**FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN**

**ADMINISTRACIÓN DE SISTEMAS Y SERVICIOS EN RED**

**I PAO 2021**

INFORME TÉCNICO

**APLICACIÓN WEB SOBRE “ANOMALÍAS EN UNA RED DE DATOS USANDO EL PROTOCOLO BGP”**

Grupo: 1

Integrantes:

* Castillo Cusme Gustavo Gerardo
* Indacochea Villamar Nathaly María
* Pulecio Villalva Pedro Tyrone
* Serrano Espinoza Ramiro Sebastian

Profesor: Msig. Adriana Collaguazo

Paralelo :1

Fecha de entrega: 15 agosto 2021

Contenido

[Introducción 3](#_Toc81778297)

[Diseño 3](#_Toc81778298)

[DIAGRAMA DE RED 3](#_Toc81778299)

[DIAGRAMA DE DESPLIEGUE 3](#_Toc81778300)

[DIAGRAMA ENTIDAD-RELACIÓN 5](#_Toc81778301)

[RESULTADOS 5](#_Toc81778302)

[PRESUPUESTO 9](#_Toc81778303)

[PARTE 9](#_Toc81778304)

[CANTIDAD 9](#_Toc81778305)

[VENDEDOR 9](#_Toc81778306)

[PRECIO 9](#_Toc81778307)

[CONCLUSIONES 9](#_Toc81778308)

[APENDICES 9](#_Toc81778309)

# 

# Introducción

La revolución tecnológica permite que la empresa “Hadita S.A” provea un servicio de internet a diferentes destinos locales, para dicho fin hace uso de múltiples equipos y configuraciones con distintos protocolos, se busca que sea una empresa escalable comercialmente, esto quiere decir que se encuentre en crecimiento constante, de esta forma aumentaría también el tamaño de la red. Debido al extenso procesamiento de datos surgen desafíos como el de identificar errores que se causan en las redes.

Por esta razón una propuesta de solución es una topología escalable usando dispositivos cisco, especialmente los routers cisco XR que se manejan en topologías CLOS, gracias a esos routers podremos extraer archivos XML del modelo YANG y así poder analizar las anomalías generadas dentro de nuestra red. El modelo YANG (Putina, Rossi, & Bifet, 2018) que es aceptado por el sector para la modelación de datos y de pila de programabilidad de Cisco será usado para crear un conjunto de datos estructurados que se puede codificar y transporta lo más rápido posible a través de la red. Con esto el administrador de esta información podrá visualizar que error está ocurriendo inmediatamente.

Para este proyecto nos centraremos en crear una aplicación de gestión web que nos permita visualizar las anomalías que se generan cuando se hace uso de una red que tenga empleado una conexión con el protocolo BGP.

# Diseño

DIAGRAMA DE RED

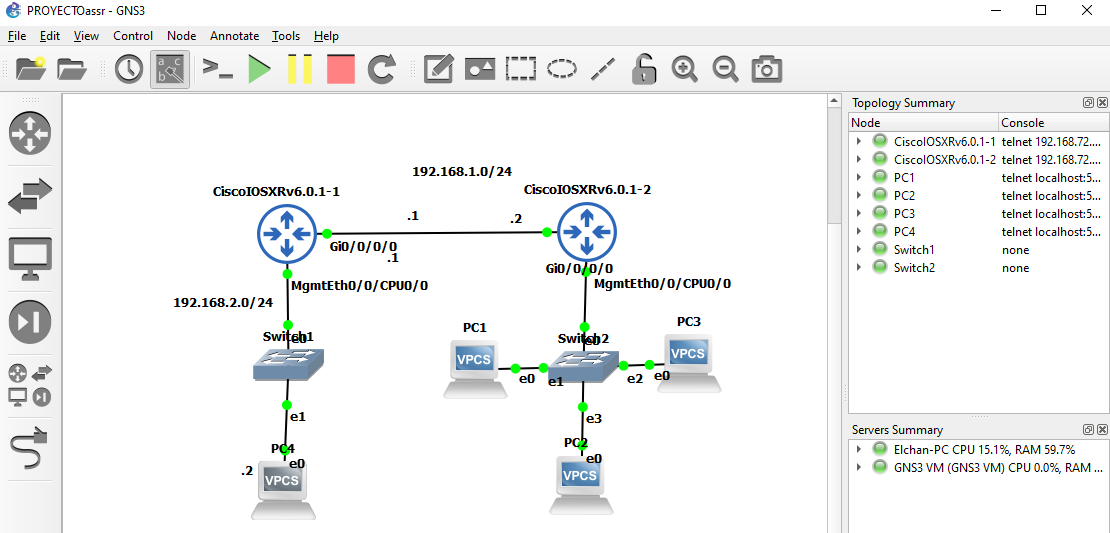


Figura1. Simulación de la comunicación entre dos redes usando protocolo BGP.

DIAGRAMA DE DESPLIEGUE

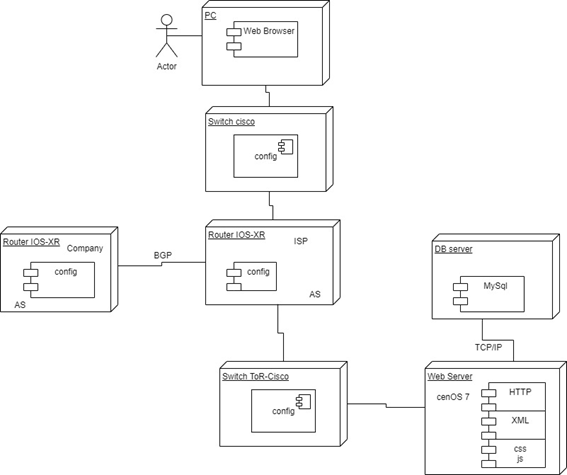


Figura2. Diagrama de despliegue

## DIAGRAMA ENTIDAD-RELACIÓN

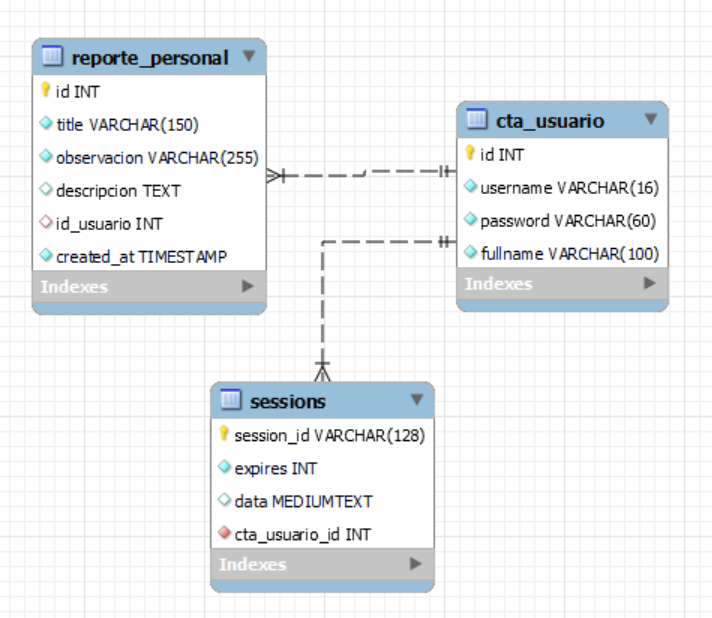


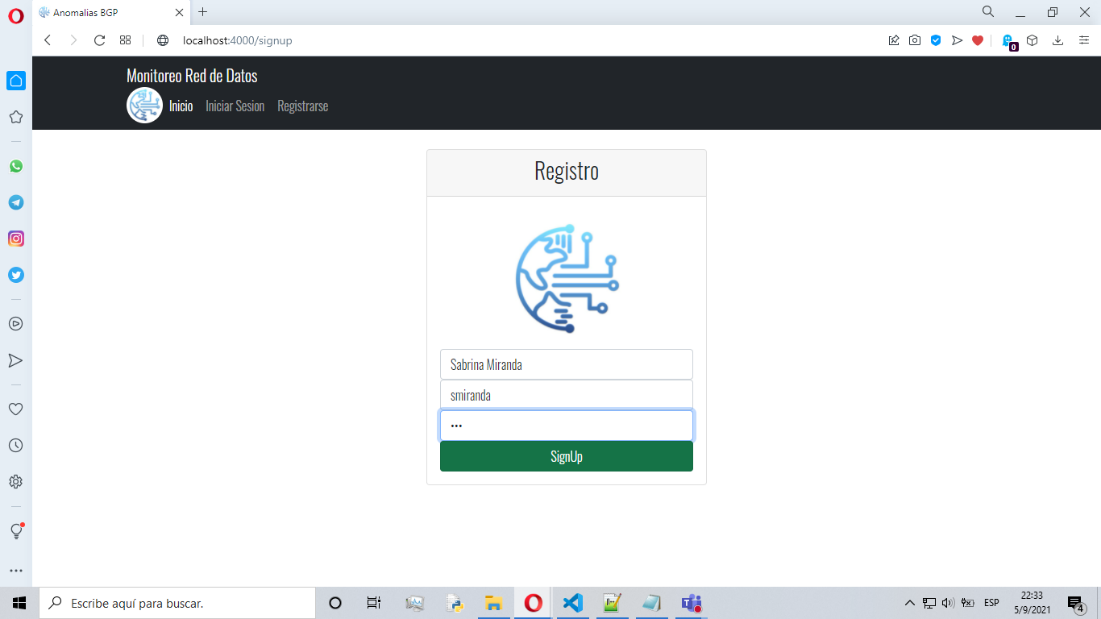
Figura2. Diagrama de entidades que intervienen en la página web.

# RESULTADOS

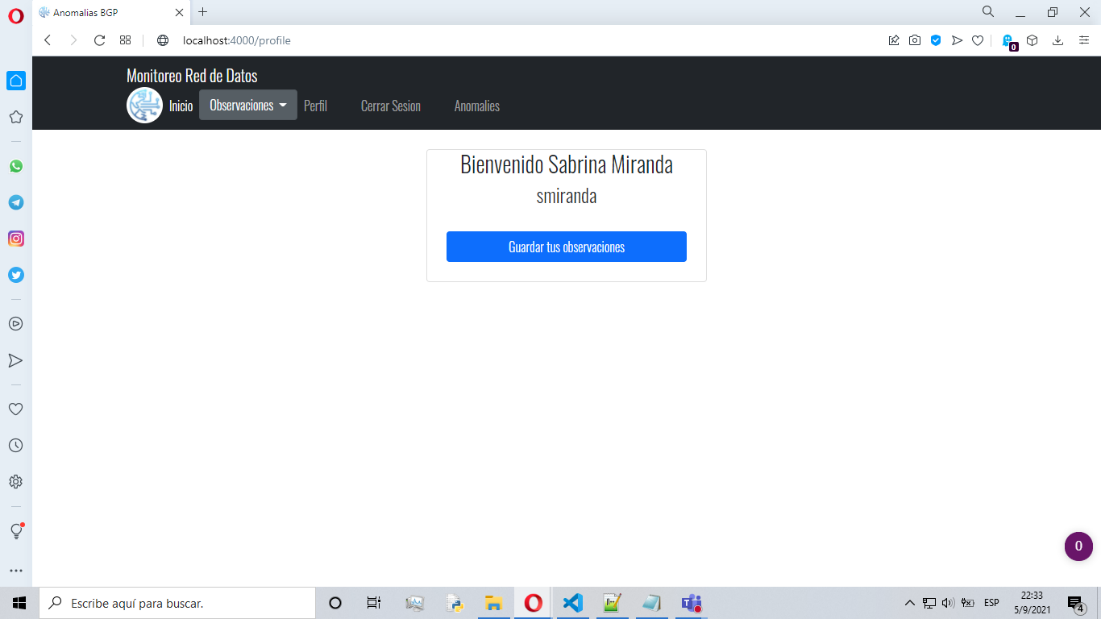
Esta es nuestra pantalla principal de la aplicación web:



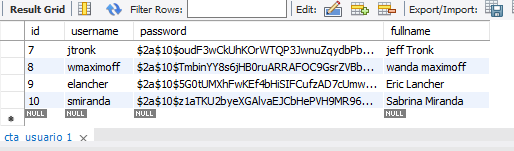
Registro de un usuario:



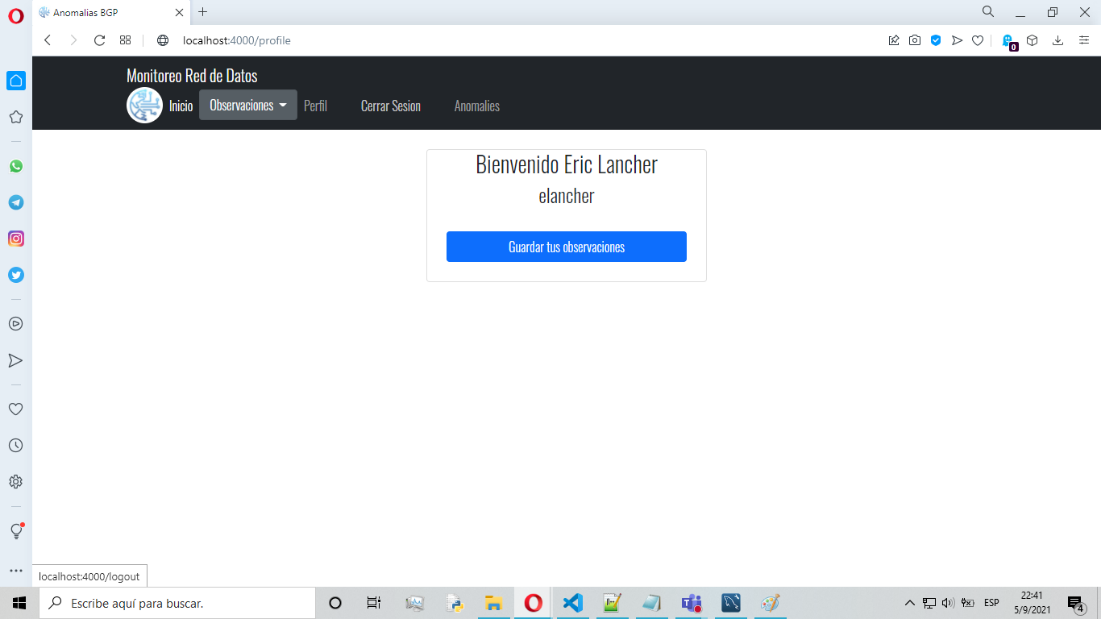
Redirige a perfil de usuario:



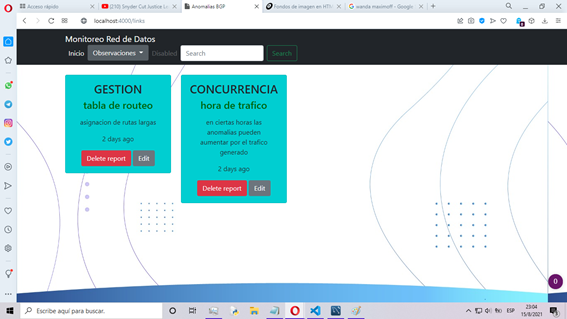
Evidencia que el usuario ya está en la base de datos

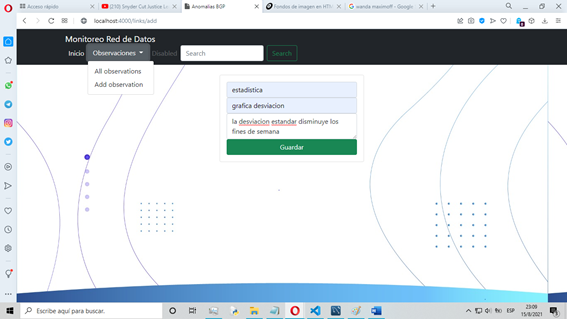


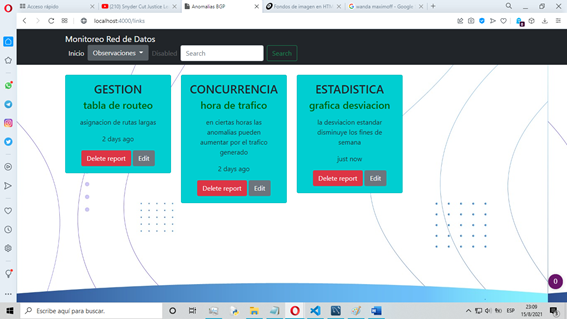
Ejemplo usuario elancher inicia sesión, es redirigido al perfil y luego puede cerrar sesion :

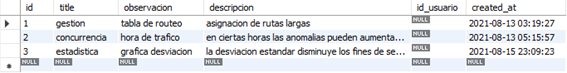


Usuario puede agregar notas, observaciones acerca de lo que los resultados de monitoreo.









Logo de nuestra aplicación:



# PRESUPUESTO

|  |  |  |  |
| --- | --- | --- | --- |
| PARTE | CANTIDAD | VENDEDOR | PRECIO |
| ROUTER CISCO IOS-XR 6.2.2 | **2** | **CISCO** | **$1.960 c/u** |
| Dispositivos  (PC-Local) | **4** | **amazon** | **$900 c/u** |
| CentOS7 (Server) | **1** | **Centos.org** | **Hosting gratuito** |
| Mysql | **1** | **oracle** | **Open-source** |

Tabla1. Presupuesto para aplicación de gestión WEB

# CONCLUSIONES

1. Para esta nueva era informática, es prácticamente indispensable el almacenamiento de información, que gracias al base de datos se facilita el almacenamiento y la manipulación de los datos, pero de todas las formas de almacenamiento las bases de datos relacionales son las más utilites porque evitamos la redundancia de datos , esto facilita el trabajo de los administradores, las base de datos no relacionales funcionan siempre y cuando no sean demasiados grandes como futuros Ing. simple debemos pensar en diseños que tengo una larga vida útil.
2. Api son como un recepcionista que se encargar de enviar nuestras peticiones a los demás departamentos sin que el usuario sepa cómo se están procesando los datos o de donde se los toma esto le da un nivel de seguridad a nuestra aplicación.
3. Una de las ventajas de los routers configuración IOS-XR es que se basa en QNX(que es un sistema operativo similar a Unix). Las aplicaciones de IOS-XR se han separado en diferentes procesos y su alguno de los procesos se detiene o funciona mal, los otro proceso no se ven afectados.

**REPOSITORIO GITHUB:** <https://github.com/GustavoTetian96/ApplicationWebAnomalies.git>

# APENDICES

Putina, A., Rossi, D., & Bifet, A. (20 de Agosto de 2018). *Telemetry-based stream-learning of BGP anomalies*. Obtenido de <https://dl.acm.org/doi/pdf/10.1145/3229607.3229611>

**Index.js**

|  |
| --- |
| const express = require('express');  const morgan = require('morgan');  const path= require('path');  const exphbs = require('express-handlebars');  const flash = require('connect-flash');  const session = require('express-session');  const mysqlstore = require('express-mysql-session');    const passport = require('passport');    const { database } = require('./keys');    //inicialization  const app = express();  require('./lib/passport');  //settings  app.set('port', process.env.PORT || 4000);  app.set('views', path.join(\_\_dirname,'views'));  app.engine('.hbs',exphbs({  defaultLayout: 'main',  layoutsDir: path.join(app.get('views'),'layouts'),  partialsDir: path.join(app.get('views'),'partials'),  extname: '.hbs',  helpers: require('./lib/handlebars')    }));  app.set('view engine','.hbs');    //middlewares  app.use(session({  secret: 'castillomysqlnodesession',  resave: false,  saveUninitialized: false,  store: new mysqlstore(database)  }));    app.use(flash());    app.use(morgan('dev'));  app.use(express.urlencoded({extended: false}));  app.use(express.json());    app.use(passport.initialize());  app.use(passport.session());  //routes  app.use(require('./routes'));  app.use(require('./routes/authentication'));  app.use('/links',require('./routes/links'));    //globaal variables  app.use((req,res,next)=>{  app.locals.success =req.flash('success');  next();  });    //public  app.use(express.static(path.join(\_\_dirname, 'public')));  //starting the server  app.listen(app.get('port'), () =>{  console.log('server on port ', app.get('port'));  }); |

**Database.js**

|  |
| --- |
| const mysql = require('mysql');  const {promisify} = require('util');  const {database} = require('./keys');    const pool = mysql.createPool(database);    pool.getConnection((err,connection) =>{  if(err){  if(err.code === 'PROTOCOL\_ CONECCTIONS \_LOST'){  console.error('DATABASE CONNECTION WAS CLOSED');  }  if(err.code === 'ER\_CON\_COUNT\_ERROR'){  console.error('DATABASE HAS TO MANY CONNECTIONS');  }  if(err.code === 'ECONNREFUSED'){  console.error('DATABASE CONNECTION WAS REFUSED');  }    }  if(connection) connection.release();  console.log('DB is Connected');  return;  });    pool.query = promisify(pool.query);    module.exports = pool; |

**Keys.js**

|  |
| --- |
| module.exports ={  database: {  host: 'localhost',  user: 'root',  password: 'Tetian96',  database: 'anomalias\_bgp'  }  } |

**Links.js**

|  |
| --- |
| const express = require('express');  const router = express.Router();    const pool = require('../database');  const {isLoggedIn} = require('../lib/validauth');  router.get('/add' , isLoggedIn,(req,res)=>{  res.render('links/add');  });    router.post('/add',isLoggedIn,async(req,res)=>{  const {title,observacion,descripcion}= req.body;  const newObservation = {  title,  observacion,  descripcion,  id\_usuario: req.user.id  };  await pool.query('INSERT INTO reporte\_personal set ?',[newObservation]);  req.flash('success', 'report saved successfully');  //console.log(await pool.query('select \* from reporte\_personal'));  //console.log(newObservation);  res.redirect('/links');  });    router.get('/',isLoggedIn ,async (req,res) =>{  const reportes = await pool.query('select \* from reporte\_personal where id\_usuario=?',[req.user.id]);  res.render('links/list',{ reportes });  });    router.get('/delete/:id',isLoggedIn,async (req,res) =>{  const { id } = req.params;  await pool.query('DELETE FROM reporte\_personal WHERE ID = ?', [id]);  req.flash('success', 'report Deleted successfully');  res.redirect('/links');    });    router.get('/edit/:id',isLoggedIn,async (req,res)=>{  const { id } = req.params;  const reportes =await pool.query('SELECT \* FROM reporte\_personal WHERE ID= ?',[id]);  console.log(reportes[0]);  res.render('links/edit',{ reporte: reportes[0] });  });    router.post('/edit/:id',isLoggedIn,async(req,res)=>{  const { id } = req.params;  const {title, observacion, descripcion }= req.body;  const newreport = {  title,  observacion,  descripcion  };  console.log(newreport);  await pool.query('UPDATE reporte\_personal set ? WHERE id = ?',[newreport,id]);  res.redirect('/links');  });  module.exports = router; |

**Authentication.js**

|  |
| --- |
| const express = require('express');  const router = express.Router();  //const passport = require('../lib/passport');  const passport = require('passport');  const {isLoggedIn} = require('../lib/validauth');  router.get('/signup', (req,res) =>{  res.render('auth/signup');  });    router.post('/signup',  //console.log(req.body);  passport.authenticate('local.signup',{  successRedirect: '/profile',  failureRedirect: '/signup',  failtureFlash: true    }  )  //res.send('ingresado');  );    router.get('/signin',(req, res)=>{  res.render('auth/signin');  });    router.post('/signin',(req,res,next)=>{  passport.authenticate('local.signin',{  successRedirect: '/profile',  failureRedirect: '/signin',  failureFlash: true  })(req, res, next);  });    router.get('/profile',isLoggedIn,(req,res)=>{  //res.send('este es tu perfil');  res.render('profile'); //renderizo  });  router.get('/logout',(req,res)=>{  req.logOut();  res.redirect('/signin');  });    module.exports = router; |