Desarrolladores: Herman Pineda Rubio – Romel Saquicela Loja

Descripción

El contenido del documento presenta información enfocado a la duplicación de software diseñado para las personas que lo deseen implementar.

Manual técnico

Sistema de Control de un Laboratorio, Recursos y Manejo de Sesiones en PC

# Resumen ejecutivo

El software desarrollado y propuesto como solución para la administración de computadoras en un laboratorio plantea facilitar la administración a la persona encargada del laboratorio. El software recopila información del estado de las computadoras además de mostrar avisos en tiempo real sobre la desconexión de sus periféricos USB a través de una página web.

Entre los clientes del mercado objetivo resaltan quienes son jefes de laboratorios de facultades universitarias, dueños de negocios tipo ciber-café. Además, como mercado potencial tenemos a cualquier negocio que disponga de una red ordenadores que son accedidos y deben su estado deba ser monitoreado.

# Descripción del problema

En un laboratorio de computación se desea tener control remoto y constante monitoreo sobre las estaciones de trabajo, de tal manera que se detecten en ellas la extracción de dispositivos mediante alarmas mostradas en la pantalla de la PC administrador que estará siendo supervisada por el encargado del laboratorio. Además de la opción de apagar la y las PC’s de manera remota.

# Objetivos específicos

Mantener un control de la seguridad de un laboratorio tal que permita al usuario detectar alguna interrupción en los equipos mediante la activación de alertas.

Apagar de manera remota las computadoras de un laboratorio mediante el uso de SSH, desde la computadora del administrador.

Crear un front end con una página web donde se muestren las notificaciones, información de inventario y la interacción con las opciones de control de las computadoras.

# Solución propuesta

Dentro de la red ya establecida en el laboratorio, con un ambiente de trabajo en Linux, se establecen dos tipos de configuraciones; clientes usada por las personas que acceden a los ordenadores; y servidor, la cual es una única PC administrada por encargado del laboratorio.

Los ordenadores tipo Usuario, tendrán instalado; MongoDB para guardar información del PC, periféricos conectados y sobre las notificaciones de los periféricos removidos; SSH para establecer la conexión remota y que el administrador pueda controlarlas.

La Workstation del ordenador tendrá instalado del mismo modo MongoDB y SSH, además de tener ejecutándose scripts para establecer la conexión de las rutas, IP’s de las PC’s clientes, para llevar a cabo las funciones del lado del backend como apagado, descargas de archivos y consultas a la base de datos de MongoDB.

También se hizo una página web, cuyo front end es manipulado por el usuario administrador desde su estación de trabajo

# ¿Qué van a construir para resolver el problema?

Se construirá un front end para que exista una fácil interacción entre el monitoreo del estado, información de las PC’s, alarmas y el control por parte del administrador.

Existirá un back end, el cual consiste en una serie de scripts ejecutándose en las computadoras clientes y Workstation, útiles para recopilar datos y enviarlos a MOONGODB.

Se asume que la infraestructura de red ya esta implementada en el laboratorio a trabajar.

# Recursos de hardware y de software

El laboratorio ya esta implementado y también su infraestructura de red, pero entre los componentes de hardware usados están:

* Los ordenadores para los usuarios y la Workstation para el administrador
* Además de los periféricos como; teclado, ratón, monitor; del cuales se tendrán información si llegan a ser son removidos.

Como recurso de software se utilizará:

* MongoDB se instala en todos los ordenadores para el manejo de la base de datos con información de las PC clientes.
* También se instala SSH en todos los ordenadores para el control remoto de las PC’s usuario por parte del administrador.
* Java Script para realizar el front end, que es la pagina web que permite la fácil interacción del administrador con el sistema de monitoreo.
* Python es usado para realizar scripts de la obtención de información de los periféricos desconectados.

# Explicación paso a paso de la implementación del proyecto

Los ordenadores clientes tendrán instalado SSH y MongoDB, además de tener registrado en ellos 1 usuario extra llamado “off\_pc” el cual es utilizado para apagar la PC de manera remota. SSH se lo utiliza para tener acceso remoto desde la Workstation del administrador. MongoDB es la base de datos que se decidió utilizar debido a la estabilidad horizontal que ofrece este tipo de base de datos no SQL, los datos almacenados son información de estas PC como; IP, nombre de usuario, apagado o encendido, hostname, tiempo de inicio de sesión y si tiene acceso a internet; también se crea otra tabla que mantiene información sobre los periféricos conectados este pc; además existen otra tabla con información volátil con el mensaje de las alarmas mostrado en el front end. Los ordenadores clientes además tendrán ejecutándose scripts para; establecer la conexión a MongoDB y enviar y actualizar la información de la base de datos.

La Workstation de igual manera tendrá instalado SSH y MongoDB. Además, ejecutara scripts para obtener el direccionamiento de las PC conectadas a la red que serán controladas por SSH, y de este modo mediante líneas de código apagarlas de manera remoto. El Shutdown se lo realiza accediendo a un usuario administrador creado en los ordenadores clientes, “off\_pc”, desde el cual se manda la orden mediante un script que se debe apagar el ordenador y con ello terminar la sesión de todos los usuarios logeados en ese ordenador.

# Diagramas

## diagrama de circuito (topologia)

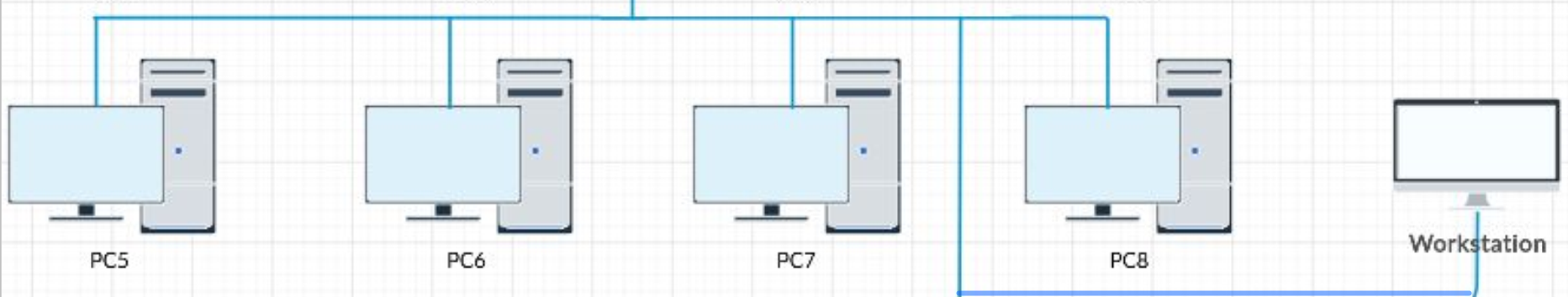


Ilustración 1: Topología del laboratorio.

## diagrama del modelo entidad-relación

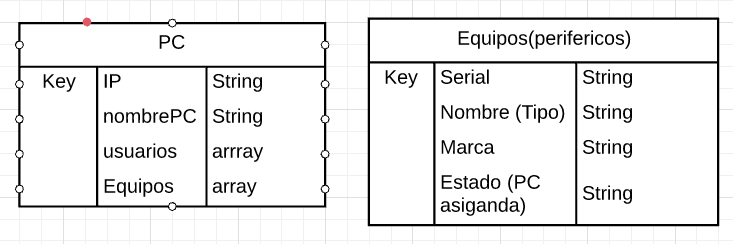
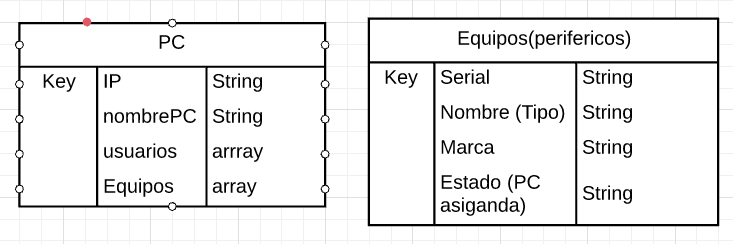


Ilustración 2: Tablas con la información de las PC “clientes” en MongoDB.

# Base de datos, código mongodb

La base de datos creada es mediante MongoDB que es una base de datos no SQL. Se eligió este tipo respecto a MySQL porque el guardado de información es más sencillo y además por la opción del escalamiento horizontal que propone MongoDB por sobre MySQL, ventaja si se llega llenar información usando campos diferentes en las tablas creadas.

## Descripción de los campos y tipo de datos

Existen 2 tablas principales y otra que temporal y almacena información sobre las alarmas.

### Tabla pc

Como campos están:

* “name” el cual describe el nombre asignado de manera practica al pc. El tipo de dato usado en este campo es de tipo String.
* “status” representa el estado actual del PC, encendido o apagado. El tipo de dato usado en este campo es de tipo String.
* “IP” es la dirección IP del computador. El tipo de dato usado en este campo es de tipo String.
* “hostname” es el nombre con el cual está registrado el PC. El tipo de dato usado en este campo es de tipo String.
* “start time” indica la hora a la cual se empezó a utilizar el ordenador. El tipo de dato usado en este campo es de tipo String.
* “internet” indica si el ordenador cuenta con acceso a internet. El tipo de dato usado en este campo es de tipo String.

### tabla perifericos

Entre los campos utilizados constan:

* “nombre” indica el nombre con el cual se describe al periférico. El tipo de dato usado en este campo es de tipo String.
* “marca” indica la marca asociada al periférico. El tipo de dato usado en este campo es de tipo String.
* “serial” es el número de serie que tiene asociado el periférico. El tipo de dato usado en este campo es de tipo String.
* “estado” indica si el periférico está asignado a un PC o no, si está asignado a un PC, aparece el nombre del PC. El tipo de dato usado en este campo es de tipo String.

### Tabla alarmas

Temporalmente será almacenada con la única finalidad de ser mostrada en el front end.

* “name” indica el nombre de la alarma. El tipo de dato usado en este campo es de tipo String.
* “description” es el texto con el que se mostrará la alarma. El tipo de dato usado en este campo es de tipo String.
* “type” indica el timpo de alarma, puede ser un WARNING (color amarillo) o una NOTIFICATION (color azul). El tipo de dato usado en este campo es de tipo String.

## código Base de datos (mongodb)

### log\_out.sh

#!/bin/sh

**unset** tecreset os architecture kernelrelease internalip externalip nameserver

# Check Internal IP

internalip**=$(hostname -I | awk '{print $1}')**

# Check System Uptime

end\_time**=$(who | awk '{print $4}')**

# Check User

userNow**=$(who | awk '{print $1}')**

#Insert document to DB

**printf** "use laboratorio\n" **>** **.**out.js

**printf** "db.wkwebs.update({'name':'PC-C'},{\$set: {status:\"Libre\",internet: \"--\", hostname: \"--\", ip: \"$internalip\",start\_time: \"$end\_time\"}})\n" **>>** **.**out.js

**printf** "db.alarmas.update({'name':'PC-C'},{\$set: {description: \"$userNow has closed on to PC-C\", type: \"info\"}})\n" **>>** **.**out.js

#Connection to MongoDB

mongo -u admin -p admin **--**authenticationDatabase admin 200**.**126**.**13**.**226 **<** **.**out.js

# Unset Variables

**unset** tecreset os architecture kernelrelease internalip externalip nameserver

**exit** 0

Comentario: Este script se ejecuta cada vez que un usuario cierre sesión en la PC, está colocado en todos los PC´s clients. Principalmente se conforma de tres partes: - recopilación de datos del equipo - Inserción de datos a la base de datos – y conexión a mongodb**.**

### log\_in.sh

#! /bin/bash

**unset** tecreset os architecture kernelrelease internalip externalip nameserver

# Check if connected to Internet or not

#ping -c 1 google.com &> /dev/null && Status="Connected" || Status="Disconnected"

Status**=**"Connected"

# Check User

userNow**=$(who | awk '{print $1}')**

# Check Internal IP

internalip**=$(hostname -I | awk '{print $1}')**

# Check System Uptime

dateuptime**=$(who | awk '{print $3}')**

start\_time**=$(who | awk '{print $4}')**

#Insert document to DB

**echo** -e "use laboratorio\ndb.workstations.insert({Internet: \"$Status\", hostname: \"$userNow\", ip: \"$internalip\",date: \"$dateuptime\", start\_time: \"$start\_time\"})\n" **>** **.**in.js

**echo** -e "db.wkwebs.update({'name':'PC-C'},{\$set: {status:\"Ocupado\",internet: \"$Status\", hostname: \"$userNow\", ip: \"$internalip\",start\_time: \"$start\_time\"}})\n" **>>** **.**in.js

**echo** -e "db.alarmas.update({'name':'PC-C'},{\$set: {description: \"$userNow has logged on to PC-C\", type: \"info\"}})\n" **>>** **.**in.js

#Connection to MongoDB

mongo -u admin -p admin **--**authenticationDatabase admin 200**.**126**.**13**.**226 **<** **.**in.js

# Unset Variables

**unset** tecreset os architecture kernelrelease internalip externalip nameserver

**echo** "SI SE EJECUTO at $(date)" **>>** inlogs

**Comentario:** Este script se ejecuta cada vez que un usuario inicie sesión en la PC, está colocado en todos los PC´s clients. Principalmente se conforma de tres partes: - recopilación de datos del equipo - Inserción de datos a la base de datos – y conexión a MongoDB.

### alarmas.js

module.exports = function() {

var db = require('../libs/connection-db')();

var mongoose = require('mongoose');

var alarma = mongoose.Schema({

name: String,

description: String,

type: String

})

return mongoose.model('alarmas',alarma);

}

Comentario: Esquema para la conexión a la base de datos a la colección donde se guarda las alarmas en tiempo real.

### workstation.js

module.exports = function() {

var db = require('../libs/connection-db')();

var mongoose = require('mongoose');

var wk = mongoose.Schema({

name: String,

status: String,

ip: String,

hostname: String,

start\_time: String,

internet: String

})

return mongoose.model('wkwebs',wk);

}

Comentario: Esquema para la conexión a la base de datos a la colección donde se guarda la información de las PC’s clients en tiempo real.

# código fuente

### index.js

const express **=** require**(**'express'**;**

const router **=** express.Router**();**

const model **=** require**(**'../model/workstation'**)();**

const alarmas **=** require**(**'../model/alarmas'**)();**

router.get**(**'/'**,(**req**,**res**)** **=>** **{**

model.find**({},(**err**,** data**)=>{**

console.log**(**data**)**

**if(**err**)** throw err**;**

res.render**(**'index'**,{**

title**:** 'ASO Session Monitor'**,**

description**:** 'Sistema de control de laboratorio, recursos y manejo de sesiones en PC'**,**

workstations**:** data

**});**

**});**

**});**

**//**Mongo Export

var exec **=** require**(**'child\_process'**).**exec**;**

var child**;**

**function** mongoexport**(**out\_string**)**

**{**

child **=** exec**(**out\_string**,** function **(**error**,** stdout**,** stderr**)**

**{**

console.log**(**'stdout: ' **+** stdout**);**

console.log**(**'stderr: ' **+** stderr**);**

**if** **(**error **!==** null**)** **{**

console.log**(**'exec error: ' **+** error**);**

**}**

**})**

**}**

var wk\_string **=** "mongoexport --db=laboratorio --collection=workstations --type=csv --fields=Internet,hostname,ip,date,start\_time --out workstations.csv"

var hw\_string **=** "mongoexport --db=laboratorio --collection=hardware --type=csv --fields=nombre,marca,serial,Estado --out hardware.csv"

var hosts\_string **=**"mongoexport --db=laboratorio --collection=hosts --type=csv --fields=nombre,usuarios,IP,Equipos --out hosts.csv"

router.get**(**'/historial'**,(**req**,**res**)=>{**

mongoexport**(**wk\_string**);**

**})**

router.get**(**'/Inventario'**,(**req**,**res**)=>{**

mongoexport**(**hw\_string**);**

**})**

router.get**(**'/hosts'**,(**req**,**res**)=>{**

mongoexport**(**hosts\_string**);**

**})**

**//**Downloads

router.get**(**'/downloadHistorial/'**,** **(**req**,** res**)** **=>** **{**

res.download**(**'./workstations.csv'**);**

**})**

router.get**(**'/downloadInventario/'**,** **(**req**,** res**)** **=>** **{**

res.download**(**'./hardware.csv'**);**

**})**

router.get**(**'/downloadHWHosts/'**,** **(**req**,** res**)** **=>** **{**

res.download**(**'./hosts.csv'**);**

**})**

**//**Shutdown

**function** off**(ip){**

console.log**(**'off@'**+**ip.trim**()+**'hola'**)**

const **{** spawn **}** **=** require**(**'child\_process'**);**

const ls **=** spawn**(**'ssh'**,** **[**'off@'**+**ip.trim**()]);**

ls.stdout.on**(**'data'**,** **(**data**)** **=>** **{**

console.log**(`stdout: ${data}`);**

**});**

**}**

router.get**(**'/shutdown/:id'**,** function **(**req**,** res**)** **{**

**let** id **=** req.params.id**;**

model.findById**(**id**,(**err**,**pc**)=>{**

**if** **(**err**)** throw err**;**

console.log**(**pc.ip**)**

off**(**pc.ip**)**

pc.status**=**"Apagado"

pc.hostname**=**" "**;**

pc.internet**=**" "**;**

pc.start\_time**=**" "**;**

pc.save**().**then**(()** **=>** res.redirect**(**'/'**))**

**})**

**});**

**//**PowerOFF All

router.get**(**'/offall'**,**function**(**req**,**res**){**

console.log**(**'Apagando Equipos...'**)**

model.find**({},(**err**,**pc**)=>{**

**if** **(**err**)** throw err**;**

**for(**var i **=** 1**;** i **<** pc.length**;**i++**){**

console.log**(**pc**[**i**].**ip**)**

off**(**pc**[**i**].**ip**)**

pc**[**i**].**status**=**"Apagado"

pc**[**i**].**hostname**=**" "**;**

pc**[**i**].**internet**=**" "**;**

pc**[**i**].**start\_time**=**" "**;**

pc**[**i**].**save**()**

**}**

res.redirect**(**'/'**)**

**})**

**})**

**//**Alarmas

router.get**(**'/alarmas'**,(**req**,**res**)=>{**

alarmas.find**({},(**err**,**data**)=>{**

res.render**(**'alarmas'**,{**

title**:** 'Alarmas'**,**

alarma**:** data

**});**

**})**

**})**

router.get**(**'/deletAlarma/:id'**,** function **(**req**,** res**)** **{**

**let** id **=** req.params.id**;**

alarmas.findById**(**id**,(**err**,**pc**)=>{**

**if** **(**err**)** throw err**;**

console.log**(**pc.ip**)**

pc.description**=**0

pc.save**().**then**(()** **=>** res.redirect**(**'/'**))**

**})**

**});**

module.exports **=** router**;**

Comentario: Archivo que nodeJS usa para la conexión de las rutas, en este se llevan a cabo las funciones del lado del backend como apagado, descargas de archivos y consultas a la base de datos de MongoDB.

### mongo.py

#!/usr/bin/env python

import pymongo

import time

import os

from pymongo import MongoClient

client **=** MongoClient**()**

client **=** MongoClient**(**'200.126.13.226'**,** 27017**)**

db **=** client.laboratorio

alarmas**=**db.alarmas

file**=**"/tmp/devices.log"

def logsExist**(file):**

try**:**

logs **=** open**(file)**

logs.close**()**

**return** 1

except**:**

**return** 0

**while(**1**):**

**if(**logsExist**(file)):**

logs **=** open**(file)**

evento**=**logs.readline**()**

alarmas.update\_one**({**"name"**:** "PC-C"**},{**"$set"**:** **{**"description"**:** evento,"type"**:** "warning" **}})**

time.sleep**(**150**)** #Wait 2.5mins

alarmas.update\_one**({**"name"**:** "PC-C"**},{**"$set"**:** **{**"description"**:** 0**}})**

os.remove**(file)**

logs.close**()**

time.sleep**(**0**.**5**)**

Comentario: App de python que se ejecuta cada vez que el equipo inicia, colocado en cada PC cliente, este se conecta a mongodb y notifica cuando la regla: 80-test.rules se cumple.

### 80-test.rules

SUBSYSTEM**==**"usb"**,** ACTION**==**"remove"**,** ENV**{**DEVTYPE**}==**"usb\_device"**,** RUN+**=**"/bin/device\_removed.sh"

**Comentario:** Regla que usa el SO Linux para notificar que un porifierico ha sido removido.

### device\_removed.sh

#!/bin/bash

**echo** "Device removed in PC-C at $(date)" **>** **/**tmp**/**devices.log

**Comentario:** Script que se ejecuta cuando se cumple la regla del archivo: 80-test.rules

# Análisis de presupuesto

Se asume que la implementación física del laboratorio, lo que involucra los ordenadores de los clientes, la Workstation del administrador y la infraestructura de la red ya está establecida al iniciar con la instalación del sistema de monitoreo. Además, recalcar el uso de software libre para el desarrollo del sistema. Debido a esto los costos solo se distribuyen por la mano de obra utilizada en la implementación del sistema de monitoreo, esto es el desarrollo del código de los scripts para crear el back end y front end del sistema de red, enlazar y subir información del pc a la página creada para que le usuario encargado del laboratorio la administre.

# Conclusiones

Se logro mantener el control de un laboratorio con 5 PC's tal que permitió detectar interrupciones en los equipos mediante la activación de alertas, las mismas que se activan cuando periféricos son removidos, Sesiones inicias y terminadas.

Al hacer uso de frameworks de parte del servidor en backend se logró el apagado remoto de las computadoras del laboratorio mediante el uso de SSH, desde la computadora del administrador.

Para la administración de los recursos y el monitoreo en tiempo real se creó una app web bajo el modelo de Single-Page Application(aplicación de página única) donde se muestren las notificaciones, información de inventario y la interacción con las opciones de control de las computadoras.

# Referencias bibliográficas

Wallen, j. (2018). Viewing linux logs from the command line - linux.com. Retrieved 18 november 2019, from <https://www.linux.com/tutorials/viewing-linux-logs-command-line/>

Kumar, P. (2018). How to capture and analyze packets with tcpdump command on Linux. Retrieved 18 November 2019, from <https://www.linuxtechi.com/capture-analyze-packets-tcpdump-command-linux/>

Parker, S. (2014). Shell Scripting Tutorial. Retrieved 18 November 2019, from <https://www.shellscript.sh/first.html>