

# UNIVERSITATEA TEHNICĂ “GH ASACHI” IAŞI FACULTATEA AUTOMATICĂ ŞI CALCULATOARE

**SPECIALIZAREA TEHNOLOGIA INFORMAŢIEI**

**DISCIPLINA PROIECTAREA BAZELOR DE DATE**

**Gestiunea unei reţele**

# Coordonator,

**Cătălin Mironeanu**

**Student,**

**Vrînceanu George**

**Iaşi, 2021**

**Titlu proiect : Gestiunea reţelei**

Analiza, proiectarea şi implementarea unei baze de date și a aplicației aferente care să realizeze conectarea & gestiunea la o reţea de internet ( în proiectarea acestei teme s-a folosit un singur switch și un singur router ).

# Descrierea cerinţelor şi modul de organizare al proiectului

Gestiunea unei reţele eficiente de internet poate fi o provocare indiferent de modul în care aceasta a fost gândită şi implementată, din cauza potenţialului “rău” ce ar putea fi generat printr-o simplă greşeală. Din acest motiv, acest proiect ar trebui sa “simuleze” mai mult o reţea locala, complexitatea temei fiind astfel drastic micşorată. Astfel, în acest proiect am folosit tabele ce simulează următoarele dispozitive hardware:

* dispozitive, în care se introduce tipul de dispozitiv cu care te-ai conectat la reţea; dispozitive permise: computer, printer, laptop, smarthpone;
* wireless access point – 4 la număr - plasate a.î. un utilizator să se poată conecta la internet de pe laptop / smarthpone din interiorul incintei;
* router cu 4 interfeţe IP, cu rolul de a conecta aceasta reţea cu reţele externe, ex.: ISP;
* switch de 48 de porturi, în care se salvează toate dispozitivele conectate – puncte de acces wireless, calculatoare, servere etc.

Informaţiile de care ar fi nevoie pentru a crea această reţea sunt legate de:

* dispozitivul conectat: ne interesează să ştim ce fel de dispozitiv se conectează la reţea ( de exemplu, să ştim dacă utilizatorul s-a conectat de pe un telefon, sau dacă pur şi simplu s-a conectat cu un calculator la reţea );
* adresa fizica a dispozitivului, numita adesea MAC: această adresă fiind unică fiecărui dispozitiv în parte, munca de identificare a dispozitivului ne este semnificativ uşurată;
* adresa IP a dispozitivului: asemeni adresei MAC, ajută la identificarea dispozitivului;
* sistemul de operare a dispozitivului: în cazul in care se doreşte o anumită deosebire pe baza sistemului de operare folosit;
* wireless-ul la care este conectat user-ul: numai în cazul în care user-ul foloseşte un smarthpone / laptop - acest lucru este total transparent pentru user, SSID-ul fiind comun pentru întreaga reţea;
* port-ul la care este asociat dispozitivul in tabela switch: atât computerele, imprimantele cât și wireless-urile.

# Descrierea funcţională a aplicaţiei

Principalele lucruri ce ar trebui sa ni le ofere rețeaua ar trebui să fie reprezentate de:

* posibilitatea de a vedea ce dispozitiv s-a conectat la rețea;
* tipul dispozitivului – dacă s-a conectat la switch / wireless;

**Descrierea detaliată a entităţilor și a relaţiilor dintre tabele Tabelele** din această aplicaţie sunt:

* access\_point;
* access\_point\_bridge;
* devices;
* router;
* server;
* switch;

În proiectarea acestui proiect nu s-au folosit relații între toate tabelele, interacțiunea unora dintre acestea fiind realizată prin intermediul **triggerelor** pentru o apropiere mai buna faţă de o rețea adevărata. Astfel, foreign key-urile au fost păstrate doar pentru tabelul de router (o relație de **1:1 switch**) și pentru cel de server (o relație de **n:1**  **switch**). Mai departe, vom face abstracție de aceste două tabele, și vom vorbi doar despre tabele de devices, access\_point & access\_point\_bridge. Odată ce un dispozitiv nou este introdus la rețea prin intermediul tabelei devices, acesta este adăugat în tabela access\_point sau switch automat.

PS1: Prin folosirea foreign key-urilor acest lucru era cam dificil, deoarece tabela switch se dorește a fi tabela părinte, însă, tot odată ea să fie populată ultima.

PS2: Tabela devices ar fi trebuit să aibă ca și PK un foreign key de la switch, și un alt UK din tabela access\_point. Totodată cheia unică ce aparținea de acess\_point și care era referențiată în tabela devices, era de tipul FK din tabela access\_point\_bridge care la rândul ei era tot de acest tip, numai că aparținea de tabela switch.

În fiecare tabel sunt folosite constrângeri, prin intermediul triggerelor pentru a se verifica dacă tipul de date introdus este valid ( prin folosirea de regex).

- valori verificate prin regex: MAC, IP;

- valori verificate prin trigger asemănător constrângerii CHECK: tipul dispozitivului, sistemul de operare, tipul serverului.

De asemenea, având un switch de 48 de porturi, trebuie avut în considerare o tabela CAM ( cu adresele MAC a dispozitivelor, VLAN & port\_id ), astfel s-au împărțit cele 48 de porturi în următorul mod:

- primele 40 pentru dispozitive ce se conectează direct la internet prin intermediul cablurilor – pentru calculatoare și imprimante;

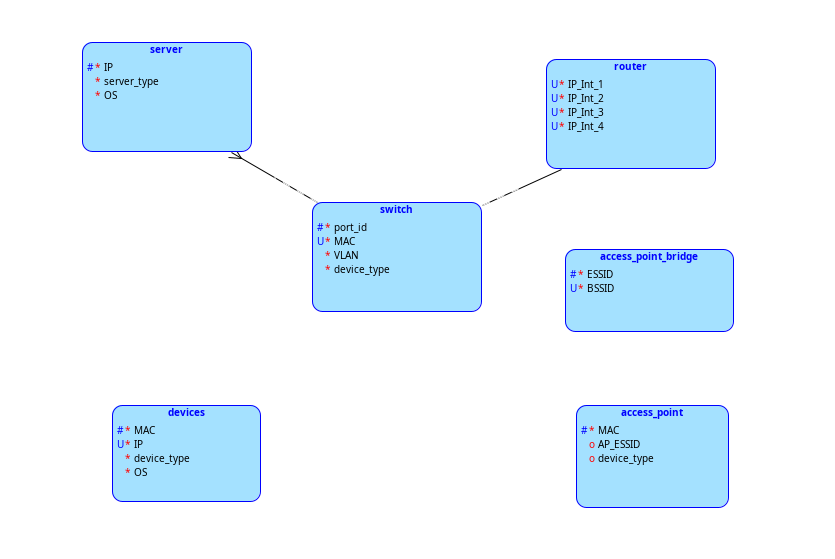
- următoarele 4 ( 41 – 44 ) pentru cele 4 wireless-uri;

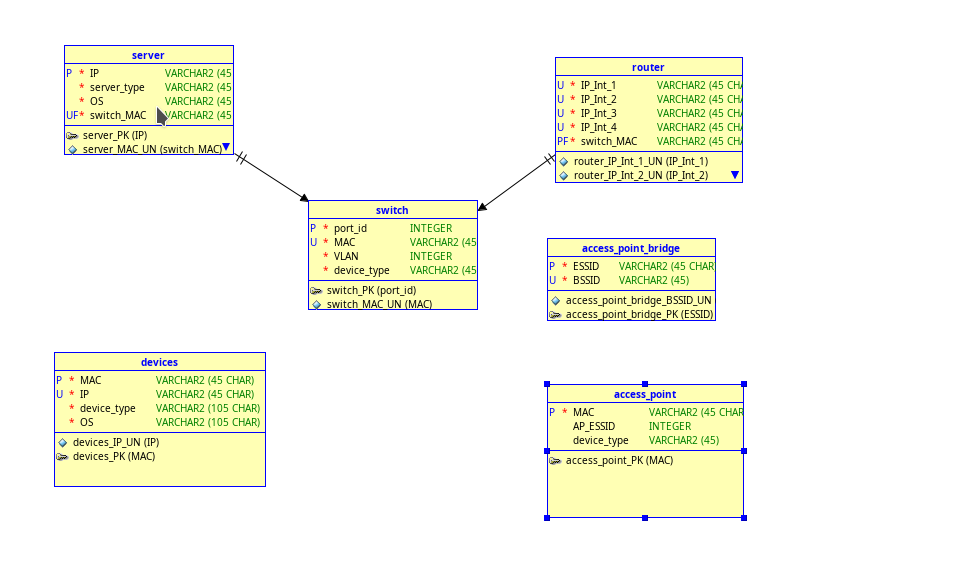
- porturile 45, 46, 47 pentru servere, rămânând ca portul 48 să fie pentru router;

Acestor porturi, VLAN-ul le este asignat automat prin intermediul triggerelor ( din 4 în 4, ex.: porturile de la 1 – 4 au VLAN 1, 5 – 8 au VLAN 2 ... 45 – 48 VLAN 12 ).

Cu toate că s-au folosit triggere pentru legătura dintre cele 4 tabele, relaţiile de 1:n,. ar exista – nu în schemă, ci într-un mod „transparent”. Odată un device nou introdus, înainte de a se adăuga și în switch se verifica tipul acestuia, de aici, împărţindu-se pe cazuri:

* dacă este pe cablu ( computer / imprimanta – dacă laptopul este conectat pe cablu se tratează ca și computer ) MAC-ul acestui dispozitiv este adăugat automat în tabela switch, astfel legătura dintre **switch** – **devices** ar fi **1** : **n**;
* dacă este pe wireless, traseul este unul mai lung, MAC-ul dispozitivului salvându-se în tabela **access\_point;** pe lângă MAC, tabela **access\_point** mai are o coloana device\_type unde se salvează tipul dispozitivului și **AP\_ESSID**; în aceasta coloana se salvează id-ul wireless-ului la care este conectat dispozitivul. Astfel, prin existența tabelului **access\_point\_bridge** în care se află cele 4 wireless-uri și MAC-urile asociate lor, utilizatorii au acces la internet, această tabela fiind conectată direct la switch. Legăturile dintre tabele ar fi:
  + **access\_point** – **device** ar fi **1** : **n**
  + **access\_point\_bridge – access\_point** ar fi **1 : n**
  + **switch – access\_point\_bridge** ar fi **1 : n**

Schema logică a tabelelor



Cu toate astea, am implementat și schema în cazul în care tabelele erau construite pe baza de foreign key-uri.

Schema relațională a tabelelor

# **Descrierea logicii stocate**

Toată logica prezentată mai sus a fost dezvoltată prin intermediul procedurilor, acestea fiind încapsulate în pachete. De asemenea, așa cum am precizat adineauri, avem triggere atât pentru verificarea datelor înainte de inserare / actualizare cât și validarea informației în toate tabelele odată ce aceasta a fost inserată. În cazul în care trigger-ele nu sunt valide, o excepție va fi apelată.

Tabela access\_point

Triggeri

* ACCESS\_POINT\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Verifică dacă datele din select / update sunt valide (1 <= essid <= 4), mac-ul și ip-ul respectă structura corectă a unui mac / ip.

Tabela access\_point\_bridge:

Pachete și proceduri

* ACCESS\_POINT\_BRIDGE\_PACK
  + ADD\_ACCESS\_POINT\_BRIDGE
  + UPD\_ACCESS\_POINT\_BRIDGE
  + DEL\_ACCESS\_POINT\_BRIDGE

Așa cum se observă și din numele procedurilor, acestea urmează paradigma CRUD.

Triggeri

* ACCESS\_POINT\_BRIDGE\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Verifică dacă datele din select / update sunt valide (1 <= essid <= 4), mac-ul și ip-ul respectă structura corectă a unui mac / ip.
* ACCESS\_POINT\_BRIDGE\_SELECT\_PORT\_ID **( After Insert )**
  + Introduce datele în switch cu un port\_id valid.
* ACCESS\_POINT\_BRIDGE\_UPDATE\_BSSID **( After Update )**
  + Odată actualizat MAC-ul dintr-o intrare a tabelei access\_point\_bridge, acest trigger are rolul de a modifica MAC-ul și din tabela switch.
* ACCESS\_POINT\_BRIDGE\_DELETE\_FROM\_SWITCH **( After Delete )**
  + Odată șterse datele din access\_point\_bridge, acest trigger are rolul de a șterge datele din switch.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tabela switch:

Pachete și proceduri

* SWITCH\_PACK
  + GET\_SWITCH\_DATA

Nu avem nevoie de metode precum Create, Update sau Delete deoarece toate se execută automat prin triggere.

Triggeri

* SWITCH\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Validează tipul datelor prin regex-uri și atribuie VLAN-uri în funcție de disponibilitatea porturilor și a tipului dispozitivului.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tabela router:

Pachete și proceduri

* ROUTER\_PACK
  + ADD\_ROUTER
  + DEL\_ROUTER

Triggeri

* ROUTER\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Verifică dacă datele din select / update sunt valide: mac-ul și ip-ul respectă structura corectă a unui mac / ip;
  + Introduce datele despre router în switch.
* ROUTER\_DELETE\_FROM\_SWITCH **( After Delete )**
  + Odată ștearsă o intrare din router, acest trigger are rolul de a șterge intrarea și din switch.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tabela server:

Pachete și proceduri

* SERVER\_PACK
  + ADD\_SERVER

Triggeri

* SERVER\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Verifică dacă datele din select / update sunt valide: tipul server-ului este valid, mac-ul și ip-ul respectă structura corectă a unui mac / ip.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tabela devices:

Pachete și proceduri

* DEVICE\_PACK
  + ADD\_ROUTER

Triggeri

* DEVICE\_VALIDATE\_ENTRY **( Before Insert or Update )**
  + Verifică dacă datele din select / update sunt valide: tipul dispozivitului este valid, mac-ul și ip-ul respectă structura corectă a unui mac / ip.
* MANAGE\_DEVICES **( After Insert** **or Update )**
  + Are scopul de a intrdouce dispozivitul în tabela corespunzătoare (access\_point / switch) și de selecta PORT\_ID-ul corespunzător.

Alt aspect de precizat ar fi faptul că, executarea a mai multor proceduri simultan sau a unor instrucțiuni clasice precum Select / Insert a fost realizată prin **tranzacții.**