|  |  |
| --- | --- |
| **ĐẠI HỌC QUỐC GIA HÀ NỘI** | |
| **TRƯỜNG ĐẠI HỌC CÔNG NGHỆ** | |
| **----------------------------------------** | |
| **https://lh5.googleusercontent.com/jnQBZlXHwHYTghGtnCOg_bRjT0V1H9A8LoWZNHFWQCNM0PyJzdO4Dpf_5U5Rb-NS0KEARdcRP2mm4Na-xpsmR_MsAiAVGDMfngh8037PAY3yuqLfY2se36gFzPdn1Y2W5uh7fvIRQW7QzjGeTY8mAeZgt2n38rHXxgy3XdEsvckikR1d0I6WSQ** | |
| **BÁO CÁO THỰC TẬP** | |
| NGÀNH: ĐIỆN TỬ VIỄN THÔNG | |
|  | |
| ĐỀ TÀI: TÌM HIỂU MÃ NGUỒN MỞ NETBOX VÀ XÂY DỰNG THÊM TÍNH NĂNG QUẢN LÝ STORAGE CHO NETBOX | |
|  | |
|  | |
|  | |
|  | **Cán bộ hướng dẫn**: Nguyễn Văn Hùng  **Giảng viên đánh giá:** PGS.TS.Nguyễn Nam Hoàng |
|  | **Sinh viên:** Trần Bảo Thịnh |
|  | **Mã sinh viên:** 19021516 |
|  | **Lớp:** DACLC2 |

**Hà Nội - tháng 9 năm 2022**

# **LỜI MỞ ĐẦU**

Thực hiện nhiệm vụ học tập của nhà trường trong đợt thực tập của học phần Thực tập thiết kế hệ thống, được sự đồng ý của ban lãnh đạo Khoa Điện tử-Viễn thông và sự hỗ trợ của Tổng công ty mạng lưới Viettel Network. Bám sát đề cương của Khoa đưa ra, những lý luận và kiến thức đã học. Trong quá trình thực tập em đã được trau dồi thêm kiến thức về mạng lập trình mạng chung và kĩ năng phát triển code nói riêng; cũng như cơ cấu và tình hình hoạt động của công ty.

Được sự tận tình giúp đỡ hướng dẫn của anh Nguyễn Văn Hùng (Cán bộ hướng dẫn thực tập) cùng với sự hỗ trợ của giáo viên hướng dẫn Nguyễn Nam Hoàng; và nỗ lực trong học tập, tìm tòi học hỏi, em đã hoàn thành báo cáo thực tập của mình.

# **LỜI CẢM ƠN**

Em xin chân thành cảm ơn tổng công ty mạng lưới Viettel đã tạo điều kiện cho em được thực tập tại đây. Ngoài ra, anh Nguyễn Văn Hùng - cán bộ hướng dẫn đã luôn giúp đỡ chỉ dẫn và giảng dạy cho em những kiến thức chuyên môn để em có thể hoàn thiện hơn.

Em xin gửi lời cảm ơn tới ban lãnh đạo Khoa đào tạo đã thiết lập khung chương trình có môn học “Thực tập” để em có thể vận dụng những kiến thức học được trên giảng đường áp dụng vào công việc và được tham quan, tìm hiểu về môi trường doanh nghiệp.

Em cũng xin gửi lời cảm ơn tới giảng viên hướng dẫn thầy Nguyễn Nam Hoàng, thầy đã luôn giúp đỡ và chỉ bảo cho bọn em để hoàn thành tốt khóa thực tập

Em xin chân thành cảm ơn cảm ơn!

Hà Nội - tháng 9 năm 2022

**Mục lục**

[**LỜI MỞ ĐẦU** 2](#_Toc114433942)

[**LỜI CẢM ƠN** 3](#_Toc114433943)

[**Danh mục hình ảnh** 5](#_Toc114433944)

[**Danh mục từ viết tắt** 6](#_Toc114433945)

[**I.** **Tổng quan về đề tài** 7](#_Toc114433946)

[**1.** **Lý do chọn đề tài** 7](#_Toc114433947)

[**2.** **Giới thiệu phần mềm nghiên cứu** 7](#_Toc114433948)

[**3.** **Mô tả đề tài** 7](#_Toc114433949)

[**4.** **Yêu cầu** 7](#_Toc114433950)

[**5.** **Kết quả cần đạt được** 7](#_Toc114433951)

[**II.** **Các khái niệm và phần mềm** 7](#_Toc114433952)

[**1.** **Python Framework** 7](#_Toc114433953)

[**2.** **Django** 8](#_Toc114433954)

[**3.** **Pycharm** 8](#_Toc114433955)

[**4.** **GitHub** 8](#_Toc114433956)

[**5.** **Virtual Box** 8](#_Toc114433957)

[**6.** **Ubuntu** 8](#_Toc114433958)

[**7.** **Netbox** 9](#_Toc114433959)

[**8.** **API** 9](#_Toc114433960)

[**9.** **Docker** 10](#_Toc114433961)

[**III.** **Nội dung thực tập** 10](#_Toc114433962)

[**1.** **Tìm hiểu về các khái niệm** 10](#_Toc114433963)

[**2.** **Cài đặt VirtualBox, Ubuntu và PyCharm.** 10](#_Toc114433964)

[**3.** **Cài đặt và thiết lập khởi tạo một app Django đơn giản** 12](#_Toc114433965)

[**A.** **Cài đặt Django và Python** 12](#_Toc114433966)

[**B.** **Tạo app Django cơ bản** 13](#_Toc114433967)

[**4.** **Cài đặt và thiết lập NetBox** 24](#_Toc114433968)

[**A.** **PostgreSQL Database** 24](#_Toc114433969)

[**B.** **Redis** 25](#_Toc114433970)

[**C.** **Netbox components** 25](#_Toc114433971)

[**D.** **Gunicorn** 27](#_Toc114433972)

[**E.** **HTTP server** 28](#_Toc114433973)

[**F.** **LDAP authentication (optional)** 29](#_Toc114433974)

[**5.** **Phát triển tính năng quản lý Storage trên NetBox** 29](#_Toc114433975)

[**A.** **Tạo model** 30](#_Toc114433976)

[**B.** **Tạo ChoiceSet** 33](#_Toc114433977)

[**C.** **Tạo View** 35](#_Toc114433978)

[**D.** **Tạo đường dẫn url** 39](#_Toc114433979)

[**E.** **Tạo API** 39](#_Toc114433980)

[**F.** **Tạo File HTML để hiển thị trên web** 40](#_Toc114433981)

[**G.** **Kết quả** 46](#_Toc114433982)

[**6.** **Debug** 61](#_Toc114433983)

[**IV.** **Kết quả đạt được** 61](#_Toc114433984)

[**Tài liệu tham khảo** 62](#_Toc114433985)

[**NHẬN XÉT KẾT QUẢ THỰC TẬP** 64](#_Toc114433986)

# **Danh mục hình ảnh**

[*Hình ảnh 1: Mô tả hệ thống Netbox* 9](#_Toc114433987)

[*Hình ảnh 2: Cấu hình máy ảo virtualbox/Ubuntu đã tạo* 11](#_Toc114433988)

[*Hình ảnh 3: Sau khi cài đặt Pycharm trên máy ảo* 12](#_Toc114433989)

[*Hình ảnh 4: Giao diện sau khi tạo xong một app vote web* 23](#_Toc114433990)

[*Hình ảnh 5: Giao diện khi sửa các choices* 24](#_Toc114433991)

[*Hình ảnh 6: Demo của NetBox có sẵn* 29](#_Toc114433992)

[*Hình ảnh 7:Giao diện của Device trong NetBox* 30](#_Toc114433993)

[*Hình ảnh 8: File dcim/model được mở trên pycharm* 31](#_Toc114433994)

[*Hình ảnh 9: Giao diện của model 1 Storage bất kì* 33](#_Toc114433995)

[*Hình ảnh 10: Các Choices được tạo khi xuất hiện trên giao diện* 34](#_Toc114433996)

[*Hình ảnh 11: Các tính năng được tạo* 36](#_Toc114433997)

[*Hình ảnh 12: Thay đổi port bằng VirtualBox* 48](#_Toc114433998)

[*Hình ảnh 13: Giao diện NetBox sau khi có sửa đổi* 49](#_Toc114433999)

[*Hình ảnh 14: Mục Storage riêng đã được tạo* 49](#_Toc114434000)

[*Hình ảnh 15: Giao diện danh sách các Storage* 50](#_Toc114434001)

[*Hình ảnh 16: Thông tin của Storage 3* 50](#_Toc114434002)

[*Hình ảnh 17: Các thay đổi của Storage 3* 51](#_Toc114434003)

[*Hình ảnh 18: Giao diện khi thay đổi nhiều Storage* 51](#_Toc114434004)

[*Hình ảnh 19: Giao diện khi thêm một Storage* 52](#_Toc114434005)

[*Hình ảnh 20: Giao diện Import* 53](#_Toc114434006)

[*Hình ảnh 21: Giao diện thay đổi một Storage* 53](#_Toc114434007)

[*Hình ảnh 22: Giao diện Filter trong Storage* 54](#_Toc114434008)

[*Hình ảnh 23: Đường dẫn API của Devices chứa Storages* 54](#_Toc114434009)

[*Hình ảnh 24: Tạo một storage 20* 57](#_Toc114434010)

[*Hình ảnh 25: Thông tin Storage sau khi tạo* 57](#_Toc114434011)

[*Hình ảnh 26: Storage 20 được hiển thị trên danh sách* 58](#_Toc114434012)

[*Hình ảnh 27: Thử chỉnh sửa Storage 20* 58](#_Toc114434013)

[*Hình ảnh 28: Thông tin sau chỉnh sửa* 59](#_Toc114434014)

[*Hình ảnh 29: Thử xóa Storage 20* 59](#_Toc114434015)

[*Hình ảnh 30: Kết quả thành công sau khi xóa* 60](#_Toc114434016)

[*Hình ảnh 31: Thử tính năng lọc* 60](#_Toc114434017)

# **Danh mục từ viết tắt**

|  |  |
| --- | --- |
| IPAM | Internet protocol address management |
| DCIM | Data center infrastructure management |
| REST (API) | Representational state transfer |
| API | Application Programming Interface |
| SOAP (API) | Simple Object Access Protocol |
| RPC (API) | Remote Procedure Call |
| SQL | Structured Query Language |
| MAC OS | MAC Operating System |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |
| IAAS | Infrasructure as a service |
| PAAS | Platform as a service |
| SAAS | Software as a service |

# 

1. **Tổng quan về đề tài**
2. **Lý do chọn đề tài**

* Do nhu cầu quản lý thông tin của con người trên mạng lưới tăng lên và việc lập trình càng ngày càng phát triển và được biết đến rộng rãi vì vậy em muốn thử sức với một đề tài về mạng lưới cloud và thiên về lập trình để có thể phát triển bản thân từ đó định hướng và phát triển cho việc làm khóa luận tốt nghiệp không những thế em còn muốn tiếp cận với những công cụ, kỹ thuật mới nhất đồng thời có thể học hỏi thêm các kĩ năng lập trình, khái niệm để hỗ trợ cho sau này.

1. **Giới thiệu phần mềm nghiên cứu**

* Netbox là giải pháp hàng đầu để mô hình hóa và ghi lại các mạng hiện đại. Bằng cách kết hợp các quy tắc truyền thống của Quản lý địa chỉ IP (IPAM) và Quản lý cơ sở hạ tầng trung tâm dữ liệu (DCIM) với các API và tiện ích mở rộng mạnh mẽ, Netbox cung cấp ý tưởng về việc tự động hóa mạng ảo.
* Cloud Computing hay điện toán đám mây là một mô hình cung cấp tài nguyên máy tính cho người dùng thông qua Internet. Tài nguyên máy tính ở đây gồm rất nhiều thứ liên quan đến điện toán và máy tính. Chẳng hạn như phần mềm, phần cứng, dịch vụ … nằm tại các máy chủ ảo trên mạng.[1]
* Cloud Computing bao gồm 4 mô hình triển khai:
* Public Cloud
* Private Cloud
* Hybrid Cloud
* Community Cloud
* Các mô hình cung cấp Cloud Computing:
  + Iaas
  + Paas
  + Saas

1. **Mô tả đề tài**

* Từ resource của Netbox build một web app trên local và xây dựng thêm tính năng quản lý Storage như View/add/Edit/Import/Delete.

1. **Yêu cầu**

* Có kiến thức về lập trình Python , Django Framework, RESTful API, Docker,…

1. **Kết quả cần đạt được**

* Biết thiết kế Model, xây dựng được các tính năng.View/Add/Edit/Import/Delete thông tin Storage trên giao diện và viết API cho các tính năng trên.
* Việc thực hiện đề tài này sẽ hỗ trợ cho việc tiếp cận thông tin về những công nghệ và kỹ năng, phần mềm lập trình mới trong công nghệ.

1. **Các khái niệm và phần mềm**
2. **Python Framework [2]**

* Python Framework là các đoạn code đã được viết sẵn, một bộ khung và các thư viện lập trình được đóng gói. Việc này giúp cho các nhà phát triển tập trung nhiều vào logic ứng dụng hơn là các yếu tố thường.
* Tồn tạo rất nhiều loại python framework ở đây em tìm hiểu chính về các loại framework để phát triển web theo đề tài đang làm:
* Django Python framework
* Pyramid Python framework
* CherryPy Python framework
* Flask Python framework
* ….

1. **Django**

* Django là một trong những Python Back-end framework tốt nhất, là một khung Web Python cấp cao, khuyến khích phát triển nhanh và thiết kế dễ dàng và gọn gàng.  
  Django có thể xử lý nhiều rắc rối của việc phát triển Web nên ta có thể tập trung vào viết ứng dụng của mình mà không cần phải tạo lại khung.
* Các ưu điểm:
* Quá trình xử lý nhanh
* Rất nhiều thư viện có sẵn
* Khả năng mở rộng tốt
* Tính linh hoạt

1. **Pycharm**

* Pycharm là một nền tảng kết hợp được JetBrains phát triển như một IDE để phát triển các ứng dụng cho lập trình trong Python.
* Pycharm có thể chạy trên Window,Linux hoặc Mac OS
* Pycharm hỗ trợ lập trình viên những tính năng như:
* Trình sửa mã thông minh
* Điều hướng mã
* Tái cấu trúc

1. **GitHub**

* GitHub là một dịch vụ cung cấp kho lưu trữ mã nguồn Git dựa trên nền web cho các dự án phát triển phần mềm.
* GitHub có đầy đủ các tính năng của Git và còn bổ sung them những tính năng về social để các developer tương tác với nhau.

1. **Virtual Box**

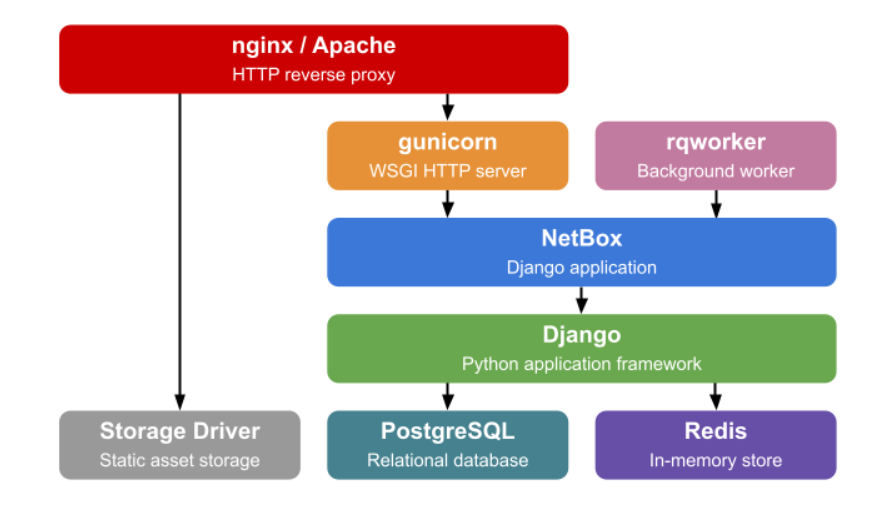
* Virtual Box là công cụ ảo hóa kiến trúc máy tính x89 và AMSD64/Intel64 cho phép triển khai máy chủ, máy tính để bàn và các hệ điều hành dưới dạng máy ảo

1. **Ubuntu**

* Ubuntu là hệ điều hành mở do người dùng phát triển dựa trên Debian GNU/Linux bao gồm nhiều bản phân

1. **Netbox**

* NetBox là một công cụ mô hình hóa tài nguyên cơ sở hạ tầng (IRM) được thiết kế để trao quyền tự động hóa mạng, được sử dụng bởi hàng nghìn tở chức trên thế giới.
* Vô số thành phần cơ sở hạ tầng có thể được mô hình hóa trong NetBox.
* Ngoài các mô hình và chức năng có sẵn, NetBox có thể được tùy chỉnh và mở rộng theo nhiều cách.
* NetBox cũng có API REST hoàn chỉnh cũng như API GraphQL.
* NetBox là một ứng dụng web dựa trên Django Python với cơ sở dữ liệu PostgreSQL
* NetBox app stack:



*Hình ảnh 1: Mô tả hệ thống Netbox*

1. **API**

* API hay còn gọi là giao diện chương trình ứng dụng hiểu đơn giản là cơ chế cho phép 2 thành phần mềm giao tiếp với nhau bằng một tập hợp các định nghĩa và giao thức.
* API hoạt động theo 4 cách khác nhau:
* API SOAP
* API RPC
* API Websocket
* API REST (Tìm hiểu chính)
* API REST là loại API phổ biến và linh hoạt nhất trên web hiện nay. REST là viết tắt của Representational state transfer. REST bao gồm các hàm như GET,PUT,DELETE,… mà máy khác có thể truy cập vào dữ liệu của máy chủ thông qua giao thức http.
* Tính năng của API REST là tính không trạng thái tức là máy chủ không lưu dữ liệu của máy khác giữa các yêu cầu.

1. **Docker**

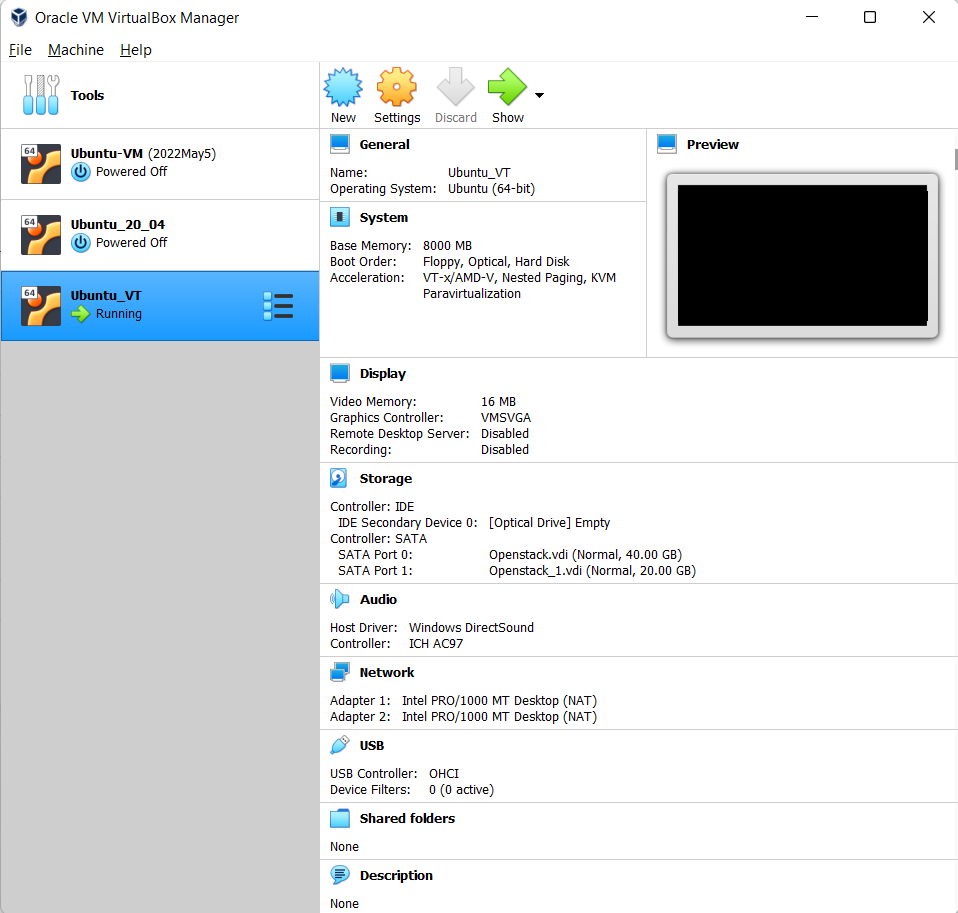
* Docker là một nền tảng để cung cấp cách để building, deloying và running ứng dụng dễ dàng hơn bằng cách sử dụng các containers. Ban đầu viết bằng Python, hiện giờ đã chuyển sang Golang.

1. **Nội dung thực tập**
2. **Tìm hiểu về các khái niệm**

Yêu cầu tìm hiểu trước về các khái niệm (đã nêu trên).

1. **Cài đặt VirtualBox, Ubuntu và PyCharm.**

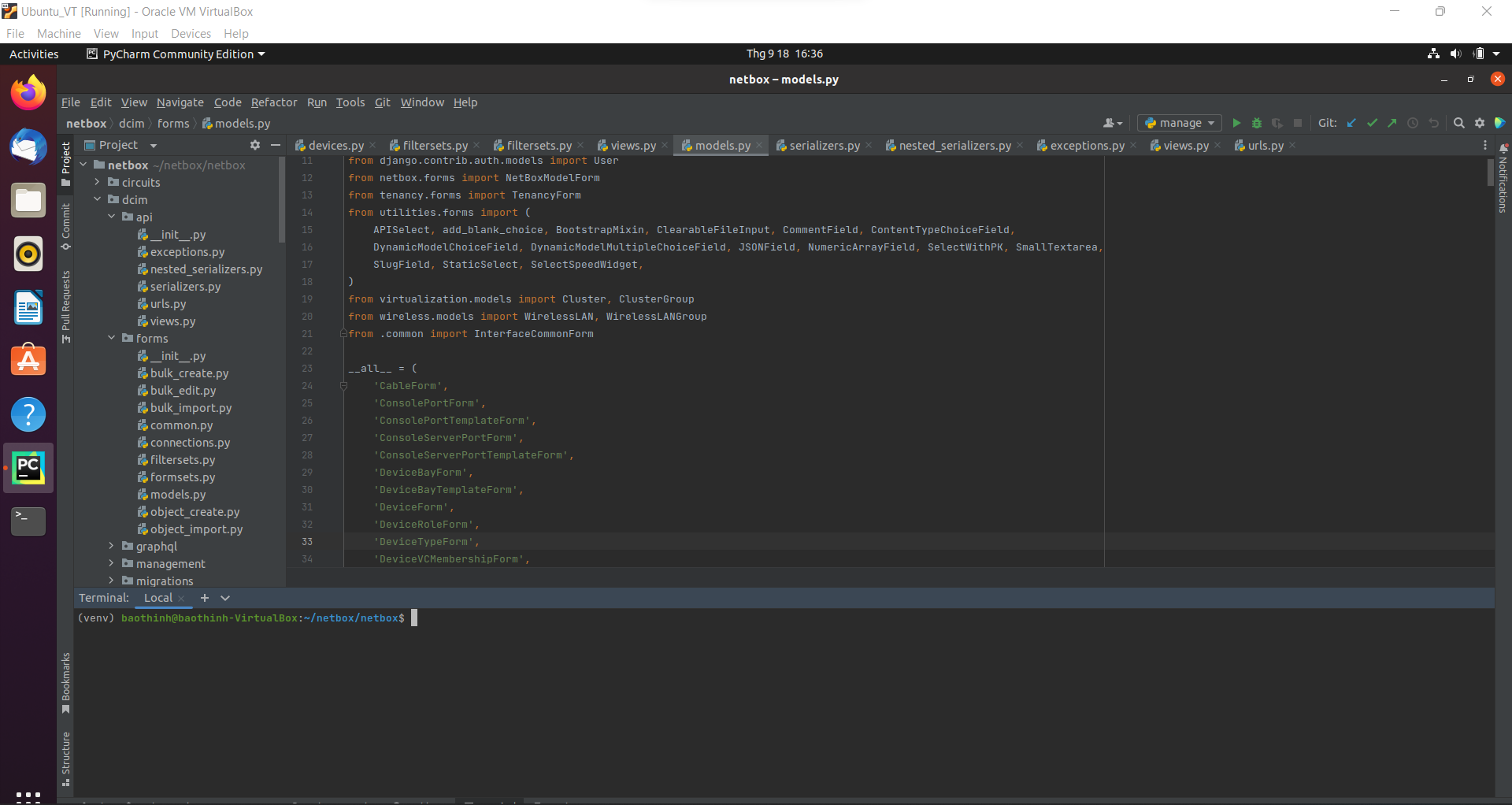
* Tải phần mềm Virtual Box và Ubuntu:
  + <https://www.virtualbox.org/wiki/Downloads>
  + <https://ubuntu.com/download/desktop>
  + Version hiện tại sử dụng là Ubuntu 20.04 và Virtual Box 6.1.34
* Tạo máy ảo bằng Virtual Box với hệ điều hành Linux (Ubuntu):



*Hình ảnh 2: Cấu hình máy ảo virtualbox/Ubuntu đã tạo*

* Cài đặt Pycharm trên máy ảo giúp dễ dàng debug xử lý thao tác trong project theo mục [4]:

$sudo snap install pycharm-community --classic



*Hình ảnh 3: Sau khi cài đặt Pycharm trên máy ảo*

1. **Cài đặt và thiết lập khởi tạo một app Django đơn giản**
2. **Cài đặt Django và Python**
   * Update hệ thống:

$ sudo apt update

* + Tải gói Python:

$ sudo apt install python3-pip python3-venv

$ git clone git://github.com/django/django ~/django-dev

$ cd ~/django-dev

* + Tạo môi trường ảo phụ thuộc virtualenv:

$ sudo apt install python3-venv

* + Tạo môi trường ảo phụ thuộc virtualenv:

$ python3 -m venv /path/to/venv

$ source /path/to/venv/bin/activate

* + Kiểm tra version:

$ $ pip install -U pip

* + Tải Django:

$ pip install -e ~/django-dev

$ django-admin --version

1. **Tạo app Django cơ bản**
   * Tạo project:

$ django-admin startproject mysite

* + Các file được tạo:

mysite/

manage.py

mysite/

\_\_init\_\_.py

settings.py

urls.py

asgi.py

wsgi.py

* + Ghi chú:
  + Thư mục gốc mysite/ ngoài cùng là container của project.
  + Thư mục mysite/ bên trong là package Python thực tế cho project
  + mysite/ \_\_ init \_\_ .py: là một file rỗng để đánh dấu đây là package của Python.
  + mysite/settings.py: Cài đặt/ cấu hình cho project này.
  + mysite/urls.py: Các khai báo URL cho project này, đóng vai trò làm router để hướng dẫn các request đến file xử lý phù hợp.
  + mysite/asgi.py và mysite/wsgi.py: file cấu hình cho server
  + Chạy server:

$ python manage.py runserver

* + Kết quả:

Performing system checks...

System check identified no issues (0 silenced).

You have unapplied migrations; your app may not work properly until they are applied.

Run 'python manage.py migrate' to apply them.

September 09, 2022 - 15:50:53

Django version 4.0, using settings 'mysite.settings'

Starting development server at <http://127.0.0.1:8000/>

Quit the server with CONTROL-C.

* + Tạo Poll app:

$ python manage.py startapp polls

* Tạo view cơ bản
* Ta cần điều hướng để có thể gọi view này -> chỉnh url
* Để load được poll/urls.py vào trong mysite/urls.py thì ta phải thông qua lệnh “include“, muốn sử dụng được lệnh include thì bạn phải import nó từ packages “urls”
* Thư mục mysite/urls.py hiện tại:

from django.contrib import admin

from django.urls import include, path

urlpatterns = [

path('polls/', include('polls.urls')),

path('admin/', admin.site.urls),

]

* + Thiết lập cơ sở dữ liệu:
  + Mở file mysite/settings.py.
  + Theo mặc định, cấu hình sử dụng SQLite.

$ sudo apt update

* + Nếu muốn sử dụng cơ sở dữ liệu khác, thay đổi từ khóa:
  + ENGINE -> 'django.db.backends.sqlite3', 'django.db.backends.postgresql', 'django.db.backends.mysql', or 'django.db.backends.oracle' etc.
  + NAME : Tên cơ sở dữ liệu
* Nếu không sử dụng SQLite làm cơ sở dữ liệu của mình, các cài đặt bổ sung như USER, PASSWORDvà HOST phải được thêm vào.
* INSTALLED\_APPS chứa tên của tất cả các ứng dụng Django được kích hoạt trong phiên bản Django này.
* django.contrib.admin- Trang web quản trị.
* django.contrib.auth- Một hệ thống xác thực.
* django.contrib.contenttypes- Một khung cho các loại nội dung.
* django.contrib.sessions- Một khung phiên.
* django.contrib.messages- Một khung nhắn tin.
* django.contrib.staticfiles- Một khung để quản lý các tệp tĩnh.
* Lệnh migrate sẽ tạo các cơ sở dữ liệu cần thiết dựa trên co sở dữ liệu ở mysite/settings.py
* Tạo model:
* Tạo model trong polls/models.py
* Với poll app chỉ cần 2 model Question and Choice
* Thư mục sau khi sửa

from django.db import models

class Question(models.Model):

question\_text = models.CharField(max\_length=200)

pub\_date = models.DateTimeField('date published')

class Choice(models.Model):

question = models.ForeignKey(Question, on\_delete=models.CASCADE)

choice\_text = models.CharField(max\_length=200)

votes = models.IntegerField(default=0)

* Mỗi mô hình được đại diện bởi một lớp phân lớp con django.db.models.Model. Mỗi mô hình có một số biến lớp, mỗi biến đại diện cho một trường cơ sở dữ liệu trong mô hình.
* Mỗi trường được đại diện bởi một thể hiện của một lớp Field
* Kích hoạt model:
* Thêm 'polls.apps.PollsConfig' vào mysite/settings.py để đưa ứng dụng vào project
* Thư mục sau khi sửa

INSTALLED\_APPS = [

'polls.apps.PollsConfig',

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

]

* Chạy lệnh:

$ python manage.py makemigrations polls

* Kết quả:

Migrations for 'polls':

polls/migrations/0001\_initial.py

- Create model Question

- Create model Choice

* Khi chạy lệnh makemigrations tức là ta đang thực hiện một số thay đổi trong model và các thay đổi đấy lưu ở dạng migration ( phần này em không hiểu lắm )
* Lệnh sqlmigrate lấy tên migration và trả về SQL mà migration chạy.( Ở đây sử dụng PostgreSQL )

$ python manage.py sqlmigrate polls 0001

* Chạy lệnh migrate để tạo các model trong cơ sở dữ liệu
* Ghi chú: Chạy python manage.py makemigrations để tạo migrations cho những thay đổi. Chạy python manage.py migrate lưu những thay đổi vào database.
* Cài đặt API:

$ python manage.py shell

>>> from polls.models import Choice, Question # Import the model classes we just wrote.

# No questions are in the system yet.

>>> Question.objects.all()

<QuerySet []>

# Create a new Question.

# Support for time zones is enabled in the default settings file, so

# Django expects a datetime with tzinfo for pub\_date. Use timezone.now()

# instead of datetime.datetime.now() and it will do the right thing.

>>> from django.utils import timezone

>>> q = Question(question\_text="What's new?", pub\_date=timezone.now())

# Save the object into the database. You have to call save() explicitly.

>>> q.save()

# Now it has an ID.

>>> q.id

1

# Access model field values via Python attributes.

>>> q.question\_text

"What's new?"

>>> q.pub\_date

datetime.datetime(2012, 2, 26, 13, 0, 0, 775217, tzinfo=<UTC>)

# Change values by changing the attributes, then calling save().

>>> q.question\_text = "What's up?"

>>> q.save()

# objects.all() displays all the questions in the database.

>>> Question.objects.all()

<QuerySet [<Question: Question object (1)>]>

* Sửa file polls/models.py:

from django.db import models

class Question(models.Model):

# ...

def \_\_str\_\_(self):

return self.question\_text

def was\_published\_recently(self):

return self.pub\_date >= timezone.now() - datetime.timedelta(days=1)

class Choice(models.Model):

# ...

def \_\_str\_\_(self):

return self.choice\_text

* Lưu thay đổi:

>>> from polls.models import Choice, Question

# Make sure our \_\_str\_\_() addition worked.

>>> Question.objects.all()

<QuerySet [<Question: What's up?>]>

# Django provides a rich database lookup API that's entirely driven by

# keyword arguments.

>>> Question.objects.filter(id=1)

<QuerySet [<Question: What's up?>]>

>>> Question.objects.filter(question\_text\_\_startswith='What')

<QuerySet [<Question: What's up?>]>

# Get the question that was published this year.

>>> from django.utils import timezone

>>> current\_year = timezone.now().year

>>> Question.objects.get(pub\_date\_\_year=current\_year)

<Question: What's up?>

# Request an ID that doesn't exist, this will raise an exception.

>>> Question.objects.get(id=2)

Traceback (most recent call last):

...

DoesNotExist: Question matching query does not exist.

# Lookup by a primary key is the most common case, so Django provides a

# shortcut for primary-key exact lookups.

# The following is identical to Question.objects.get(id=1).

>>> Question.objects.get(pk=1)

<Question: What's up?>

# Make sure our custom method worked.

>>> q = Question.objects.get(pk=1)

>>> q.was\_published\_recently()

True

# Give the Question a couple of Choices. The create call constructs a new

# Choice object, does the INSERT statement, adds the choice to the set

# of available choices and returns the new Choice object. Django creates

# a set to hold the "other side" of a ForeignKey relation

# (e.g. a question's choice) which can be accessed via the API.

>>> q = Question.objects.get(pk=1)

# Display any choices from the related object set -- none so far.

>>> q.choice\_set.all()

<QuerySet []>

# Create three choices.

>>> q.choice\_set.create(choice\_text='Not much', votes=0)

<Choice: Not much>

>>> q.choice\_set.create(choice\_text='The sky', votes=0)

<Choice: The sky>

>>> c = q.choice\_set.create(choice\_text='Just hacking again', votes=0)

# Choice objects have API access to their related Question objects.

>>> c.question

<Question: What's up?>

# And vice versa: Question objects get access to Choice objects.

>>> q.choice\_set.all()

<QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>

>>> q.choice\_set.count()

3

# The API automatically follows relationships as far as you need.

# Use double underscores to separate relationships.

# This works as many levels deep as you want; there's no limit.

# Find all Choices for any question whose pub\_date is in this year

# (reusing the 'current\_year' variable we created above).

>>> Choice.objects.filter(question\_\_pub\_date\_\_year=current\_year)

<QuerySet [<Choice: Not much>, <Choice: The sky>, <Choice: Just hacking again>]>

# Let's delete one of the choices. Use delete() for that.

>>> c = q.choice\_set.filter(choice\_text\_\_startswith='Just hacking')

>>> c.delete()

* Chỉnh sửa giao diện và thêm đường dẫn:
* Sửa file polls/views.py:

def detail(request, question\_id):

return HttpResponse("You're looking at question %s." % question\_id)

def results(request, question\_id):

response = "You're looking at the results of question %s."

return HttpResponse(response % question\_id)

def vote(request, question\_id):

return HttpResponse("You're voting on question %s." % question\_id)

* Sửa file polls/url.py:

from django.urls import path

from . import views

urlpatterns = [

# ex: /polls/

path('', views.index, name='index'),

# ex: /polls/5/

path('<int:question\_id>/', views.detail, name='detail'),

# ex: /polls/5/results/

path('<int:question\_id>/results/', views.results, name='results'),

# ex: /polls/5/vote/

path('<int:question\_id>/vote/', views.vote, name='vote'),

]

* Mỗi view chịu trách nghiệm cho việc trả về HttpResponse khi việc kết nối hoàn tất hoặc trả về Http404 khi xuất hiện lỗi.
* Sửa file polls/view.py:

from django.http import HttpResponse

from .models import Question

def index(request):

latest\_question\_list = Question.objects.order\_by('-pub\_date')[:5]

output = ', '.join([q.question\_text for q in latest\_question\_list])

return HttpResponse(output)

# Leave the rest of the views (detail, results, vote) unchanged

* Đặt Template:

{% if latest\_question\_list %}

<ul>

{% for question in latest\_question\_list %}

<li><a href="{% url 'polls:detail' question.id %}">{{ question.question\_text }}</a></li>

{% endfor %}

</ul>

{% else %}

<p>No polls are available.</p>

{% endif %}

* Update index view:

from django.http import HttpResponse

from django.template import loader

from .models import Question

def index(request):

latest\_question\_list = Question.objects.order\_by('- pub\_date')[:5]

template = loader.get\_template('polls/index.html')

context = {

'latest\_question\_list': latest\_question\_list,

}

return HttpResponse(template.render(context, request))

* Hiển thị Http404:
* Chỉnh sửa file: polls/view.py ```bash from django.http import Http404 from django.shortcuts import render

from Django.http import Http404

from Django.shortcuts import render

from .models import Question

# ...

def detail(request, question\_id):

try:

question = Question.objects.get(pk=question\_id)

except Question.DoesNotExist:

raise Http404("Question does not exist")

return render(request, 'polls/detail.html', {'question': question})

* Tạo polls/template/polls/detail.html template:

from django.urls import path

from . import views

app\_name = 'polls'

urlpatterns = [

path('', views.index, name='index'),

path('<int:question\_id>/', views.detail, name='detail'),

path('<int:question\_id>/results/', views.results, name='results'),

path('<int:question\_id>/vote/', views.vote, name='vote'),

]

* + Tối giản hóa code (Ở đây ta áp dụng các cách tối giản của tutorial04). [12] Vì ở đây em đã thực hiện xong việc tối giản hóa nên không có dòng code cũ để so sánh quá trình sửa đổi.
  + Tạo các Test cho app (Optional)
  + Sử dụng Css cho web (Optianal)
  + Chỉnh sửa mẫu admin :

from django.contrib import admin

# Register your models here.

from django.contrib import admin

from .models import Choice,Question

class ChoiceInline(admin.TabularInline):

model = Choice

extra = 3

class QuestionAdmin(admin.ModelAdmin):

fieldsets = [

(None, {'fields': ['question\_text']}),

('Date information', {'fields': ['pub\_date'], 'classes': ['collapse']}),

]

inlines = [ChoiceInline]

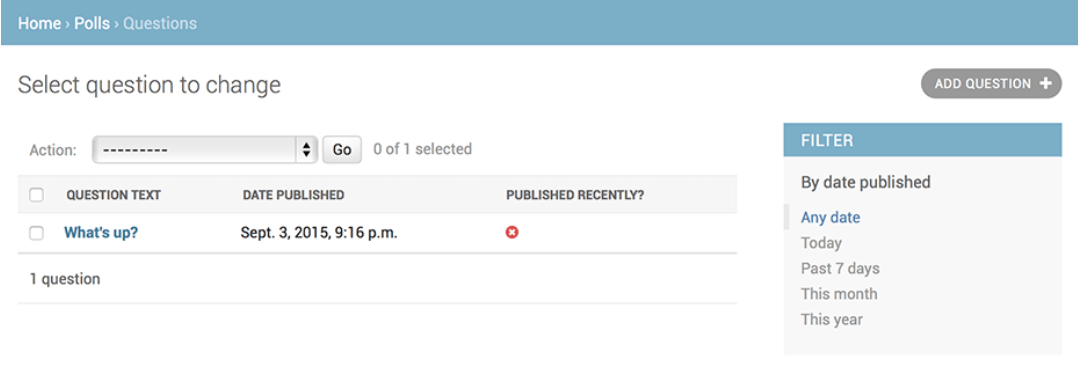
list\_display = ('question\_text', 'pub\_date', 'was\_published\_recently')

list\_filter = ['pub\_date']

search\_fields = ['question\_text']

admin.site.register(Question, QuestionAdmin)

* + Kết quả:



*Hình ảnh 4: Giao diện sau khi tạo xong một app vote web*



*Hình ảnh 5: Giao diện khi sửa các choices*

1. **Cài đặt và thiết lập NetBox**
   1. **PostgreSQL Database**
      * Cài đặt PostgreSQL Database:

$ sudo apt update

$ sudo apt install -y postgresql

* + - Khởi động hệ thống sau khi tải xong:

$ sudo systemctl start postgresql

$ sudo systemctl enable postgresql

* + - Kiểm tra version mới nhất chưa:

$ psql -V

* + - Tạo database :

$sudo -u postgres psql

CREATE DATABASE netbox;

CREATE USER netbox WITH PASSWORD 'J5brHrAXFLQSif0K'

GRANT ALL PRIVILEGES ON DATABASE netbox TO netbox;

\q để thoát

* 1. **Redis**
     + Cài đặt Redis:

$ sudo apt install –y redis-server

$ sudo apt install -y postgresql

* + - Kiểm tra version trên 4.0 chưa:

$ redis-server - v

* + - Kiểm tra tình trạng hệ thống:

$ redis-cli ping

* + - Server sẽ gửi lại tin nhắn “PONG” nếu kiểm tra thành công
    - Có thể kiểm tra cấu hình trên file /etc/redis.conf hoặc /etc/redis/redis.conf
  1. **Netbox components**
     + Cài đặt các gói hệ thống:

$ sudo apt install -y python3 python3-pip python3-venv python3-dev build-essential libxml2-dev libxslt1-dev libffi-dev libpq-dev libssl-dev zlib1g-dev

* + - Tải Netbox:

$ sudo mkdir -p /opt/netbox/

$ cd /opt/netbox/

* + - Tải git nếu chưa tải:

$ sudo apt install -y git

* + - Gitclone link github:

$ sudo git clone -b master --depth 1 https://github.com/netbox-community/netbox.git

* + - Kết quả sau khi gitclone:

Cloning into '.'...

remote: Enumerating objects: 996, done.

remote: Counting objects: 100% (996/996), done.

remote: Compressing objects: 100% (935/935), done.

remote: Total 996 (delta 148), reused 386 (delta 34), pack-reused 0

Receiving objects: 100% (996/996), 4.26 MiB | 9.81 MiB/s, done.

Resolving deltas: 100% (148/148), done.

* + - Tạo User cho hệ thống Netbox:

$ sudo adduser --system --group netbox

$ sudo chown --recursive netbox /opt/netbox/netbox/media/

* + - Vào file cấu hình Configuration.py:

$ cd /opt/netbox/netbox/netbox/

$ sudo cp configuration\_example.py configuration.py

* + - Sau khi vào file cấu hình Configuration.py sẽ có 4 thứ cần chỉnh cho một User mới:
* ALLOWED\_HOSTS
* DATABASE
* REDIS
* SECRET\_KEY
  + - Cấu hình:

ALLOWED\_HOSTS = ['netbox.example.com', '192.0.2.123']

ALLOWED\_HOSTS = ['\*']

DATABASE = {

'NAME': 'netbox', # Database name

'USER': 'netbox', # PostgreSQL username

'PASSWORD': 'secret', # PostgreSQL password

'HOST': 'localhost', # Database server

'PORT': '', # Database port (leave blank for default)

'CONN\_MAX\_AGE': 300, # Max database connection age (seconds)

}

REDIS = {

'tasks': {

'HOST': 'localhost', # Redis server

'PORT': 6379, # Redis port

'PASSWORD': '', # Redis password (optional)

'DATABASE': 0, # Database ID

'SSL': False, # Use SSL (optional)

},

'caching': {

'HOST': 'localhost',

'PORT': 6379,

'PASSWORD': '',

'DATABASE': 1, # Unique ID for second database

'SSL': False,

}

}

* + - Khởi tạo Secret key:

python3 ../generate\_secret\_key.py

* + - Save sau file cấu hình sau khi thay đổi:
  1. **Gunicorn**
     + Copy file gunicorn.py để phòng bị viết đề mà không sử dụng lại được:

$ sudo cp /opt/netbox/contrib/gunicorn.py /opt/netbox/gunicorn.py

* + - Copy file netbox.service và netbox-rq.service và khởi tại lại:

$ sudo cp -v /opt/netbox/contrib/\*.service /etc/systemd/system/

$ sudo systemctl daemon-reload

* + - Khởi động hệ thống netbox. và netbox-rq :

$ sudo systemctl start netbox netbox-rq

$ sudo systemctl enable netbox netbox-rq

* + - Kiểm tra WSGI đã chạy chưa:

$ systemctl status netbox.service

* + - Kết quả sau khi kiểm tra:

netbox.service - NetBox WSGI Service

Loaded: loaded (/etc/systemd/system/netbox.service; enabled; vendor preset: enabled)

Active: active (running) since Mon 2021-08-30 04:02:36 UTC; 14h ago

Docs: https://docs.netbox.dev/

Main PID: 1140492 (gunicorn)

Tasks: 19 (limit: 4683)

Memory: 666.2M

CGroup: /system.slice/netbox.service

├─1140492 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /va>

├─1140513 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /va>

├─1140514 /opt/netbox/venv/bin/python3 /opt/netbox/venv/bin/gunicorn --pid /va>

…

* 1. **HTTP server**
     + Tải nginx:

$ sudo apt install -y nginx

* + - Copy cấu hình nginx được tạo bởi netbox vào /etc/nginx/sites-available/netbox :

$ sudo cp /opt/netbox/contrib/nginx.conf /etc/nginx/sites-available/netbox

* + - Xóa /etc/nginx/sites-enabled/default và tạo symlink ở file sites-enabled tới file cấu hình vừa tạo đồng thời restart lại để sử dụng cấu hình mới :

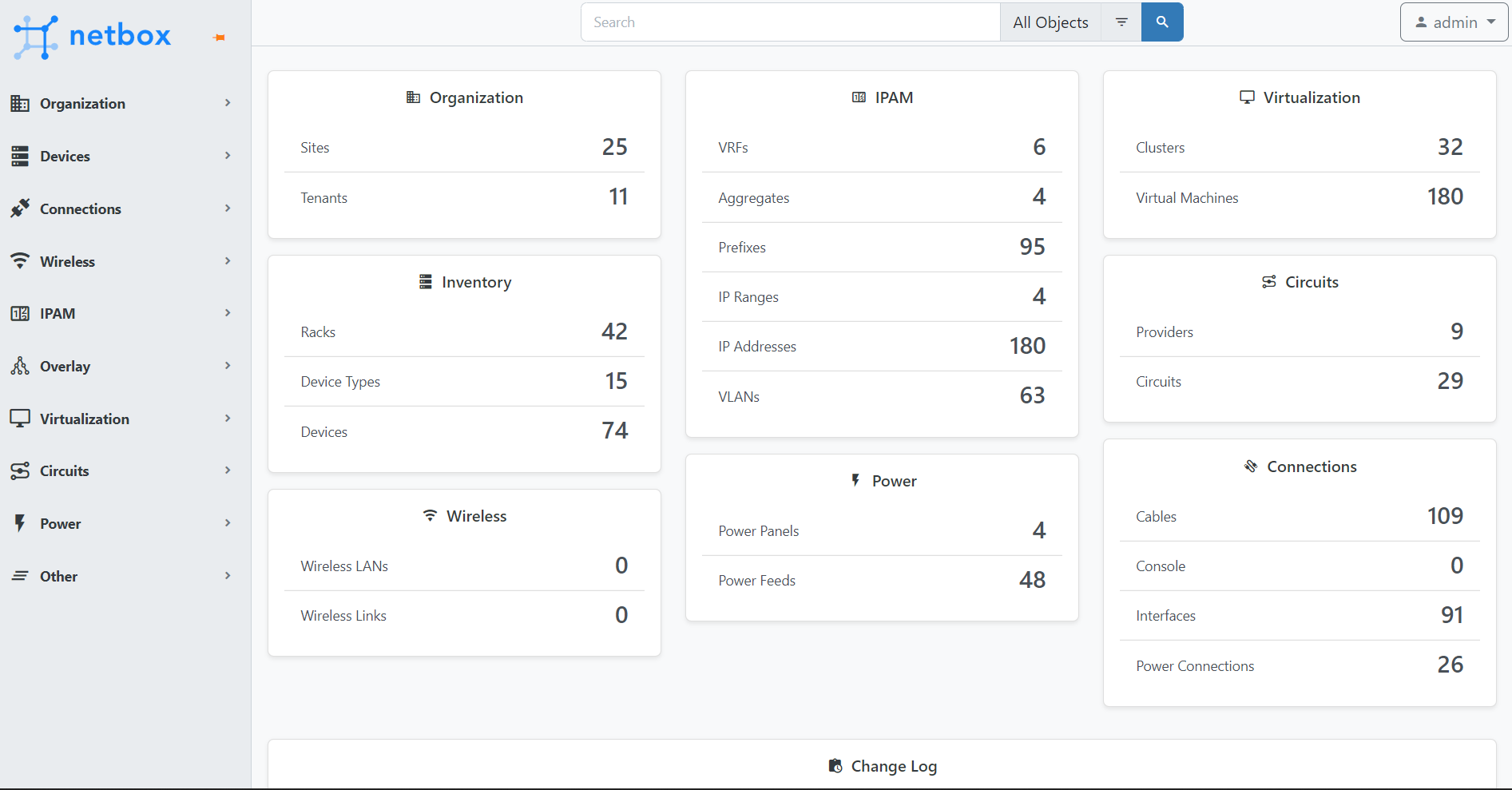
$ sudo rm /etc/nginx/sites-enabled/default

$ sudo ln -s /etc/nginx/sites-available/netbox /etc/nginx/sites-enabled/netbox

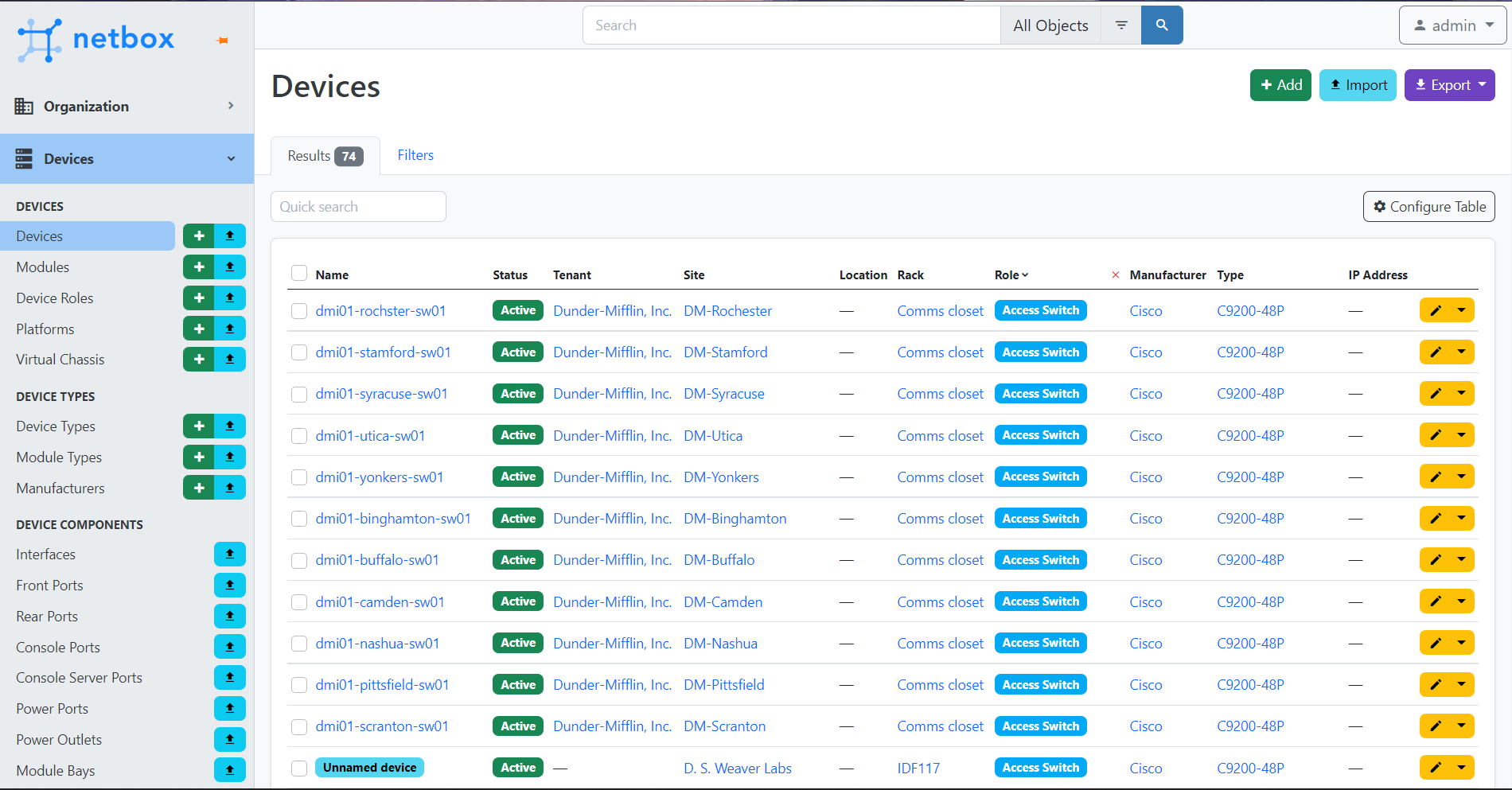
$ sudo systemctl restart nginx

* 1. **LDAP authentication (optional)**

1. **Phát triển tính năng quản lý Storage trên NetBox**

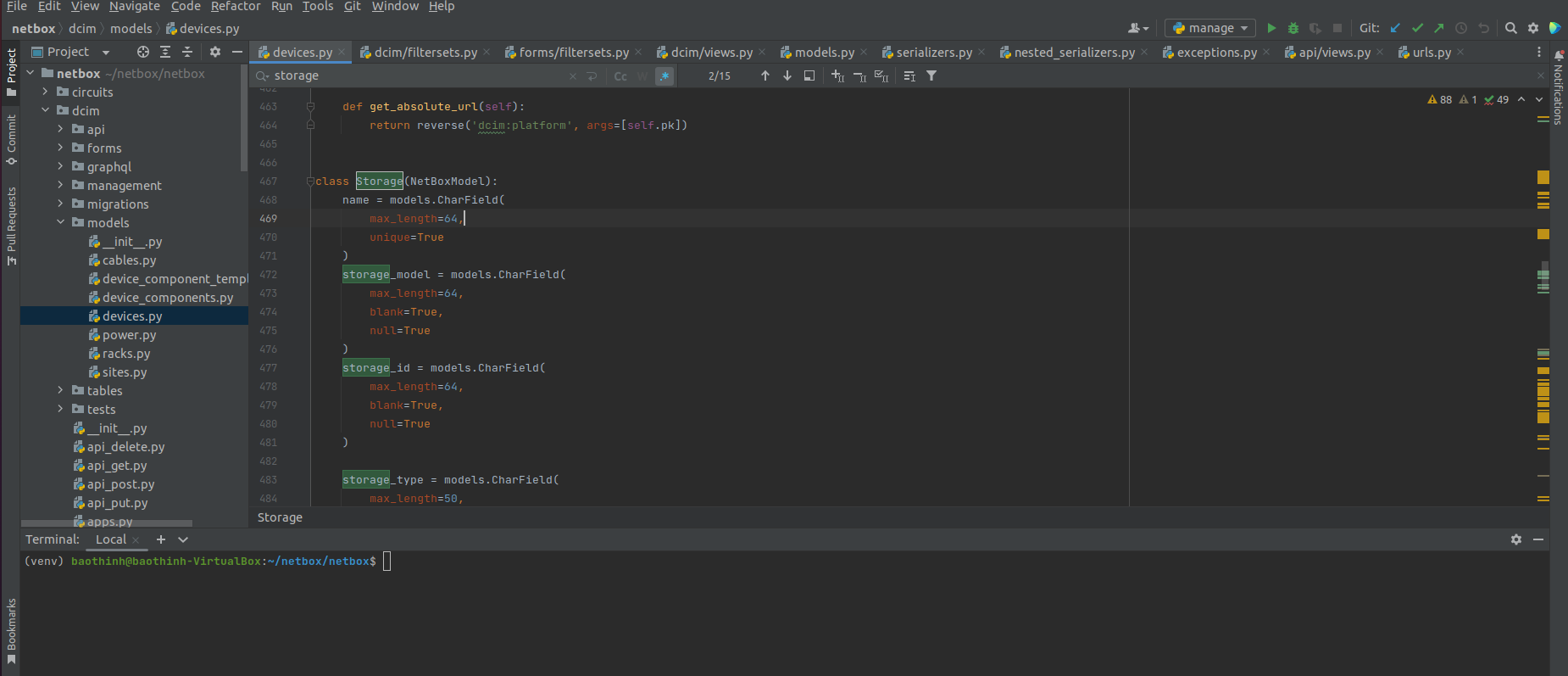


*Hình ảnh 6: Demo của NetBox có sẵn*



*Hình ảnh 7:Giao diện của Device trong NetBox*

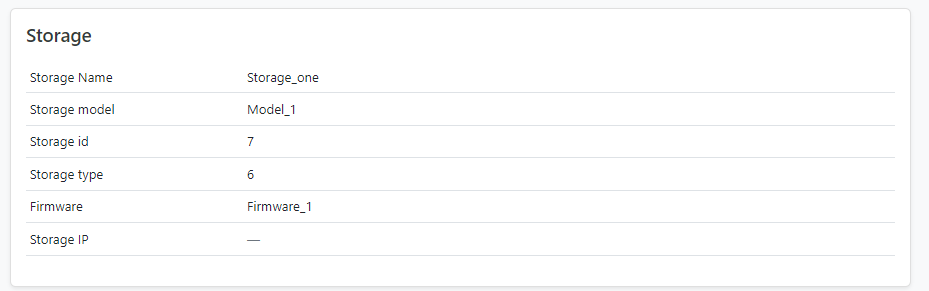
* + - Việc tạo Model cho Storage sẽ giống với Devices nên ảnh trên sẽ là một ví dụ.
    - Vì trong Django và netbox có hỗ trợ rất nhiều thư viện cũng như các hàm nên em không đào sâu vào các hàm gốc.
    - Mục tiêu là tạo được 1 mục Storage tương tự Device và có các tính năng tương tự
    - Chú ý các hình ảnh trước phần kết quả chỉ để mô tả dòng lệnh code sử dụng để hiển thị gì hay chức năng các dòng lệnh chứ không chỉ dùng mỗi dòng lệnh mô tả mà chạy được như ảnh. Ta cần xử lý tất cả các dòng lệnh ghép với nhau thì phần mềm mới chạy.
  1. **Tạo model**
     + Việc đầu tiên chúng ta phải tạo thêm model trên code có sẵn:
     + Các model cần tạo:
       - name: Kiểu text, tối đa 64 ký tự, phải unique, bắt buộc phải nhập
       - storage\_model = Kiểu text, tối đa 64 ký tự, không bắt buộc phải nhập
       - storage\_id = Kiểu text, tối đa 64 ký tự, không bắt buộc phải nhập
       - storage\_type = Không bắt buộc phải nhập, cho phép người dùng chọn các option sau:
         * STORAGE\_TYPE\_DELL\_SC8000 = 1
         * STORAGE\_TYPE\_IBM\_V7000 = 2
         * STORAGE\_TYPE\_HITACHI\_G700 = 3
         * STORAGE\_TYPE\_DELL\_UNITY = 4
         * STORAGE\_TYPE\_HPE\_MSA = 5
         * STORAGE\_TYPE\_HPE\_3PAR = 6
         * STORAGE\_TYPE\_IBM\_SVC = 7
         * STORAGE\_TYPE\_OTHER = 8
       - manager = không bắt buộc, cài đặt như manager của device
       - management\_ip = Bắt buộc nhập, như trường IP của Device
       - firmware = Không bắt buộc nhập, kiểu text, tối đa 256 ký tự
     + Ở đây ta mượn file dcim/model.py để viết model cho storage:



*Hình ảnh 8: File dcim/model được mở trên pycharm*

* + - Tạo hàm Class Storage chứa các model và được định dạng theo yêu cầu trên:

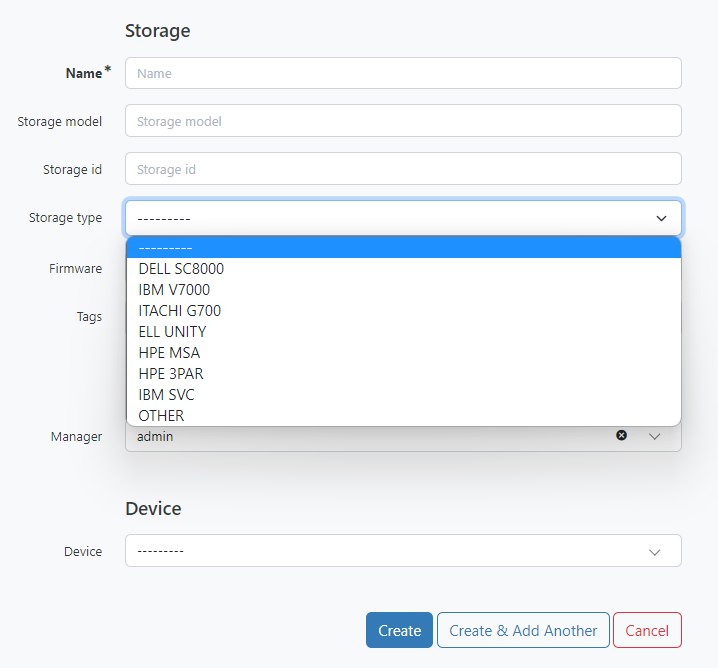
class Storage(NetBoxModel):  
 name = models.CharField(  
 max\_length=64,  
 unique=True  
 )  
 storage\_model = models.CharField(  
 max\_length=64,  
 blank=True,  
 null=True  
 )  
 storage\_id = models.CharField(  
 max\_length=64,  
 blank=True,  
 null=True  
 )  
  
 storage\_type = models.CharField(  
 max\_length=50,  
 choices=StorageTypeChoices,  
 blank=True,  
 null=True  
 )  
  
 firmware = models.CharField(  
 max\_length=256,  
 blank=True,  
 null=True  
 )  
  
 status = models.CharField(  
 max\_length=50,  
 choices=DeviceStatusChoices,  
 default=DeviceStatusChoices.STATUS\_ACTIVE  
 )  
  
 manager = models.ForeignKey(  
 to='auth.User',  
 on\_delete=models.SET\_NULL,  
 related\_name='storages',  
 blank=True,  
 null=True  
 )  
  
 device = models.OneToOneField(  
 to='dcim.Device',  
 on\_delete=models.PROTECT,  
 related\_name='storages',  
 blank = True,  
 null = True  
 )  
  
 primary\_ip4 = models.OneToOneField(  
 to='ipam.IPAddress',  
 on\_delete=models.SET\_NULL,  
 related\_name='+',  
 blank=True,  
 null=True,  
 verbose\_name='Primary IPv4'  
 )  
  
 def get\_absolute\_url(self):  
 return reverse('dcim:storage', args=[self.pk])

****

*Hình ảnh 9: Giao diện của model 1 Storage bất kì*

* + - Trong hàm Storage đa phần các model được định dạng kiểu Char nhưng có hàm IP và device được định dạng onetoonefield tức là chỉ có 1 IP với 1 device liên kết với 1 storage duy nhất và model manager là dạng khóa ngoại để tạo ra liên kết giữa một manager có thể tạo ra nhiều danh mục storage
    - Lệnh blank = True để cho phép model đấy không bắt buộc nhập
    - Lệnh Max\_length để tạo model có bao nhiêu ký tự
    - Hàm get\_absolute\_url để trả về url của storage
    - Ở đây ta phải tạo thêm hàm StorageTypeChoice để xử lý với yêu cầu tạo model ban đầu tức là có 8 lựa chọn cho model storage type, ta vào file /dcim/choice.py để chỉnh sửa thêm
  1. **Tạo ChoiceSet**

class StorageTypeChoices(ChoiceSet):  
  
 STORAGE\_TYPE\_DELL\_SC8000 = '1'  
 STORAGE\_TYPE\_IBM\_V7000 = '2'  
 STORAGE\_TYPE\_HITACHI\_G700 = '3'  
 STORAGE\_TYPE\_DELL\_UNITY = '4'  
 STORAGE\_TYPE\_HPE\_MSA = '5'  
 STORAGE\_TYPE\_HPE\_3PAR = '6'  
 STORAGE\_TYPE\_IBM\_SVC = '7'  
 STORAGE\_TYPE\_OTHER = '8'  
  
 CHOICES = [  
 (STORAGE\_TYPE\_DELL\_SC8000, 'DELL SC8000'),  
 (STORAGE\_TYPE\_IBM\_V7000, 'IBM V7000'),  
 (STORAGE\_TYPE\_HITACHI\_G700, 'ITACHI G700'),  
 (STORAGE\_TYPE\_DELL\_UNITY, 'ELL UNITY'),  
 (STORAGE\_TYPE\_HPE\_MSA , 'HPE MSA'),  
 (STORAGE\_TYPE\_HPE\_3PAR, 'HPE 3PAR'),  
 (STORAGE\_TYPE\_IBM\_SVC, 'IBM SVC'),  
 (STORAGE\_TYPE\_OTHER, 'OTHER'),  
 ]



*Hình ảnh 10: Các Choices được tạo khi xuất hiện trên giao diện*

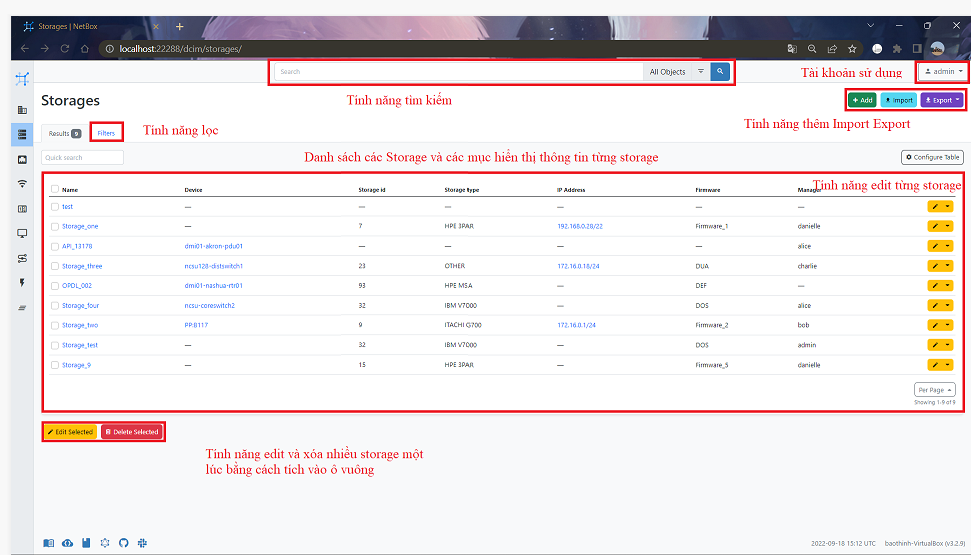
* + - Sau khi tạo xong model cho Storage ta save bằng lệnh sau

$ python3 manage.py makemigrations

$ python3 manage.oy migrate

* 1. **Tạo View**
     + Tiếp theo ta thêm view cho model của mình vào /dcim/view.py để viết

class StorageListView(generic.ObjectListView):  
 queryset = Storage.objects.all()  
 filterset = filtersets.StorageFilterSet  
 filterset\_form = forms.StorageFilterForm  
 table = tables.StorageTable  
  
class StorageEditView(generic.ObjectEditView):  
 queryset = Storage.objects.all()  
 form = forms.StorageForm  
 # model = Storage  
  
 def alter\_object(self, obj, request, args, kwargs):  
 if not obj.pk:  
 if 'storage' in request.GET:  
 obj.storage = get\_object\_or\_404(Storage, pk=request.GET.get('storage'))  
 obj.manager = request.user  
 return obj  
  
 def \_save\_obj(self, obj\_form, request):  
 *"""  
 Assign the currently authenticated user to the Storage.  
 """* instance = obj\_form.save(commit=False)  
 instance.manager = request.user  
 instance.save()  
  
 return instance  
  
class StorageBulkImportView(generic.BulkImportView):  
 queryset = Storage.objects.all()  
 model\_form = forms.StorageCSVForm  
 table = tables.StorageTable  
  
 def \_save\_obj(self, obj\_form, request):  
 *"""  
 Assign the currently authenticated user to the Storage.  
 """* instance = obj\_form.save(commit=False)  
 instance.manager = request.user  
 instance.save()  
 return instance  
  
class StorageBulkEditView(generic.BulkEditView):  
 queryset = Storage.objects.all()  
 table = tables.StorageTable  
 filterset = filtersets.StorageFilterSet  
 form = forms.StorageBulkEditForm  
  
class StorageBulkDeleteView(generic.BulkDeleteView):  
 queryset = Storage.objects.all()  
 filterset = filtersets.StorageFilterSet  
 table = tables.StorageTable  
  
class StorageView(generic.ObjectView):  
 queryset = Storage.objects.all()  
  
class StorageDeleteView(generic.ObjectDeleteView):  
 queryset = Storage.objects.all()



*Hình ảnh 11: Các tính năng được tạo*

* + - Ở đây ta phải tạo các class:
      * StorageListView tạo view cho danh sách
      * StorageEditView tạo view cho phần edit từng storage
      * StorageBulkImportView tạo view cho phần import
      * StorageBulkEditView tạo view cho phần edit nhiều storage
      * StorageBulkDeleteView tạo view cho phần delete nhiều storage
      * StorageView tạo view cho giao diện storage
      * StorageDeleteView tạo view cho phần delete từng storage
    - Các class được nhập tất cả các objects của Strorage quả lệnh queryset = Sttorage.objects.all()
    - Đồng thời thêm các tính năng cho các mục ta phải tạo thêm StorageForm,…
    - Các hàm save\_obj và alter\_obj để tự động lưu tên người tạo là tài khoản đang sử dụng để tạo thêm storage cho model manager
    - Tạo bộ Filterset tại dcim/filter.py:

class StorageFilterSet(NetBoxModelFilterSet):  
 primary\_ip4 = django\_filters.ModelMultipleChoiceFilter(  
 queryset=IPAddress.objects.all(),  
 label='IPAddress (ID)',  
 )  
 manager = django\_filters.ModelMultipleChoiceFilter(  
 queryset=User.objects.all(),  
 label='Manager',  
 )  
 class Meta:  
 model = Storage  
 fields = ['id', 'name']  
 def search(self, queryset, name, value):  
 if not value.strip():  
 return queryset  
 return queryset.filter(  
 Q(name\_\_icontains=value)  
 )

* + - Tạo một Đơn Filter tại dcim/form/filterset.py

class StorageFilterForm(NetBoxModelFilterSetForm):  
 model = Storage  
 fieldsets = (  
 (None, ('q', 'tag',)),  
 ('User', ('manager',)),  
 ('IP address', ('primary\_ip4',)),  
 )  
 manager = DynamicModelMultipleChoiceField(  
 queryset=User.objects.all(),  
 required=False,  
 label=\_('Manager'),  
 widget=APISelectMultiple(  
 api\_url='/api/users/users/',  
 )  
 )  
 primary\_ip4 = DynamicModelMultipleChoiceField(  
 queryset=IPAddress.objects.all(),  
 required=False,  
 label=\_('IPAddress'),  
 )  
  
 tag = TagFilterField(model)

* + - 2 phần này sẽ hỗ trợ việc lọc và tìm kiếm thông tin khi có quá nhiều storage trên danh sách
    - Ta tạo StorageTable để hiển thể các model trên danh mục Storage trong dcim/tables/device.py

class StorageTable(NetBoxTable):  
 name = tables.Column(  
 linkify=True,  
 )  
  
 device = tables.Column(  
 linkify=True  
 )  
  
 primary\_ip4 = tables.Column(  
 linkify=True,  
 verbose\_name='IP Address'  
 )  
  
 class Meta(NetBoxTable.Meta):  
 model = Storage  
 fields = (  
 'pk', 'id', 'name','storage\_type', 'storage\_id', 'manager', 'device',  
 'primary\_ip4', 'firmware', 'description', 'tags',  
 )  
 default\_columns = (  
 'pk', 'name', 'device', 'storage\_id', 'storage\_type','primary\_ip4','firmware',  
 )

* + - Phần linkify để hiển thị đường dẫn có thể bấm chuột khi hiển thị trên danh sách
    - Ta tạo phần StorageForm cho phần edit cũng như tạo các Storage trong dcim/form/models.py:

class StorageForm(NetBoxModelForm):  
 manager = forms.ModelChoiceField(  
 queryset=User.objects.order\_by(  
 'username'  
 ),  
 widget=StaticSelect(),  
 required=False  
 )  
 device = DynamicModelChoiceField(  
 queryset=Device.objects.all(),  
 required=False  
 )  
  
 fieldsets = (  
 ('Storage', (  
 'name', 'storage\_model', 'storage\_id', 'storage\_type', 'firmware', 'tags'  
 )),  
 ('Manager', ('manager',)),  
 ('Device', ('device',)),  
 )  
  
 class Meta:  
 model = Storage  
 fields = (  
 'name', 'storage\_model', 'storage\_id', 'storage\_type', 'firmware', 'primary\_ip4' , 'manager' , 'device', 'tags'  
 )

* + - Tạo hàm Form cho phần import trong dcim/form/bulk\_import.py

class StorageCSVForm(NetBoxModelCSVForm):  
 class Meta:  
 model = Storage  
 fields = ( 'name', 'storage\_id', 'storage\_type','firmware', 'manager', 'device')

* + - Fieldset để tạo danh sách mục edit
  1. **Tạo đường dẫn url**
     + Tiếp tục ta tạo thêm đường dẫn cho các view của Storage trong file dcim/urls.py:

# Storage  
path('storages/', views.StorageListView.as\_view(), name='storage\_list'),  
path('storages/add/', views.StorageEditView.as\_view(), name='storage\_add'),  
path('storages/import/', views.StorageBulkImportView.as\_view(), name='storage\_import'),  
path('storages/edit/', views.StorageBulkEditView.as\_view(), name='storage\_bulk\_edit'),  
path('storages/delete/', views.StorageBulkDeleteView.as\_view(), name='storage\_bulk\_delete'),  
path('storages/<int:pk>/', views.StorageView.as\_view(), name='storage'),  
path('storages/<int:pk>/edit/', views.StorageEditView.as\_view(), name='storage\_edit'),  
path('storages/<int:pk>/delete/', views.StorageDeleteView.as\_view(), name='storage\_delete'),  
path('storages/<int:pk>/changelog/', ObjectChangeLogView.as\_view(), name='storage\_changelog', kwargs={'model': Storage}),

* 1. **Tạo API**
     + Việc tạo API giúp ta dễ dàng sử dụng được các tính năng giao tiếp của API như GET POST PUT DELETE.
     + Tạo Nested Serializer, Serializer để hỗ trợ việc GET, POST, PUT trong file dcim/api/nested\_serializer.py và dcim/api/serializer.py

class NestedStorageSerializer(WritableNestedSerializer):  
 url = serializers.HyperlinkedIdentityField(view\_name='dcim-api:storage-detail')  
 site\_count = serializers.IntegerField(read\_only=True)  
 \_depth = serializers.IntegerField(source='level', read\_only=True)  
  
 class Meta:  
 model = models.Storage  
 fields = ['id', 'url', 'display', 'name', 'site\_count', '\_depth']

class StorageSerializer(NetBoxModelSerializer):  
 url = serializers.HyperlinkedIdentityField(view\_name='dcim-api:storage-detail')  
 manager = NestedUserSerializer()  
 class Meta:  
 model = Storage  
 fields = (  
 'name', 'url', 'manager', 'device', 'storage\_model', 'storage\_id', 'storage\_type', 'firmware', 'tags',  
 )

* + - Tạo Storageviewset trong file dcim/api/view.py

class StorageViewSet(NetBoxModelViewSet):  
 queryset = Storage.objects.prefetch\_related(  
 'manager',  
 'device',  
 'primary\_ip4',  
 ).prefetch\_related('tags')  
 serializer\_class = serializers.StorageSerializer

* + - Chú ý tất cả các phần cần lưu lại hàm trên mục danh sách của file thì phải thêm vào danh sách nếu không sẽ không import được
  1. **Tạo File HTML để hiển thị trên web**
     + Ta cần tạo 3 file html để tạo giao diện cho từng Storage riêng lẻ, giao diện cho mục danh sách Storage, giao diện cho mục chỉnh sửa Storage.
     + Ta vào file netbox/templates/dcim tạo 3 file storage.html, storage\_list.html, storage\_edit.html
     + File storage.html:

{% extends 'generic/object\_list.html' %}  
  
{% block bulk\_buttons %}  
 {% if perms.dcim.change\_device %}  
 <div class="dropdown">  
 <button type="button" class="btn btn-sm btn-primary dropdown-toggle" data-bs-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
 <i class="mdi mdi-plus-thick" aria-hidden="true"></i> Add Components  
 </button>  
 <ul class="dropdown-menu">  
 {% if perms.dcim.add\_consoleport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_consoleport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Console Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_consoleserverport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_consoleserverport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item ">  
 Console Server Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_powerport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_powerport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Power Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_poweroutlet %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_poweroutlet' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Power Outlets  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_interface %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_interface' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}"  
 class="dropdown-item">Interfaces  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_rearport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_rearport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Rear Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_devicebay %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_devicebay' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Device Bays  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_modulebay %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_modulebay' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Module Bays  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_inventoryitem %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_inventoryitem' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Inventory Items  
 </button>  
 </li>  
 {% endif %}  
 </ul>  
 </div>  
 {% endif %}  
 {{ block.super }}  
{% endblock %}

* + - File storage\_list.html

{% extends 'generic/object\_list.html' %}  
  
{% block bulk\_buttons %}  
 {% if perms.dcim.change\_device %}  
 <div class="dropdown">  
 <button type="button" class="btn btn-sm btn-primary dropdown-toggle" data-bs-toggle="dropdown" aria-haspopup="true" aria-expanded="false">  
 <i class="mdi mdi-plus-thick" aria-hidden="true"></i> Add Components  
 </button>  
 <ul class="dropdown-menu">  
 {% if perms.dcim.add\_consoleport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_consoleport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Console Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_consoleserverport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_consoleserverport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item ">  
 Console Server Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_powerport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_powerport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Power Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_poweroutlet %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_poweroutlet' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Power Outlets  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_interface %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_interface' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}"  
 class="dropdown-item">Interfaces  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_rearport %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_rearport' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Rear Ports  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_devicebay %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_devicebay' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Device Bays  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_modulebay %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_modulebay' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Module Bays  
 </button>  
 </li>  
 {% endif %}  
 {% if perms.dcim.add\_inventoryitem %}  
 <li>  
 <button type="submit" formaction="{% url 'dcim:device\_bulk\_add\_inventoryitem' %}{% if request.GET %}?{{ request.GET.urlencode }}{% endif %}" class="dropdown-item">  
 Inventory Items  
 </button>  
 </li>  
 {% endif %}  
 </ul>  
 </div>  
 {% endif %}  
 {{ block.super }}  
{% endblock %}

* + - File storage\_edit.html

{% extends 'generic/object\_edit.html' %}  
{% load form\_helpers %}  
{% block form %}  
 {% render\_errors form %}  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Device</h5>  
 </div>  
 {% render\_field form.name %}  
 {% render\_field form.device\_role %}  
 {% render\_field form.tags %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Hardware</h5>  
 </div>  
 {% render\_field form.manufacturer %}  
 {% render\_field form.device\_type %}  
 {% render\_field form.airflow %}  
 {% render\_field form.serial %}  
 {% render\_field form.asset\_tag %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Location</h5>  
 </div>  
 {% render\_field form.region %}  
 {% render\_field form.site\_group %}  
 {% render\_field form.site %}  
 {% render\_field form.location %}  
 {% render\_field form.rack %}  
  
 {% if object.device\_type.is\_child\_device and object.parent\_bay %}  
 <div class="row mb-3">  
 <label class="col-sm-3 col-form-label">Parent Device</label>  
 <div class="col">  
 <input class="form-control" value="{{ object.parent\_bay.device }}" disabled />  
 </div>  
 </div>  
 <div class="row mb-3">  
 <label class="col-sm-3 col-form-label">Parent Bay</label>  
 <div class="col">  
 <div class="input-group">  
 <input class="form-control" value="{{ object.parent\_bay.name }}" disabled />  
 <a href="{% url 'dcim:devicebay\_depopulate' pk=object.parent\_bay.pk %}" title="Regenerate Slug" class="btn btn-danger d-inline-flex align-items-center">  
 <i class="mdi mdi-close-thick"></i>&nbsp;Remove  
 </a>  
 </div>  
 </div>  
 </div>  
 {% else %}  
 {% render\_field form.face %}  
 {% render\_field form.position %}  
 {% endif %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Management</h5>  
 </div>  
 {% render\_field form.status %}  
 {% render\_field form.platform %}  
 {% if object.pk %}  
 {% render\_field form.primary\_ip4 %}  
 {% render\_field form.primary\_ip6 %}  
 {% endif %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Virtualization</h5>  
 </div>  
 {% render\_field form.cluster\_group %}  
 {% render\_field form.cluster %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Tenancy</h5>  
 </div>  
 {% render\_field form.tenant\_group %}  
 {% render\_field form.tenant %}  
 </div>  
  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Virtual Chassis</h5>  
 </div>  
 {% render\_field form.virtual\_chassis %}  
 {% render\_field form.vc\_position %}  
 {% render\_field form.vc\_priority %}  
 </div>  
  
 {% if form.custom\_fields %}  
 <div class="field-group my-5">  
 <div class="row mb-2">  
 <h5 class="offset-sm-3">Custom Fields</h5>  
 </div>  
 {% render\_custom\_fields form %}  
 </div>  
 {% endif %}  
  
 <div class="field-group my-5">  
 <h5 class="text-center">Local Config Context Data</h5>  
 {% render\_field form.local\_context\_data %}  
 </div>  
  
 <div class="field-group mb-5">  
 <h5 class="text-center">Comments</h5>  
 {% render\_field form.comments %}  
 </div>  
  
{% endblock %}

* 1. **Kết quả**
     + Ta chạy lệnh dưới đây để khởi chạy server cho netbox mới tạo thêm Storage

$ python3 manage.py runserver 0.0.0.0:8001 –insecure

* + - Kết quả sau khi chạy lệnh trên:

Performing system checks...

System check identified no issues (0 silenced).

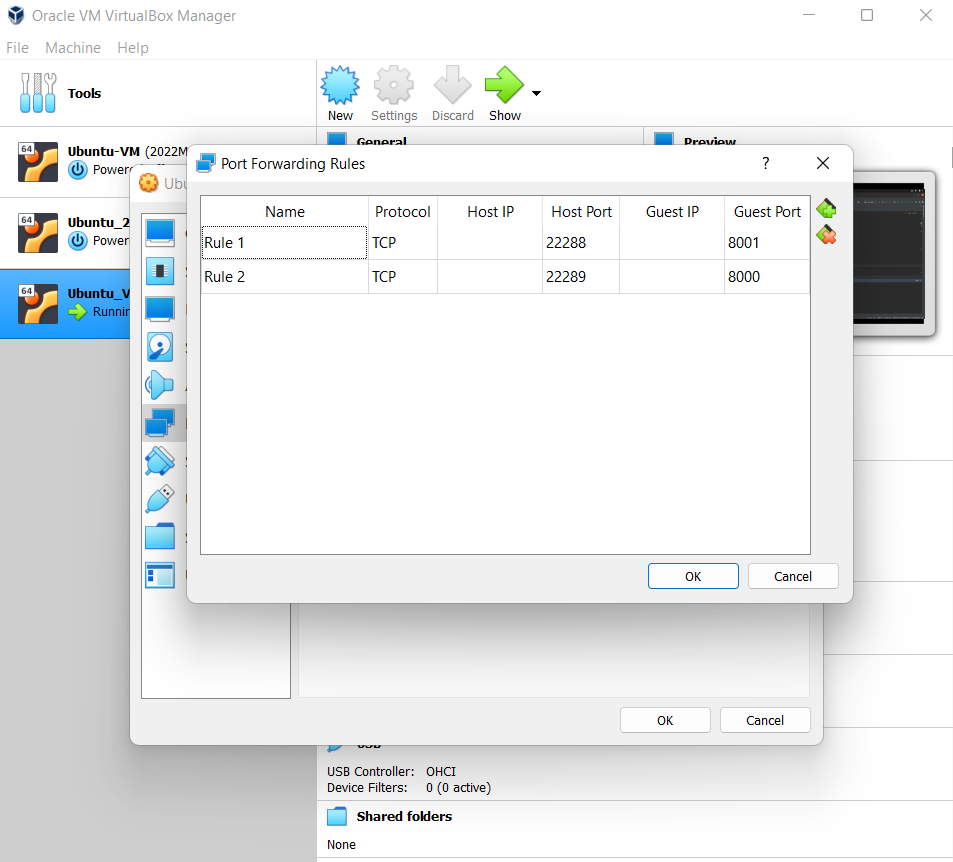
September 18, 2022 - 14:35:28

Django version 4.0.6, using settings 'netbox.settings'

Starting development server at http://0.0.0.0:8001/

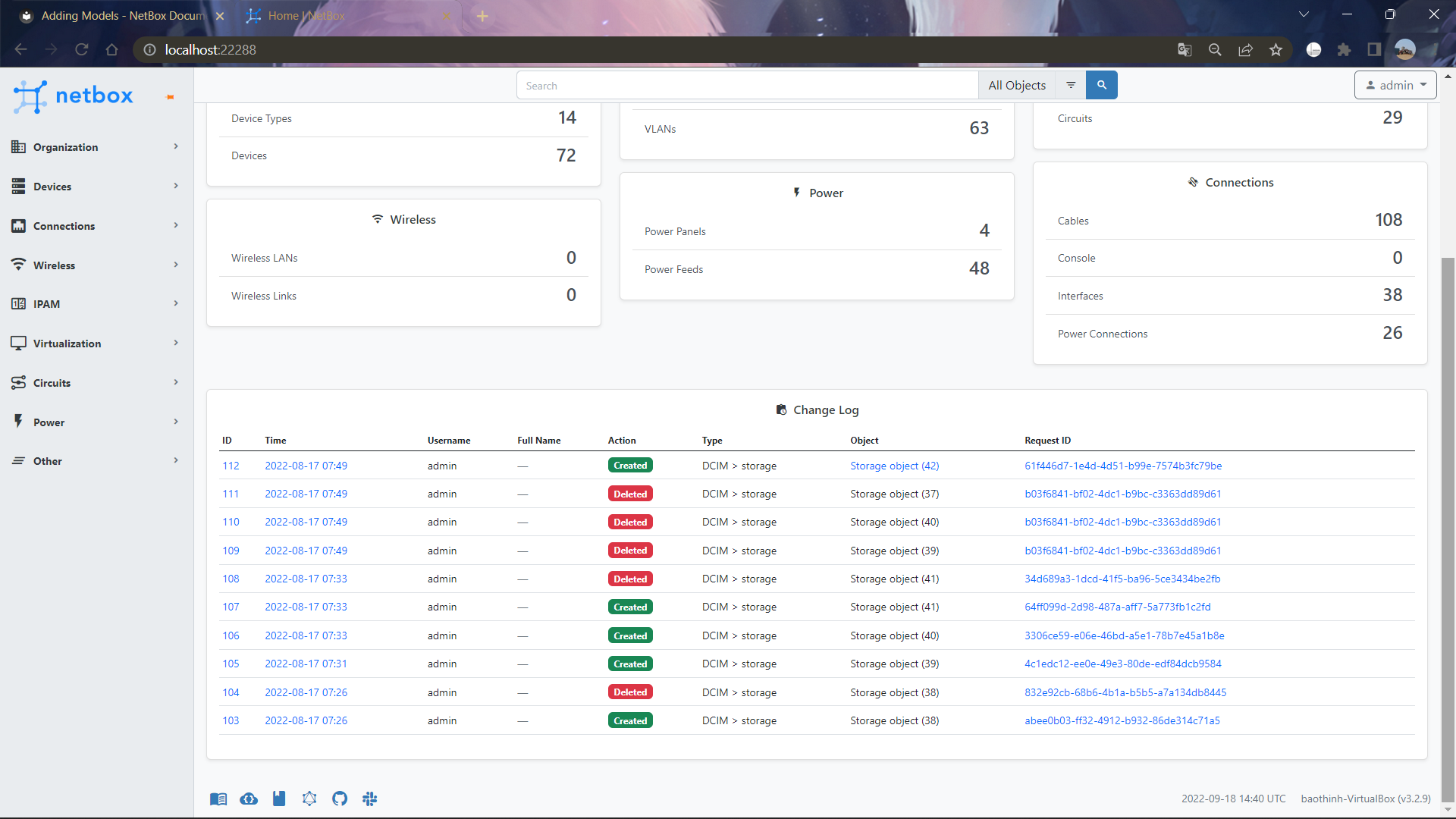
Quit the server with CONTROL-C.

* + - Ở đây ta chạy ở địa chỉ http://0.0.0.0:8001/ trên máy ảo nhưng ta sửa lại ở địa chỉ 22888 để truy cập trên máy chính (Window)

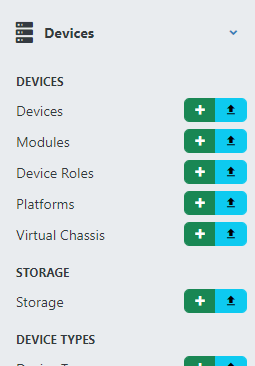


*Hình ảnh 12: Thay đổi port bằng VirtualBox*

* + - Giao diện sau khi sửa code và đã có một vài thay đổi

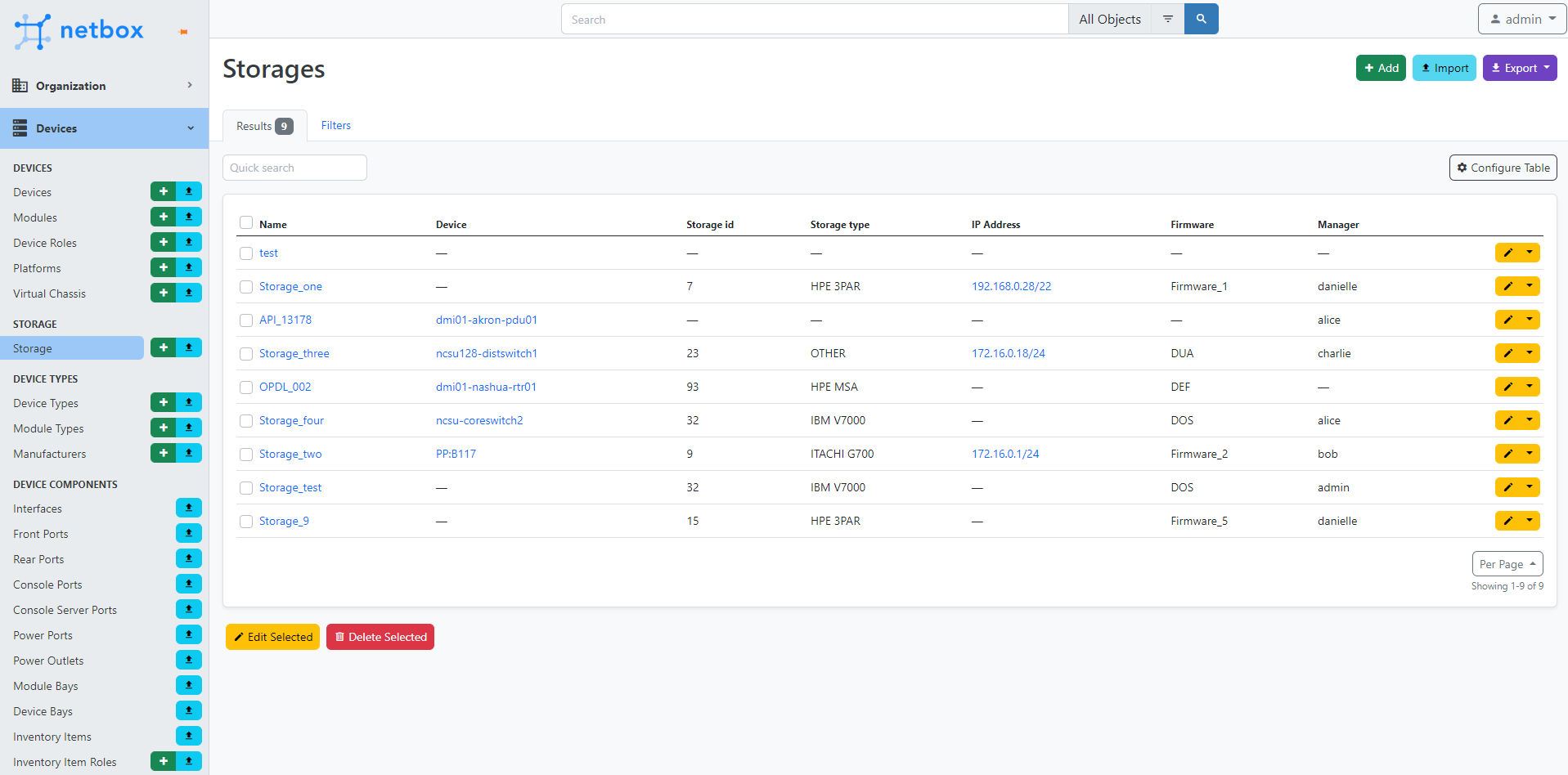


*Hình ảnh 13: Giao diện NetBox sau khi có sửa đổi*

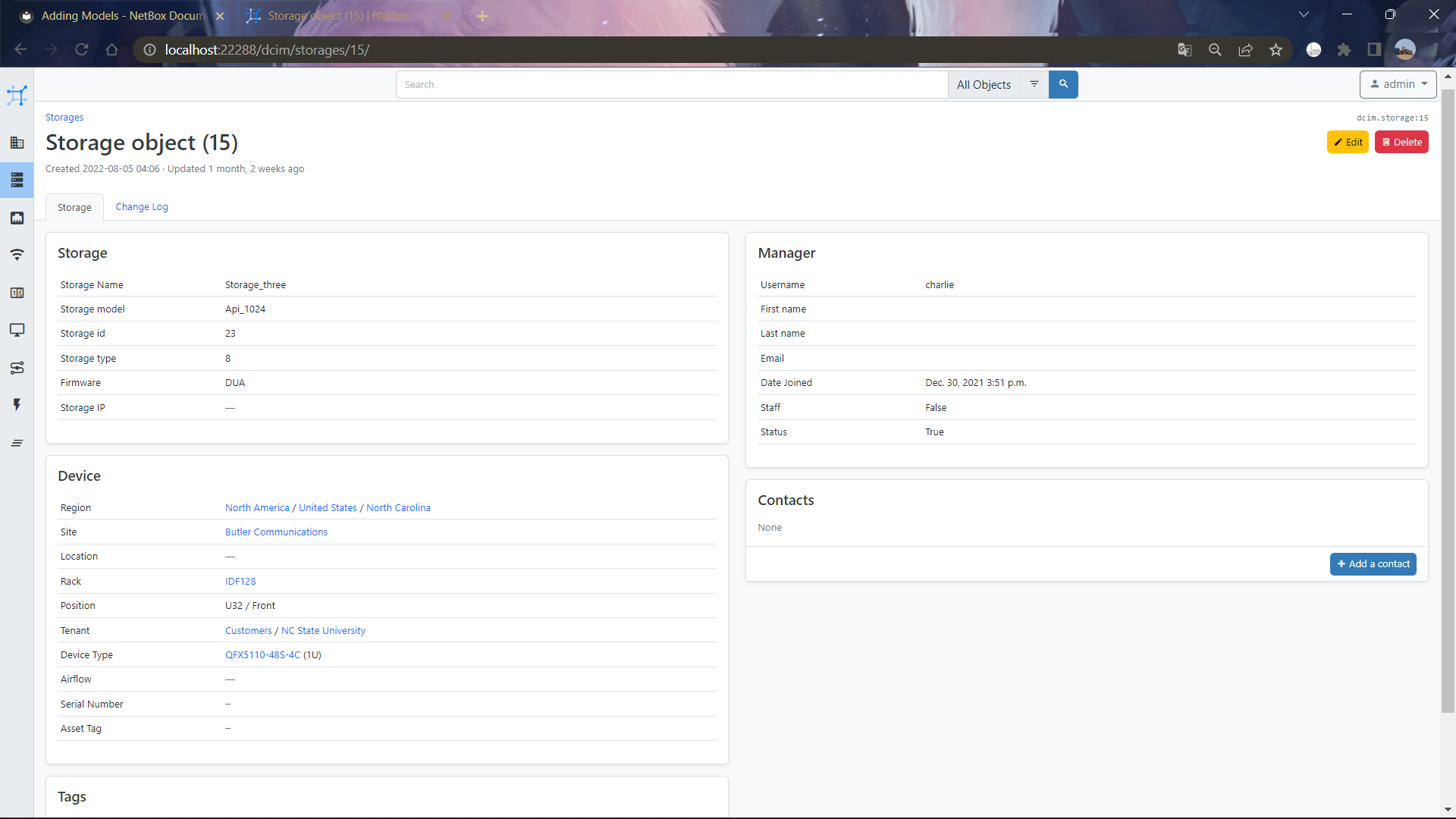


*Hình ảnh 14: Mục Storage riêng đã được tạo*

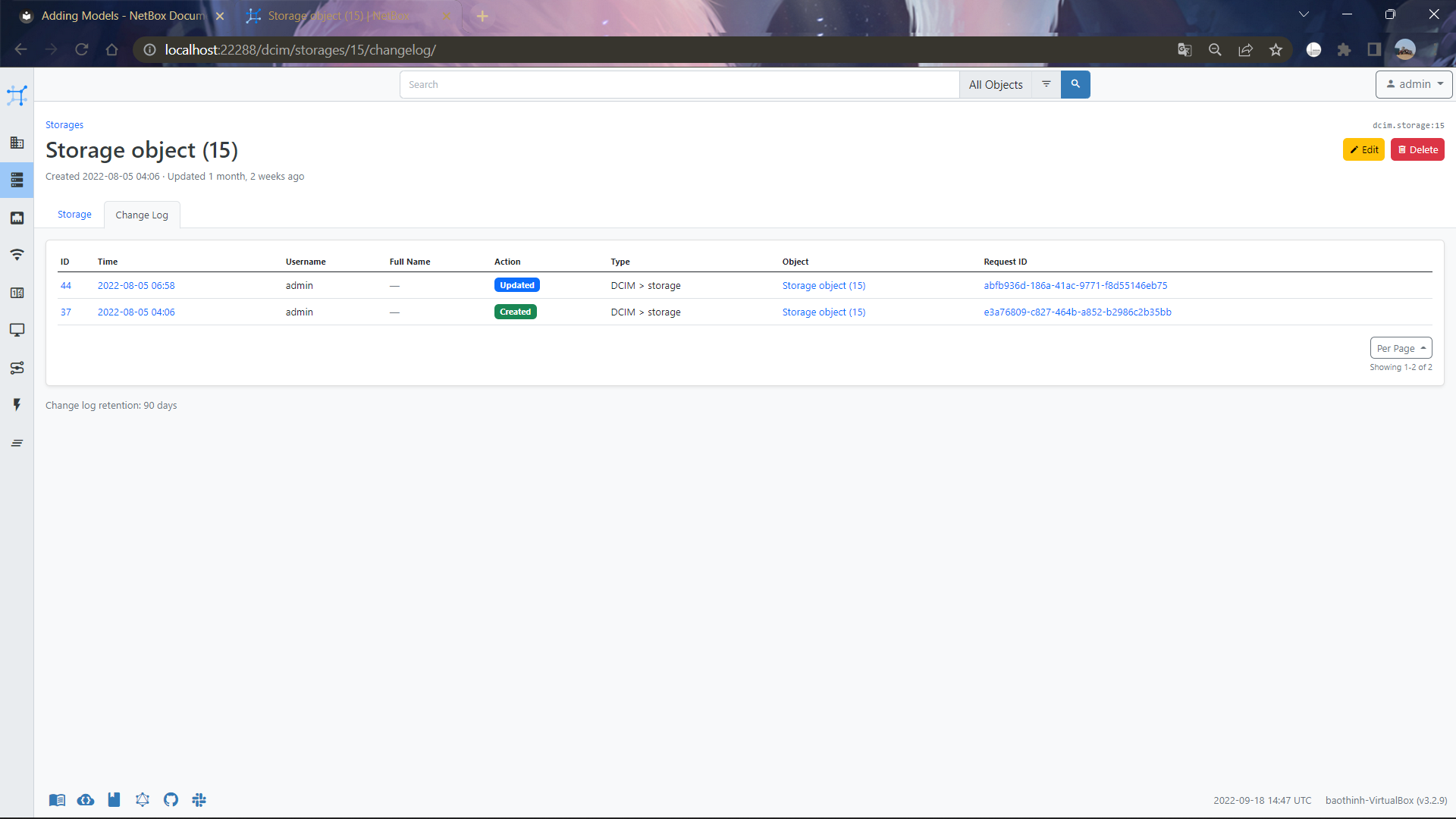
* + - Giao diện danh sách các Storage (Storage đã được tạo từ trước)
    - Các mục màu xanh là các mục đã được gán url để dẫn đến đường link trong NetBox ví dụ ta bấm vào mục IP Address của storage ta sẽ vào thông tin của IP Address đấy trong mục khác của NetBox



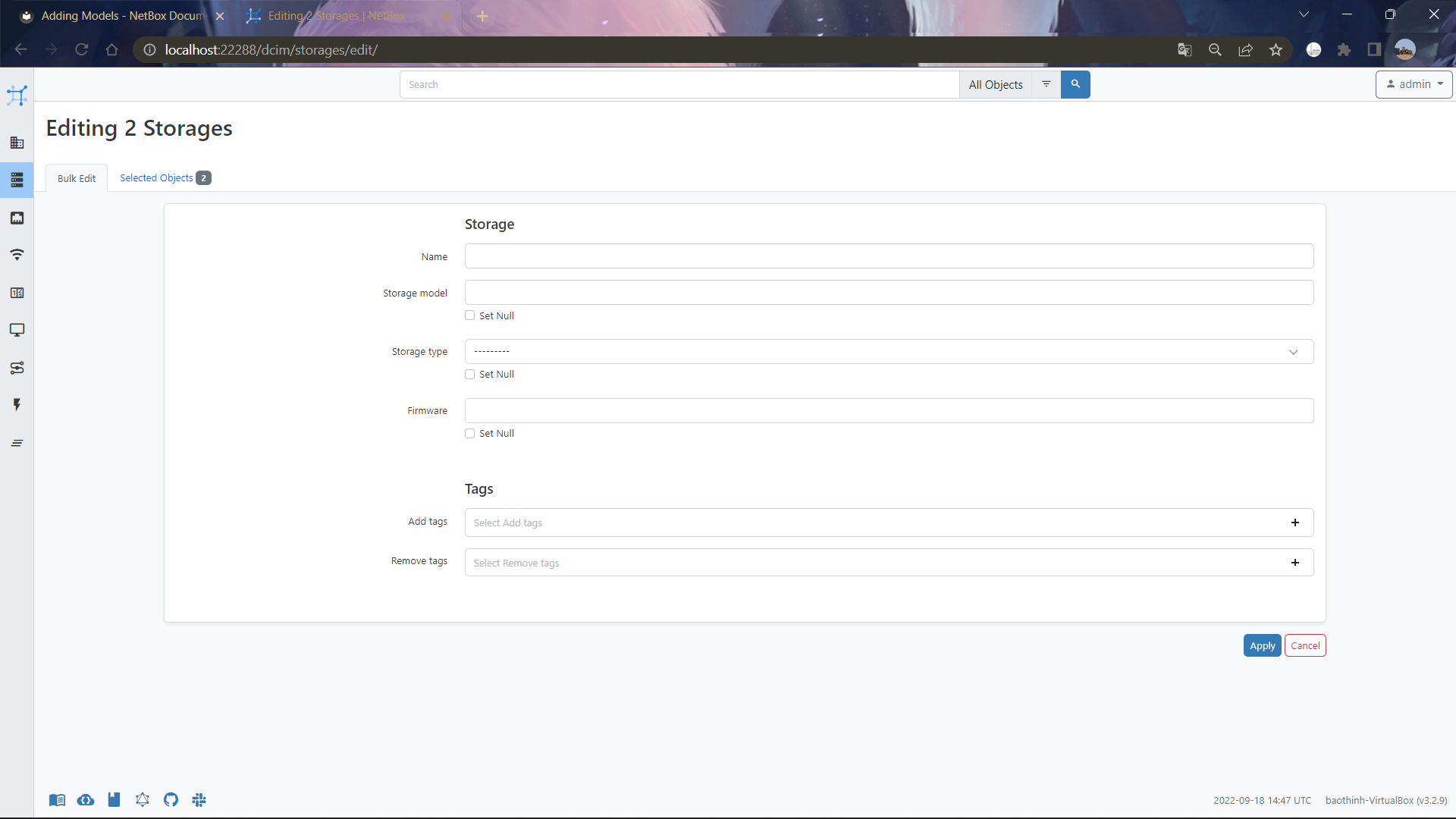
*Hình ảnh 15: Giao diện danh sách các Storage*



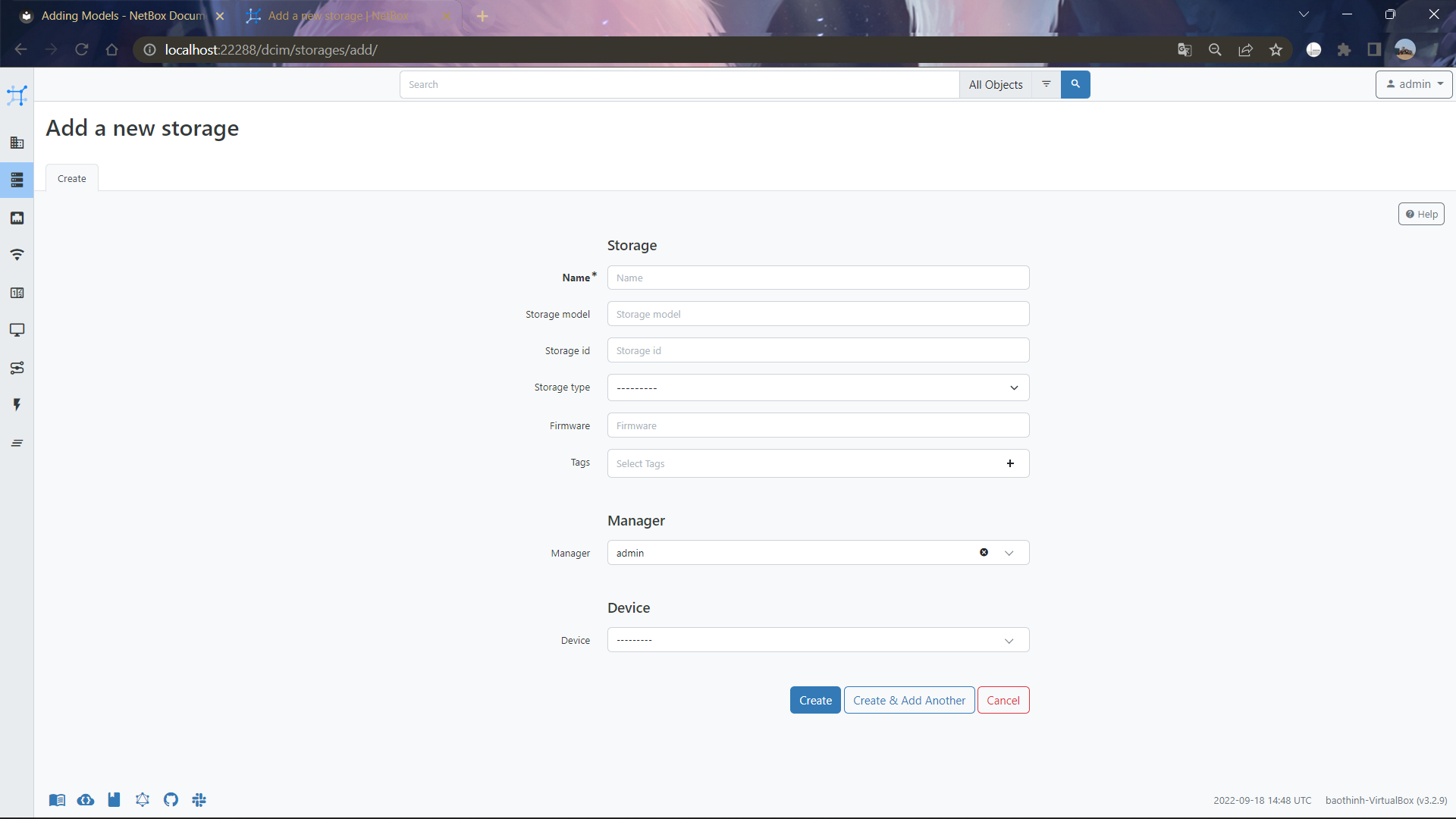
*Hình ảnh 16: Thông tin của Storage 3*



*Hình ảnh 17: Các thay đổi của Storage 3*

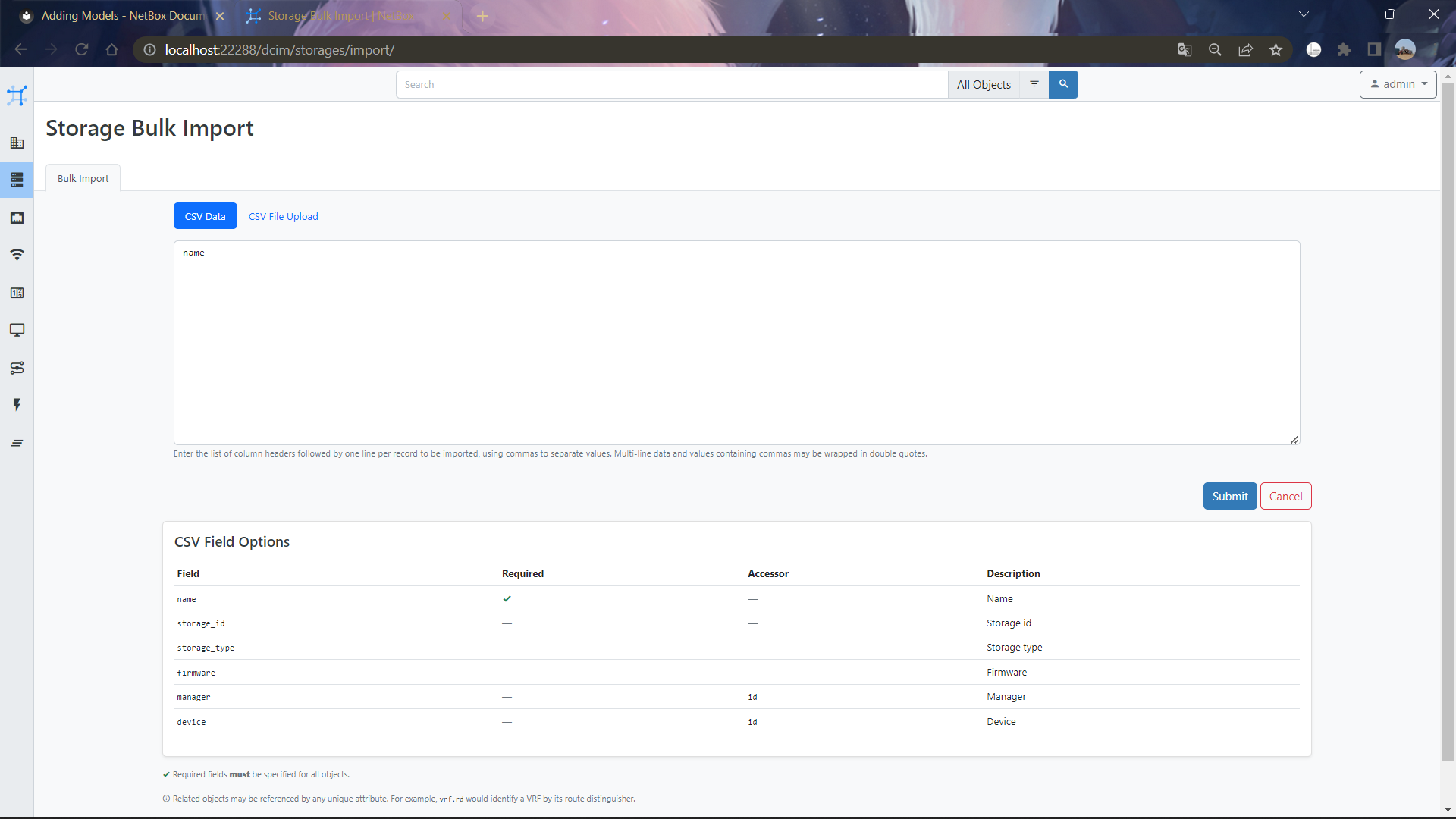


*Hình ảnh 18: Giao diện khi thay đổi nhiều Storage*

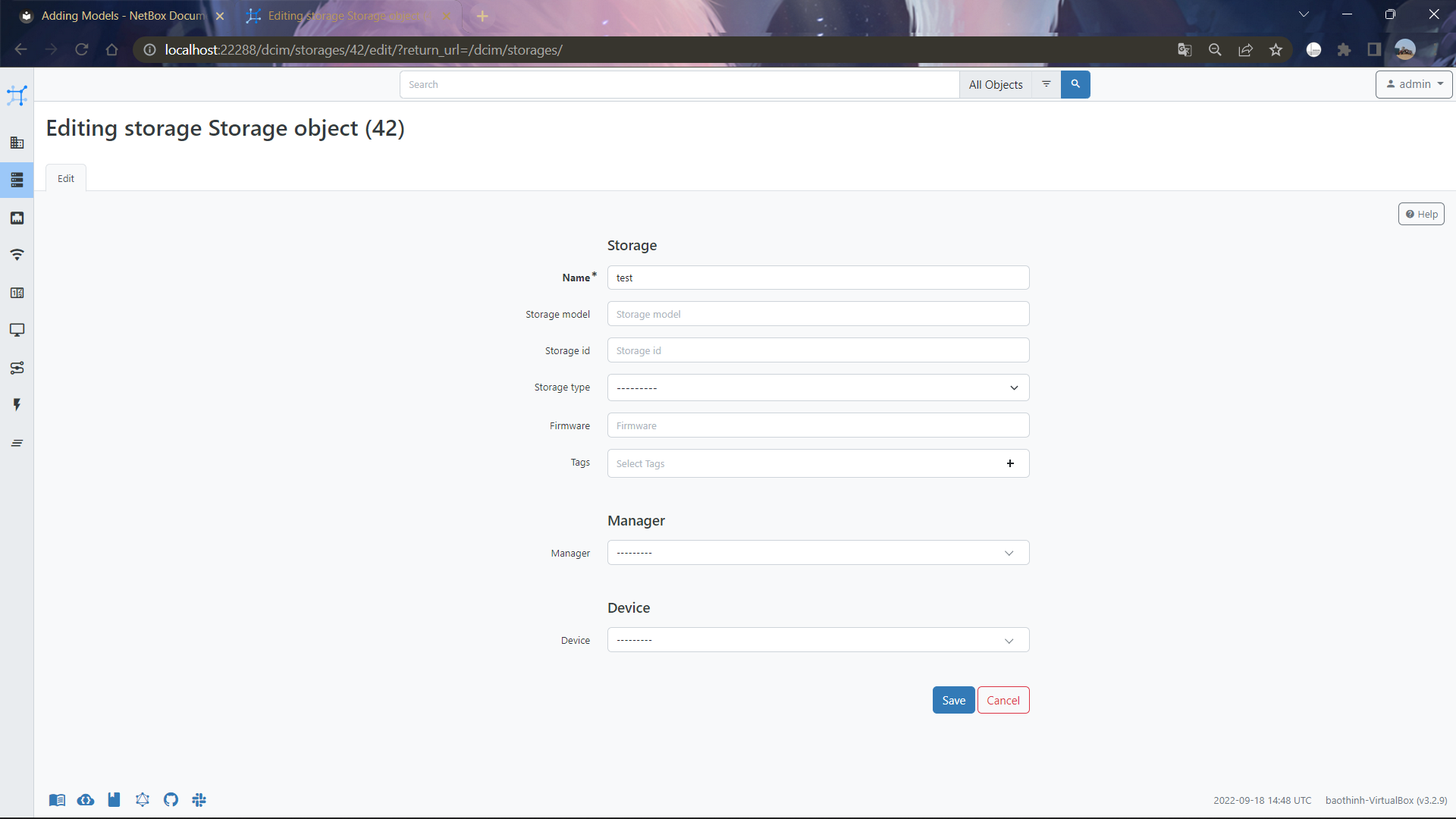


*Hình ảnh 19: Giao diện khi thêm một Storage*

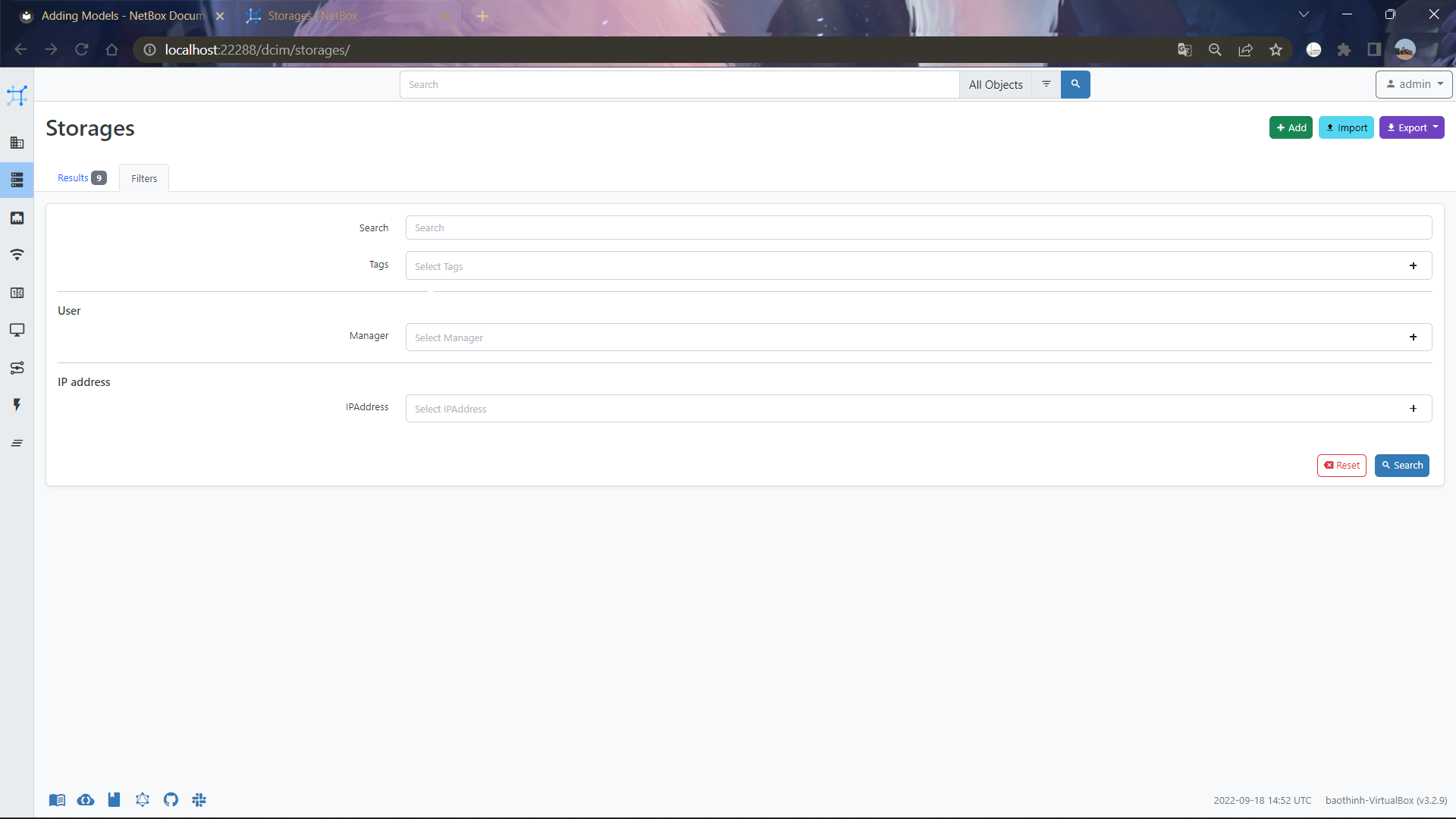
* + - Ở đây tính năng tự động nhận manager là tài khoản đang được sử dụng đã thành công
    - Các mục dấu sao là mục bắt buộc nhập



*Hình ảnh 20: Giao diện Import*



*Hình ảnh 21: Giao diện thay đổi một Storage*



*Hình ảnh 22: Giao diện Filter trong Storage*



*Hình ảnh 23: Đường dẫn API của Devices chứa Storages*

* + - Ta tạo file để thử các tính năng của API
    - API\_PUT

import requests  
import json  
  
url = "http://localhost:8001/api/dcim/storage/26/"  
  
payload = json.dumps({  
 "name": "Storage\_12",  
 "url": "http://localhost:8001/api/dcim/storage/26/",  
 "manager": {  
 "id": 5,  
 "username": "danielle"  
 },  
 "device": None,  
 "storage\_model": "Model\_1",  
 "storage\_id": "16",  
 "storage\_type": "5",  
 "firmware": "Firmware\_9",  
 "tags": []  
})  
headers = {  
 'Authorization': 'Token 3d9505626b04b4f56c5ce09406529ed48a43c624',  
 'Content-Type': 'application/json'  
}  
  
response = requests.request("PUT", url, headers=headers, data=payload)  
  
print(response.text)

* + - API\_GET

import requests  
  
url = "http://localhost:8001/api/dcim/storage/25/"  
  
payload={}  
headers = {  
 'Authorization': 'Token 3d9505626b04b4f56c5ce09406529ed48a43c624'  
}  
  
response = requests.request("DELETE", url, headers=headers, data=payload)  
  
print(response.text)

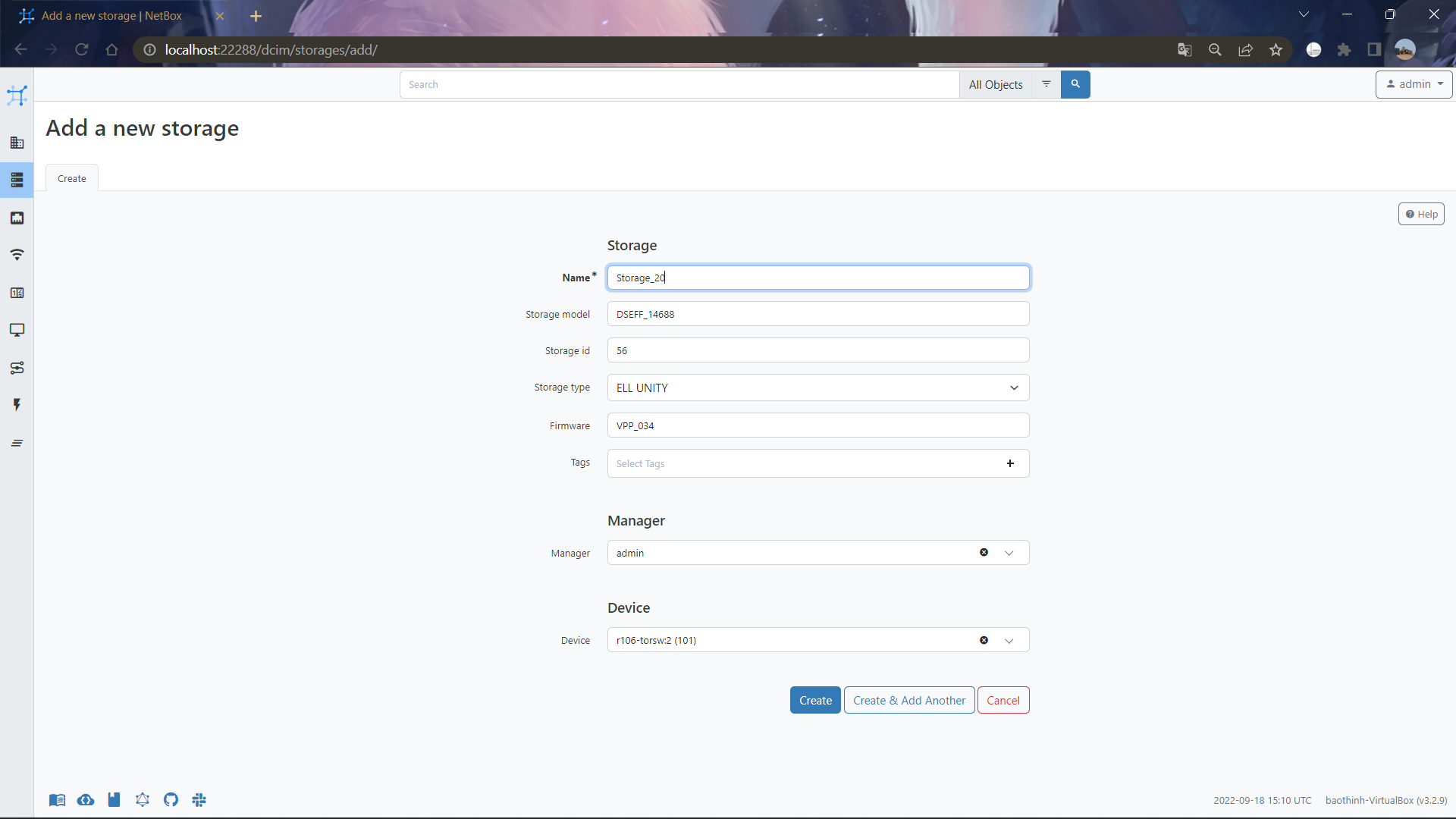
* + - API\_DELETE

import requests  
  
url = "http://localhost:8001/api/dcim/storage/25/"  
  
payload={}  
headers = {  
 'Authorization': 'Token 3d9505626b04b4f56c5ce09406529ed48a43c624'  
}  
  
response = requests.request("DELETE", url, headers=headers, data=payload)  
  
print(response.text)

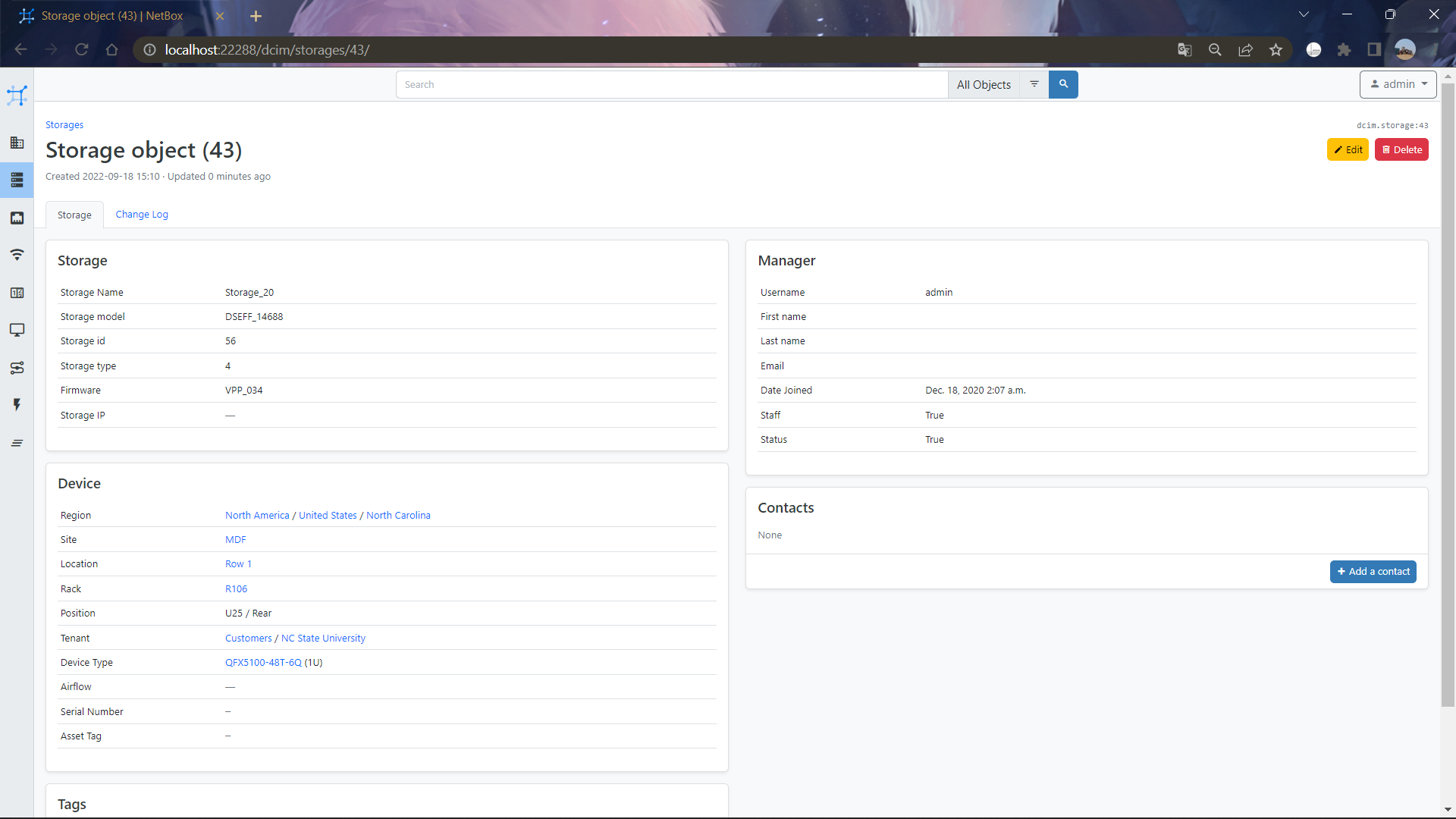
* + - API\_POST

import requests  
import json  
  
url = "http://localhost:8001/api/dcim/storage/"  
  
payload = json.dumps([  
 {  
 "name": "Storage\_9",  
 "url": "http://localhost:8001/api/dcim/storage/15/",  
 "manager": {  
 "id": 5,  
 "username": "danielle"  
 },  
 "device": None,  
 "storage\_model": "Model\_1",  
 "storage\_id": "15",  
 "storage\_type": "6",  
 "firmware": "Firmware\_5",  
 "tags": []  
 }  
])  
headers = {  
 'Authorization': 'Token 3d9505626b04b4f56c5ce09406529ed48a43c624',  
 'Content-Type': 'application/json'  
}  
  
response = requests.request("POST", url, headers=headers, data=payload)  
  
print(response.text)

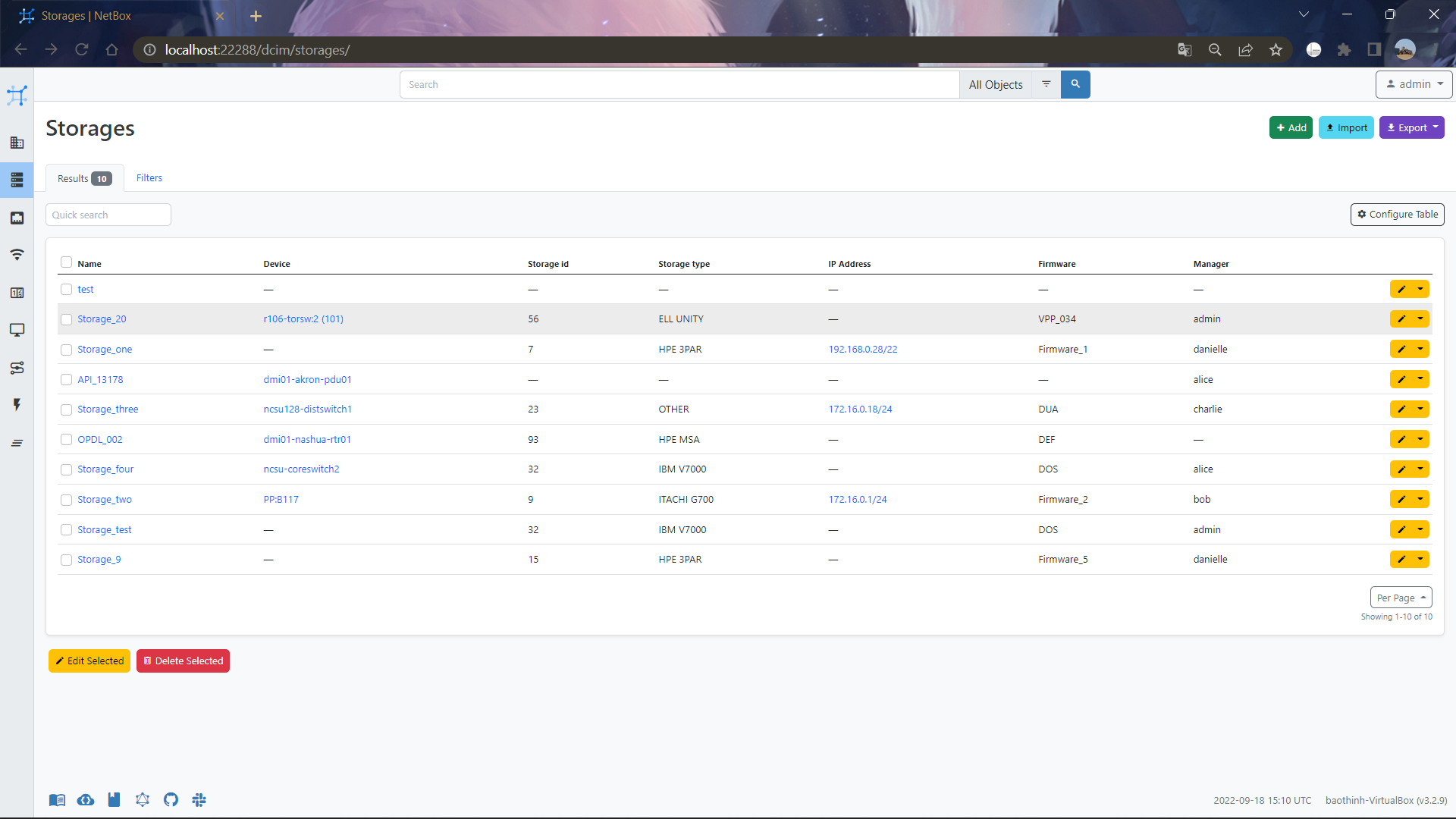
* + - Tồn tại Storage\_9 trên ảnh danh sách các storage là minh chứng cho việc khởi chạy các tính năng của API thành công
    - Ta thử một ví dụ để kiểm tra các tính năng của Storage:



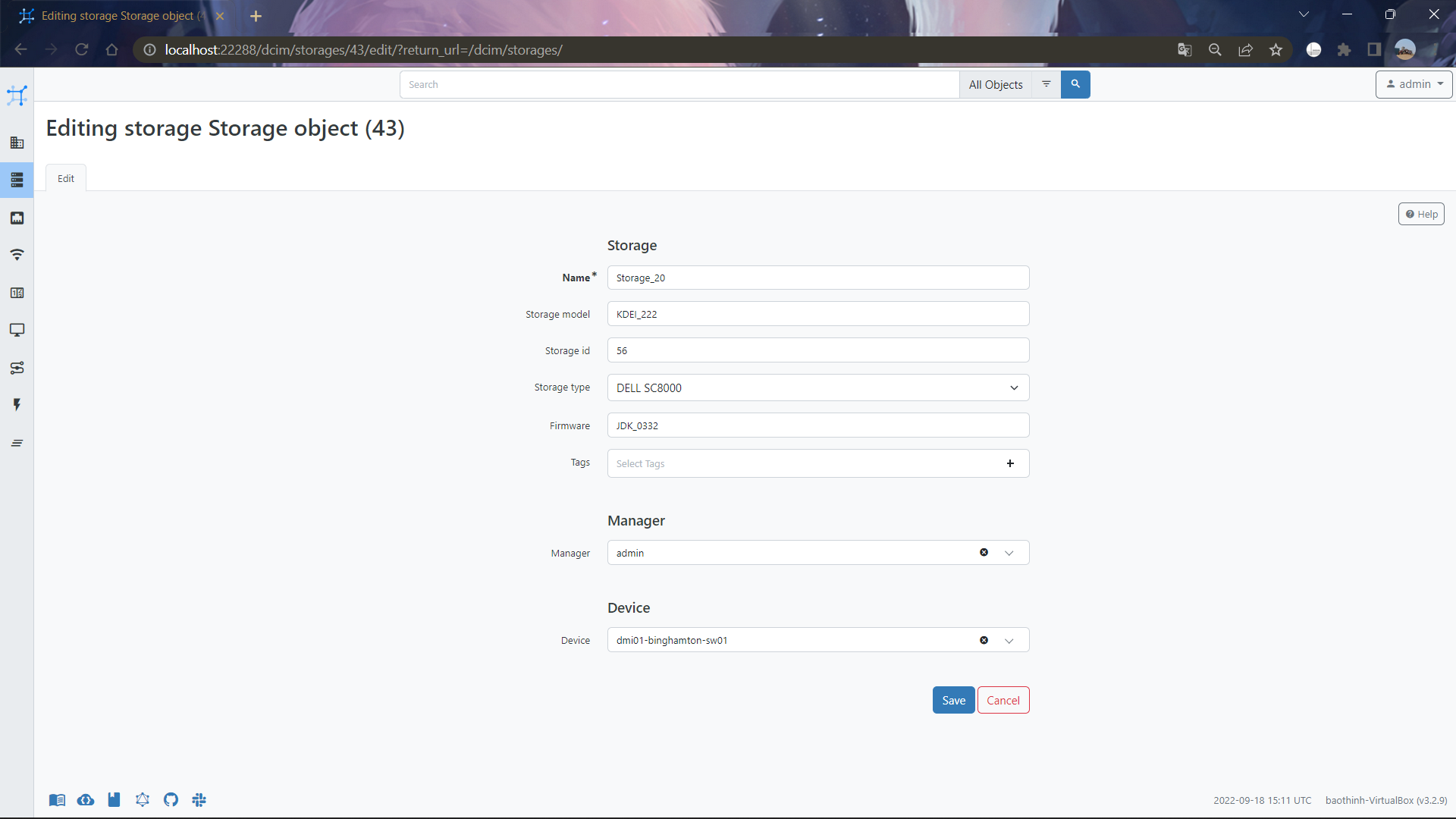
*Hình ảnh 24: Tạo một storage 20*



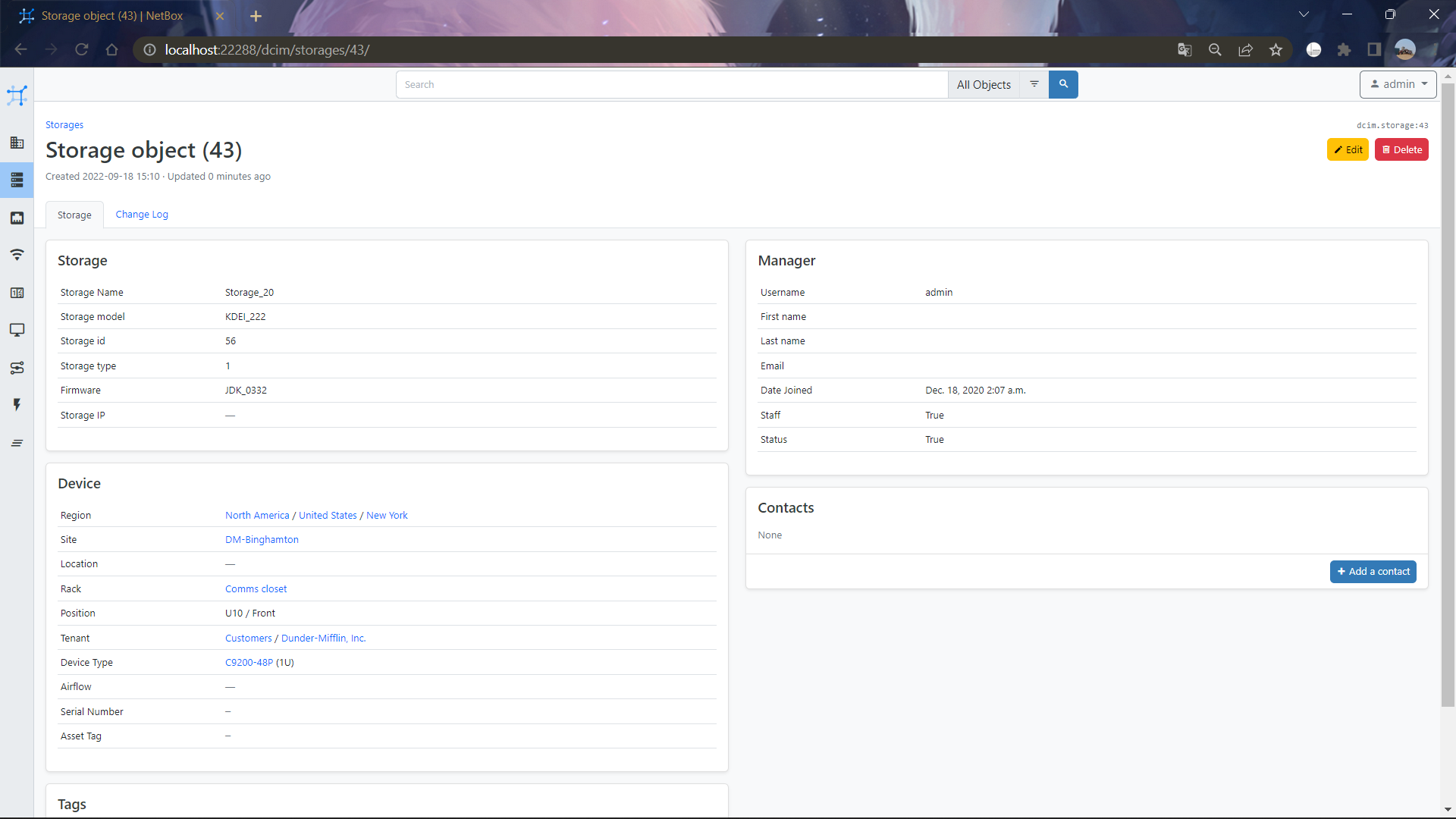
*Hình ảnh 25: Thông tin Storage sau khi tạo*



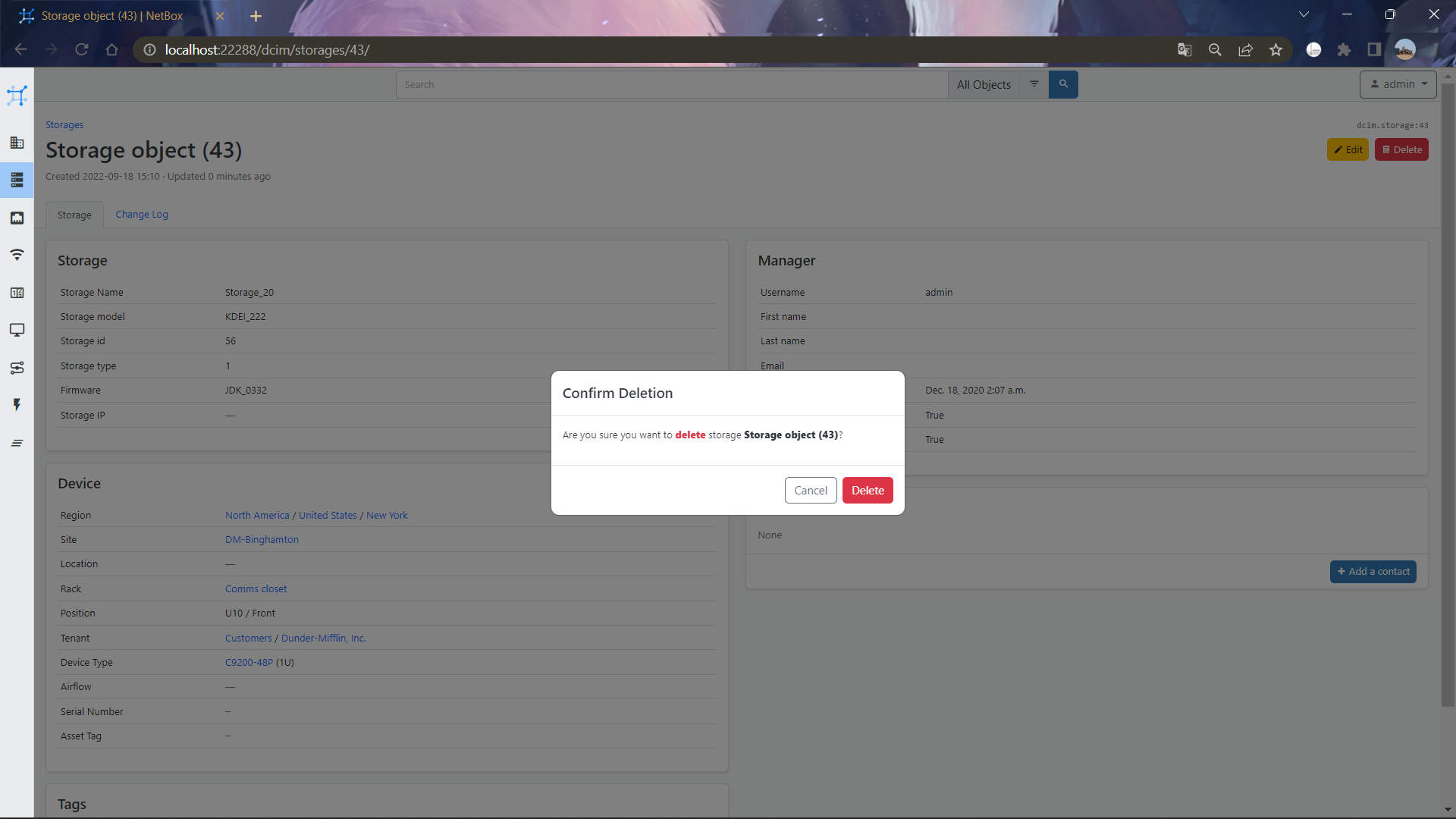
*Hình ảnh 26: Storage 20 được hiển thị trên danh sách*



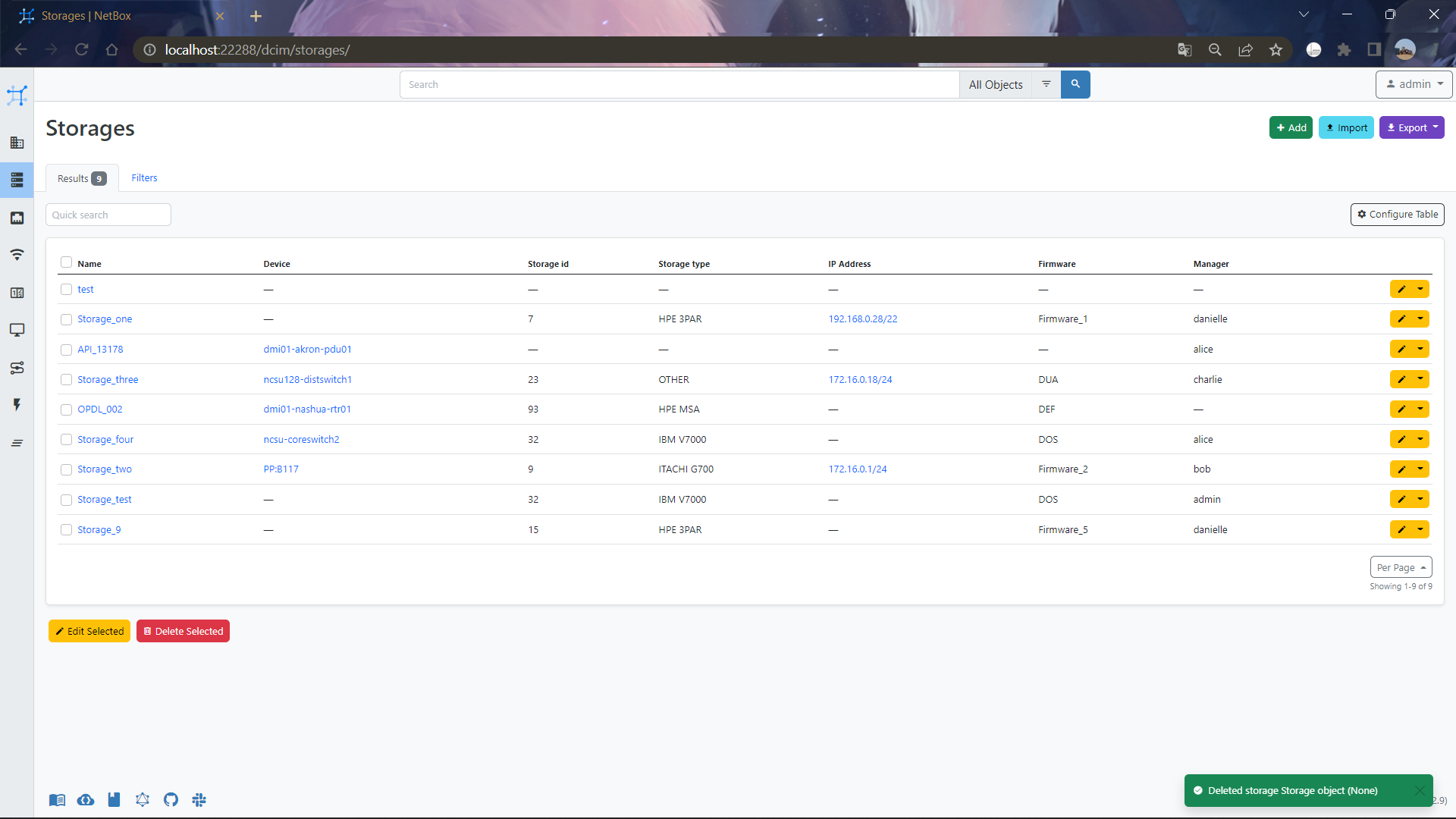
*Hình ảnh 27: Thử chỉnh sửa Storage 20*



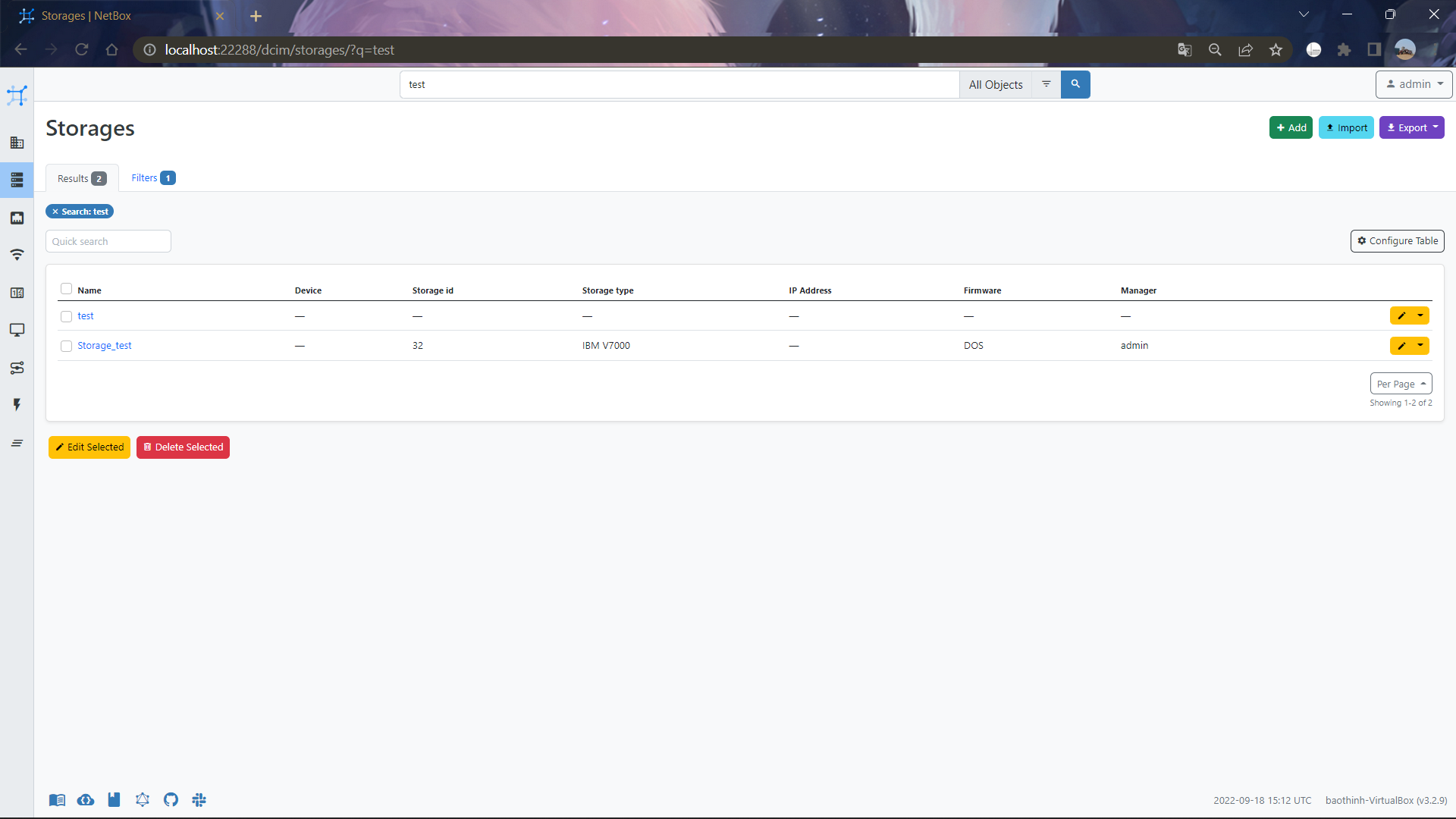
*Hình ảnh 28: Thông tin sau chỉnh sửa*



*Hình ảnh 29: Thử xóa Storage 20*



*Hình ảnh 30: Kết quả thành công sau khi xóa*



*Hình ảnh 31: Thử tính năng lọc*

1. **Debug**

* Hiện tại các lỗi em đã gặp phải gồm có:
  + Chưa thêm api cho netbox
  + Chưa thêm file html cho netbox
  + Không tìm được api của manager
  + Thêm Model chưa save
  + Định dạng sai kiểu trong quá trình viết html
  + Gặp khó khăn trong việc thêm một số hàm định nghĩa
  + Chưa add file vào danh mục tất cả của file
  + Thiếu Import
  + Định dạng sai loại hàm
  + Không tìm được đường dẫn trên máy tính tạo máy ảo (Window)
  + Lỗi chưa kill port đang chạy
* Các lỗi trên trong quá trình làm việc em đã sửa được nhưng không kịp chụp lại nên không có hình ảnh cụ thể về việc Debug

1. **Kết quả đạt được**

* Khả năng tự tìm hiểu, tìm tài liệu được phát triển.
* Khả năng viết cũng như sửa lỗi lập trình được nâng cao.
* Hiểu thêm về các kiến thức mới, những phần mềm mới, các kỹ thuật mới trong công nghệ cũng như lập trình.
* Việc đi làm ở công ty giúp em hiểu them được tính nghiêm túc và kỉ luật trong công việc cũng như ở công ty. Đồng thời em cũng học hỏi được rất nhiều từ các anh trong công ty về khả năng hòa đồng làm việc nhóm, trao đổi thông tin.
* Qua đề tài thì em cũng hiểu hơn về việc lập trình backend cũng như việc xử lý thành thạo các debug trong công việc lập trình hơn, đồng thời em có thế hiểu thêm về cách sử dụng github – một trong những website đang phát triển cho dân lập trình viên cũng như học được nhiều khái niệm mới mẻ trong thời đại công nghệ thông tin phát triển.
* Nhờ sự phát triển công nghệ qua thời gian mà con người nay càng dễ mô hình hóa, ảo hóa, dễ dàng xây dựng một hệ thống quản lý, sắp xếp cũng như ghi lại các thông tin hơn cũng như việc xuất hiện nhiều phần mềm kỹ thuật mới hơn nên việc thực tập này là cơ hội rất tốt để em có nhiều sự phát triển to lớn về sau.
* Sản phẩm của đợt thực tập được up lên trang github cá nhân https://github.com/Baothinh20

# **Tài liệu tham khảo**

1. “Thuật ngữ Cloud computing”: <https://blog.kdata.vn/cloud-computing-la-gi-cac-mo-hinh-trien-khai-cloud-computing-2701/?gclid=Cj0KCQjw9ZGYBhCEARIsAEUXITVtcj6-HVdfP7JOzB0wK3iYD7KNWaJ9rFs-msNddH7SpT6FbRl5QBQaAgf2EALw_wcB>
2. “Thuật ngữ framework”:
   * <https://tharong.com/python-framework-la-gi>
   * <https://itguru.vn/blog/10-python-framework-hang-dau-de-phat-trien-web/>
   * <https://topdev.vn/blog/framework-la-gi/>
   * <https://t3h.edu.vn/tin-tuc/top-framework-python-hoan-hao-danh-cho-lap-trinh-vien>.
3. “Thuật ngữ Django”: <https://www.djangoproject.com/>
4. “Cách cài đặt Pycharm”:
   * <https://t3h.edu.vn/tin-tuc/pycharm-la-gi-huong-dan-cai-dat-va-su-dung-pycharm>
   * <https://niithanoi.edu.vn/huong-dan-cai-dat-va-su-dung-pycharm-ide-trong-lap-trinh-python.html>
   * <https://www.jetbrains.com/help/pycharm/installation-guide.html#silent>
5. “Thuật ngữ Github”: <https://vi.wikipedia.org/wiki/GitHub>
6. “Thuật ngữ VirtualBox”: <https://en.wikipedia.org/wiki/VirtualBox>
7. “Thuật ngữ REST”: <https://en.wikipedia.org/wiki/Representational_state_transfer>
8. “Thuật ngữ Docker”: <https://topdev.vn/blog/docker-la-gi/>
9. “Cài đặt python”: <https://www.python.org/downloads/release/python-3105/>
10. “Thuật ngữ Python back end”: <https://vn.got-it.ai/blog/python-back-end-framework#:~:text=Python%20Back%2Dend%20framework%20l%C3%A0,v%C3%A0%20c%C3%B4ng%20d%E1%BB%A5ng%20kh%C3%A1c%20nhau>.
11. “Cài đặt Django”: <https://docs.djangoproject.com/en/4.0/intro/install/>.
12. “Cách viết 1 django app ”: <https://docs.djangoproject.com/en/4.0/intro/tutorial01/#writing-your-first-django-app-part-1> to part 7.
13. <https://docs.djangoproject.com/en/4.0/intro/reusable-apps/>.
14. “Resource code của NetBox trên github”: <https://github.com/netbox-community/netbox>.
15. “Hướng dẫn chi tiết về NetBox”<https://docs.netbox.dev/en/stable/>
16. “NetBox cài đặt”: <https://computingforgeeks.com/install-and-configure-netbox-ipam-dcim-tool-on-ubuntu/>.
17. “NetBox cài đặt”: <https://computingforgeeks.com/installing-postgresql-database-server-on-ubuntu/>
18. “NetBox”: <https://news.cloud365.vn/tag/netbox/>
19. Extra:
    * <https://howkteam.vn/course/upload-file-trong-lap-trinh-website-voi-python/tao-mot-web-app-va-xu-ly-khi-nguoi-dung-yeu-cau-truy-cap-trong-python-django-1517>.
20. Demo Netbox:
    * <https://demo.netbox.dev/>.

# **NHẬN XÉT KẾT QUẢ THỰC TẬP**