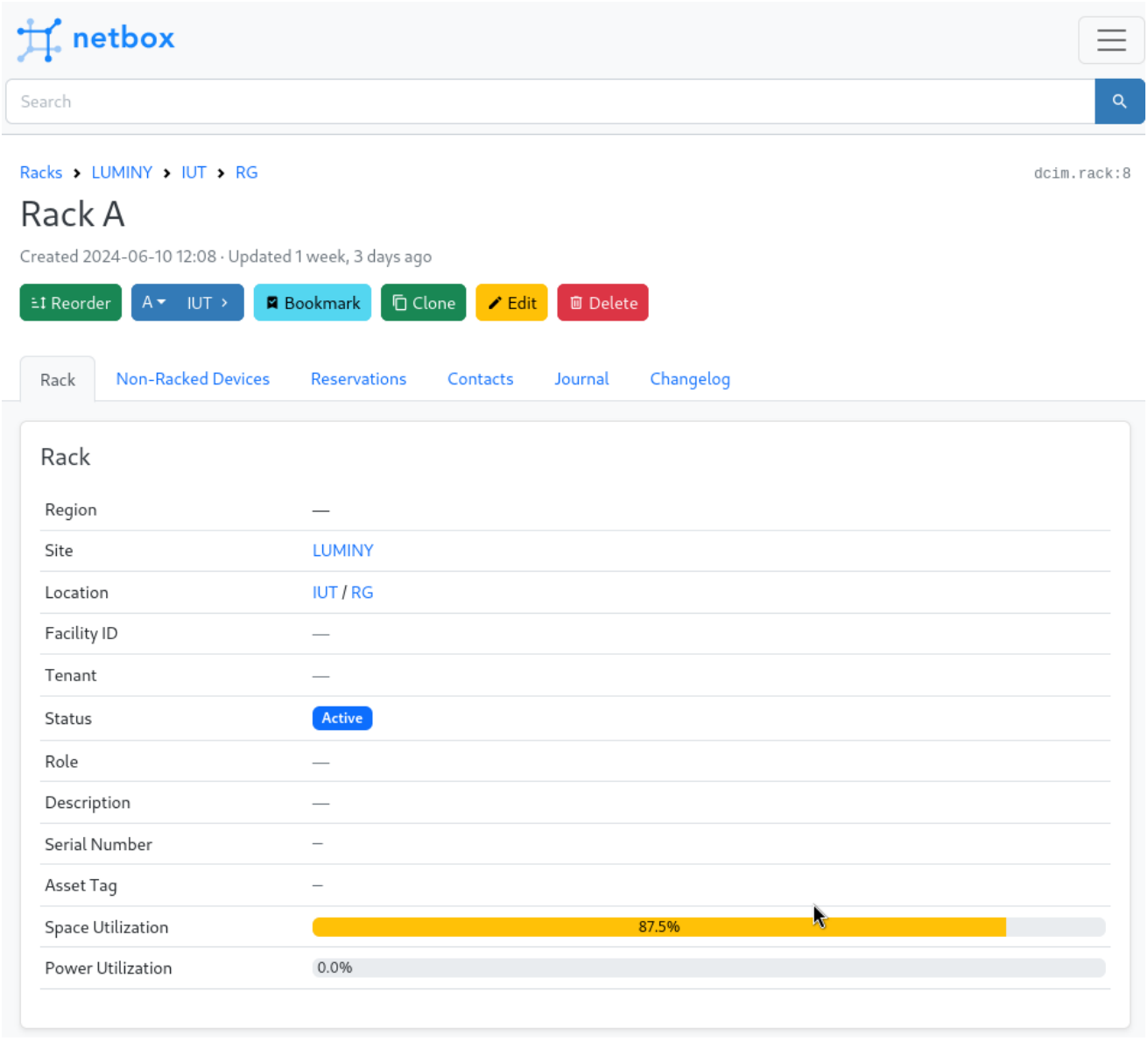
1 Exemple de template Device View pour une Prise:

2 Exemple de rendu Device View pour tiroir optique :



Celle-ci concerne le tiroir optique 24 ports provenant de l'IUT (local RG) et sa baie A.

3 Baie de l'IUT (local RG)



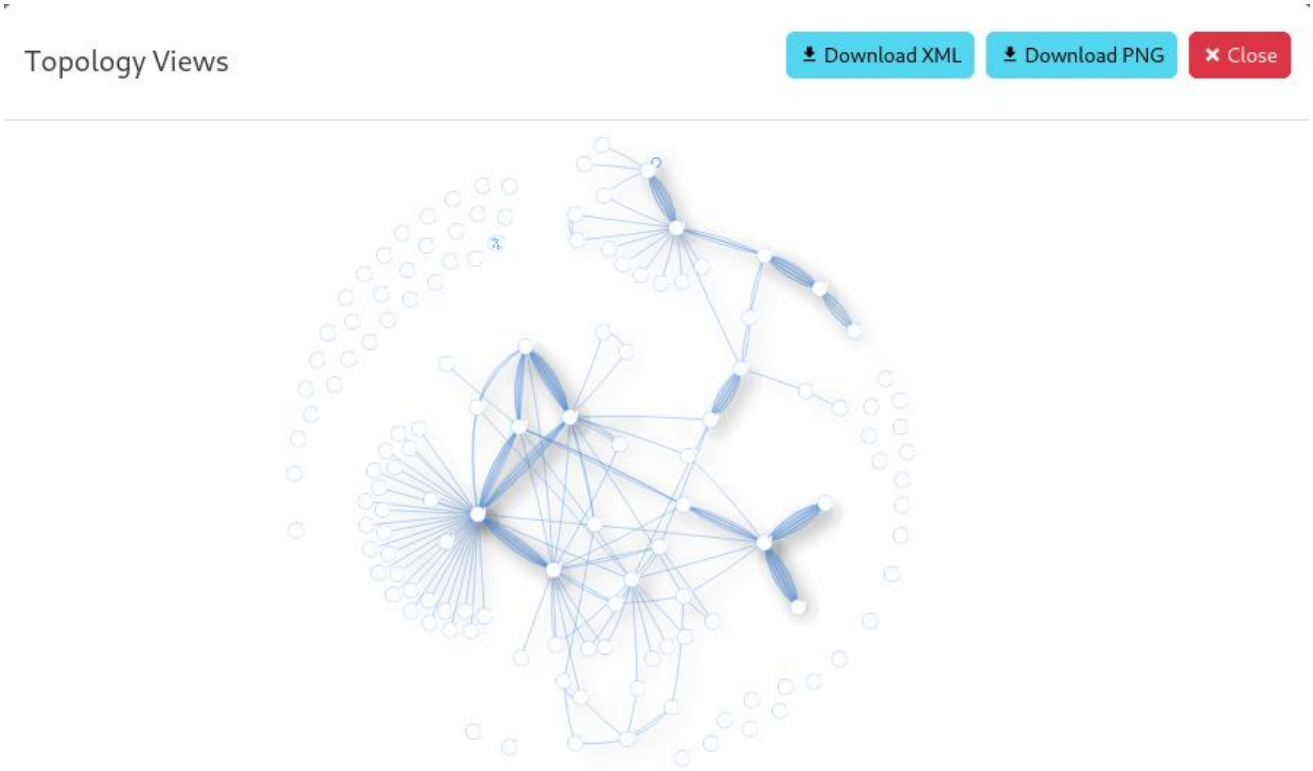
4 Baie de l'IUT (local RG) (vue graphique)

Note : Image modifiée pour protéger l'entreprise et ses locaux

5 Exemple template pour création d'un matériel

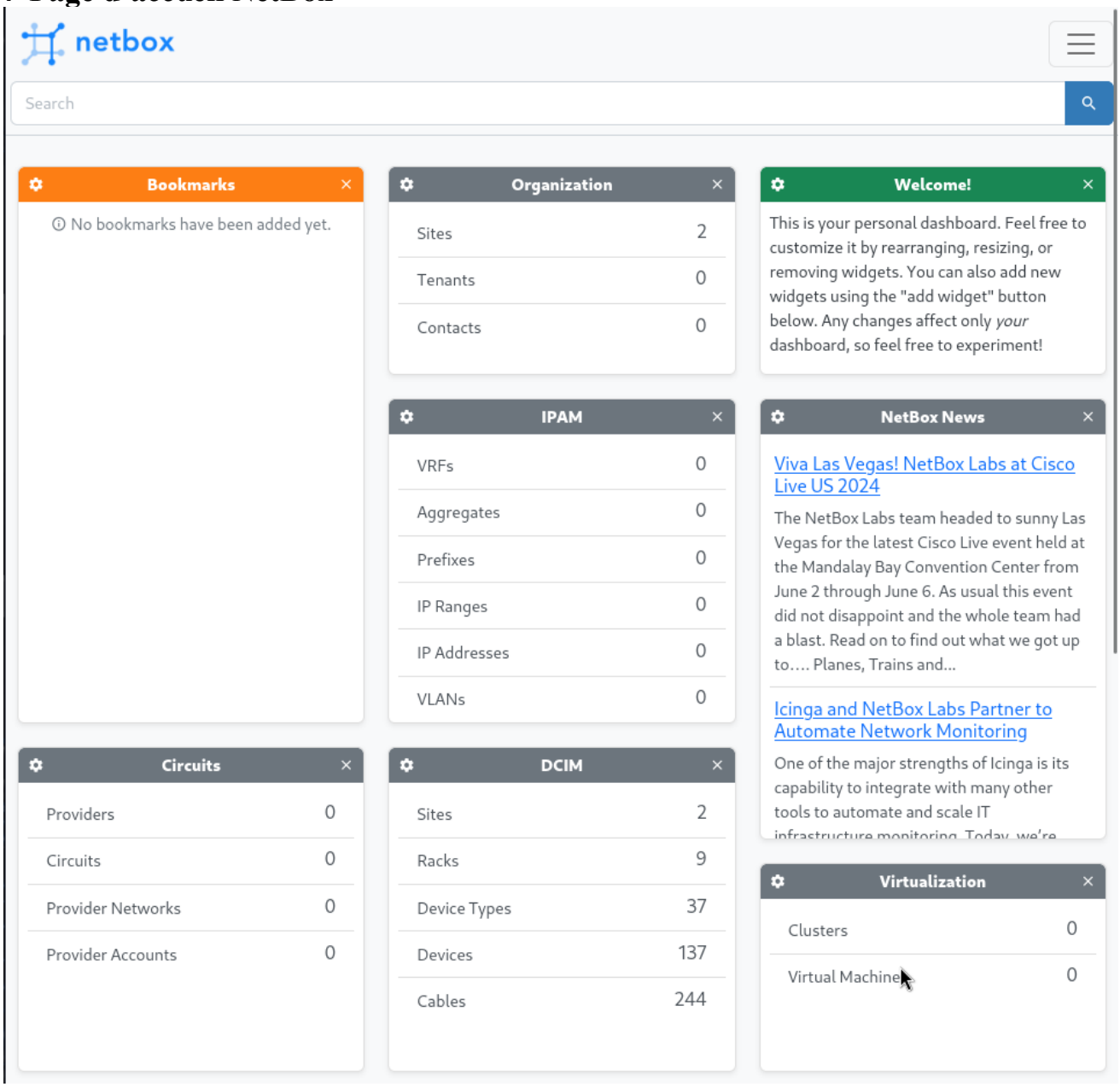
```yaml
---
manufacturer: CONVERTISSEURS
model: ConvFO-RS232
part_number: Unknown
slug: convfo-rs232
u_height: 1
comments: convertisseur Fibre optique vers RS232
interfaces:
  - name: FibreOptique
    type: 1000base-x-sfp
console-ports:
  - name: Serial
    type: de-9
```

6 Exemple Test Plugin Topology



Les connexions sont représentées par des liens, les périphériques par des cercles (modifiable)

7 Page d'accueil NetBox



8 Scripts pour automatiser la création de template Device View

8.1 Equipement Actif

```
#Pour regler le probleme du numero a remplacer il faut
#simplemet proceder comme suit:
#ex: port1_a on va demander a l'user de mettre le numero voulu qui va augmenter entre deux $
#ensuite on utilise une fonction replace qui va chercher le chiffre entre les deux $
#et le remplacer par le numero d'index i

# Vérification du format des noms de port
def format_port_name(port_name):
  if "." in port_name:
     return port_name.replace('.', '_')
  if "/" in port_name:
     return port_name.replace('/', '-')
  return port_name


def generate_css(num_ports, num_lines, first_front_port, module="non", nbr_module=None,
nom_port_module=None, inverser="non"):
  front_css = ""
  front_css_mid = ""
  y = 1
  z = 0
  module_css = ""


  if num_lines == 1:
     # Génerer les front et rear port names
     for i in range(1, num_ports + 1):
        debut_front_port = format_port_name(first_front_port).split('$')[0]
        fin_front_port = format_port_name(first_front_port).split('$')[2]

        front_port = f"{debut_front_port}{i}{fin_front_port}"

        # ajout des espaces entre ports
        if y % 6 == 0:
           front_port += f" s{z}"
           z+=1
        y += 1

        # La chaine concatenée finale
        front_css += front_port + " "

     if module.lower() == "oui":
        for j in range(1, nbr_module + 1):
           debut_module_port = format_port_name(nom_port_module).split('$')[0]
           fin_module_port = format_port_name(nom_port_module).split('$')[2]
           module_port = f"{debut_module_port}{j}{fin_module_port}"
           module_css += module_port + " "
```

```
            module_css += f"s{z+1}"
            front_css += module_css


        # generer le CSS
        css = """
            .deviceview.area.dFront {{
                /* both set to auto to overwrite the 32 columns 2 rows design */
                background: #000;
                grid-auto-rows: auto;
                grid-auto-columns: auto;
                grid-template-areas:
                    "{0}";
            }}

            """.format(front_css.strip())



    if num_lines == 2:
        front_css_p1 = ""
        front_css_p2 = ""
        for k in range(1, num_ports, 2):
            debut_front_port = format_port_name(first_front_port).split('$')[0]
            fin_front_port = format_port_name(first_front_port).split('$')[2]

            front_port = f"{debut_front_port}{k}{fin_front_port}"
            # ajout des espaces entre ports
            if y % 6 == 0:
                front_port += f" s{z}"
                z+=1
            y += 1

            front_css_p1 += front_port + " "

        for l in range(2, num_ports+1, 2):
            debut_front_port = format_port_name(first_front_port).split('$')[0]
            fin_front_port = format_port_name(first_front_port).split('$')[2]

            front_port = f"{debut_front_port}{l}{fin_front_port}"
            # ajout des espaces entre ports
            if y % 6 == 0:
                front_port += f" s{z}"
                z+=1
            y += 1

            front_css_p2 += front_port + " "


        if module == "oui":
            #On s'occupe du cas ou il y a des modules
            for m in range(1,nbr_module,2):
```

```
            debut_module_port = format_port_name(nom_port_module).split('$')[0]
            fin_module_port = format_port_name(nom_port_module).split('$')[2]

            module_port = f"{debut_module_port}{m}{fin_module_port}"
            front_css_p1 += module_port + " "
        for n in range(2, nbr_module,2):
            debut_module_port = format_port_name(nom_port_module).split('$')[0]
            fin_module_port = format_port_name(nom_port_module).split('$')[2]

            module_port = f"{debut_module_port}{n}{fin_module_port}"
            front_css_p2 += module_port + " "

    # generer le CSS
    if inverser.lower() == "oui":
        css = """
            .deviceview.area.dFront {{
                /* both set to auto to overwrite the 32 columns 2 rows design */
                background: #000;
                grid-auto-rows: auto;
                grid-auto-columns: auto;
                grid-template-areas:
                    "{0}"
                    "{1}";
            }}

            """.format(front_css_p2.strip(),front_css_p1.strip())

    else:
        css = """
            .deviceview.area.dFront {{
                /* both set to auto to overwrite the 32 columns 2 rows design */
                background: #000;
                grid-auto-rows: auto;
                grid-auto-columns: auto;
                grid-template-areas:
                    "{0}"
                    "{1}";
            }}

            """.format(front_css_p1.strip(),front_css_p2.strip())

    return css

# Demander les entrées de l'utilisateur
num_ports = int(input("Nombre de ports: "))
num_lines = int(input("Nombre de lignes pour le grid area (1 ou 2): "))
first_front_port = input("Nom du premier port avant (mettre le numero a changer entre deux $): ")
inverser = input("Disposition des ports inversés Oui/Non ? (ex: port1 sur la ligne basse, port2 sur la ligne haute: )")
module = input("Possède un module: Oui/Non? ")
if module.lower() == "oui":
```

```
  nbr_module = int(input("Nombre de ports de ce module: "))
  nom_port_module = input("Nom du premier port du module (mettre le chiffe a incrementer entre
deux $): ")
  css = generate_css(num_ports, num_lines, first_front_port,
module,nbr_module,nom_port_module,inverser)
else:
  css = generate_css(num_ports, num_lines, first_front_port, module, inverser)


# Générer et afficher le CSS
print(css)
```

8.2 Equipements Passifs

```
# Vérification du format des noms de port
def format_port_name(port_name):
  if "." in port_name:
     return port_name.replace('.', '_')
  if "/" in port_name:
     return port_name.replace('/', '-')
  return port_name


def generate_css(num_ports, num_lines, first_front_port, first_rear_port):
  front_css = ""
  rear_css = ""
  front_css_mid = ""
  rear_css_mid = ""
  y = 1
  z = 0

  if num_lines == 1:
     # Génerer les front et rear port names
     for i in range(1, num_ports + 1):
        debut_front_port = format_port_name(first_front_port).split('$')[0]
        debut_rear_port = format_port_name(first_rear_port).split('$')[0]
        fin_front_port = format_port_name(first_front_port).split('$')[2]
        fin_rear_port = format_port_name(first_rear_port).split('$')[2]

        front_port = f"{debut_front_port}{i}{fin_front_port}"
        rear_port = f"{debut_rear_port}{i}{fin_rear_port}"

        # ajout des espaces entre ports
        if y % 6 == 0:
           front_port += f" s{z}"
           rear_port += f" s{z}"
```

```
            z+=1
        y += 1

        # La chaine concatenée finale
        front_css += front_port + " "
        rear_css += rear_port + " "

    # generer le CSS
    css = """
        .deviceview.area.dFront {{
            /* both set to auto to overwrite the 32 columns 2 rows design */
            background: #FFF;
            grid-auto-rows: auto;
            grid-auto-columns: auto;
            grid-template-areas:
                {0};
        }}

        .deviceview.area.dRear {{
            /* both set to auto to overwrite the 32 columns 2 rows design */
            background: #FFF;
            grid-auto-rows: auto;
            grid-auto-columns: auto;
            grid-template-areas:
                {1};
        }}
        """.format(front_css.strip(), rear_css.strip())

if num_lines == 2:
    nbr_final = 0
    num_divise = num_ports//2
    # Génerer les front et rear port names
    while nbr_final < num_ports:
        nbr_port = 0
        while nbr_port < num_divise:
            nbr_final+=1
            nbr_port+=1
            debut_front_port = format_port_name(first_front_port).split('$')[0]
            debut_rear_port = format_port_name(first_rear_port).split('$')[0]
            fin_front_port = format_port_name(first_front_port).split('$')[2]
            fin_rear_port = format_port_name(first_rear_port).split('$')[2]

            front_port = f"{debut_front_port}{nbr_final}{fin_front_port}"
            rear_port = f"{debut_rear_port}{nbr_final}{fin_rear_port}"

            # ajout des espaces entre ports
            if y % 6 == 0:
                front_port += f" s{z}"
                rear_port += f" s{z}"
                z+=1
            y += 1
```

```
            front_css_mid += front_port + " "
            rear_css_mid += rear_port + " "

        # La chaine concatenée finale
        front_css += '"' + front_css_mid + '"' + '\n'
        rear_css += '"' + rear_css_mid + '"' + '\n'
        front_css_mid = ""
        rear_css_mid = ""


    # generer le CSS
    css = """
        .deviceview.area.dFront {{
            /* both set to auto to overwrite the 32 columns 2 rows design */
            background: #FFF;
            grid-auto-rows: auto;
            grid-auto-columns: auto;
            grid-template-areas:
                {0};
        }}

        .deviceview.area.dRear {{
            /* both set to auto to overwrite the 32 columns 2 rows design */
            background: #FFF;
            grid-auto-rows: auto;
            grid-auto-columns: auto;
            grid-template-areas:
                {1};
        }}
        """.format(front_css.strip(), rear_css.strip())

  return css

# Demander les entrées de l'utilisateur
num_ports = int(input("Nombre de ports: "))
num_lines = int(input("Nombre de lignes pour le grid area (1 ou 2): "))
first_front_port = input("Nom du premier port avant (mettre le numero a changer entre deux $): ")
first_rear_port = input("Nom du premier port arrière(mettre le numero a changer entre deux $): ")

# Générer et afficher le CSS
css = generate_css(num_ports, num_lines, first_front_port, first_rear_port)
print(css)
```

9 Script d'importation des locaux

Note : Il ressemble à 90% au script pour les bâtiments voilà pourquoi le second ne sera pas mis ici

```
import csv
import requests
import urllib3

from unidecode import unidecode
import re


# Configuration de base
#Mon serveur perso:
#NETBOX_URL = 'https://@ip/api/'  # Remplacer par l'URL du NetBox voulu
#API_TOKEN = ''  # Remplacer par le token API

#Le serveur de production
NETBOX_URL = 'https://@ip/api/'
API_TOKEN = ''


# En-têtes pour l'authentification
headers = {
  'Authorization': f'Token {API_TOKEN}',
  'Content-Type': 'application/json',
  'Accept': 'application/json',
}

# Fonction pour créer une location
def create_location(name, slug, site_id, type, parent_slug=None, description=None):
  url = f'{NETBOX_URL}dcim/locations/'
  payload = {
     'name': name,
     'slug': slug,
     'site': site_id,
     'tags': type,
  }
  if parent_slug:
     parent_id = get_parent_id(parent_slug)
     if parent_id:
        payload['parent'] = parent_id
  if description:
     payload['description'] = description
  print(payload)
  response = requests.post(url, headers=headers, json=payload, verify=False)
  if response.status_code == 201:
     print(f"Location créée avec succès: {name}")
```

```
  else:
    print(f"Erreur lors de la création de la location {name}: {response.status_code}")
    print(response.text)

# Fonction pour obtenir l'ID du parent à partir du slug
def get_parent_id(slug):
  url = f'{NETBOX_URL}dcim/locations/?slug={slug}'
  response = requests.get(url, headers=headers, verify=False)
  if response.status_code == 200:
    results = response.json().get('results', [])
    if results:
      return results[0]['id']
  else:
    print(f"Erreur lors de la récupération du parent {slug}: {response.status_code}")
    print(response.text)
  return None

# Fonction pour migrer les locations depuis un fichier CSV
def migrate_locations(csv_file_path, site_id):
  with open(csv_file_path, newline='') as csvfile:
    reader = csv.DictReader(csvfile)
    for row in reader:
      name = row['name']
      slug_mid = unidecode(name.strip())
      clean_slug = re.sub(r'[^a-z0-9-_]', '-', slug_mid.lower())
      slug_parent_mid = unidecode(row['parent'].strip())
      clean_parent_slug = re.sub(r'[^a-z0-9-_]', '-', slug_parent_mid.lower())
      parent_slug = "luminy-" + clean_parent_slug if 'parent' in row else None
      slug =  parent_slug + "-" + clean_slug
      print(slug)
      description = row['description'] if 'description' in row else None
      salle_ou_local = row['type']
      #Pour un local
      if salle_ou_local == "B":
        type = [{"id":2}]
      #Pour une salle
      if salle_ou_local == "S":
        type = [{"id":1}]
      create_location(name, slug, site_id, type, parent_slug, description)

if __name__ == '__main__':
  # Chemin vers le fichier CSV contenant les données des locations
  CSV_FILE_PATH = "/home/kali/Desktop/devicetype-library-master/templates devices view et
scripts/locauxFinal.csv"

  # ID du site dans lequel les locations doivent être ajoutées
  SITE_ID = 1  # Remplacer par l'ID du site voulu

  # Migrer les locations
  migrate_locations(CSV_FILE_PATH, SITE_ID)
```