

Zerolink

Area	Escolar
Estado	En proceso
Fin	@May 31, 2021
Inicio	@Apr 30, 2021

El presente proyecto se desarrollo para servir como entrega de ambas materias.

El código del proyecto se encuentra en Github y se presenta en la sección repositorio.

El funcionamiento del proyecto se muestra mediante un video, el link de este se encuentra en la sección video.

Introducción

El presente documento plantea el desarrollo de un proyecto conjunto de las materias de domótica y temas de ingeniería de software de tal forma que sea posible integrar los conocimientos adquiridos y potenciales de ambas materias.

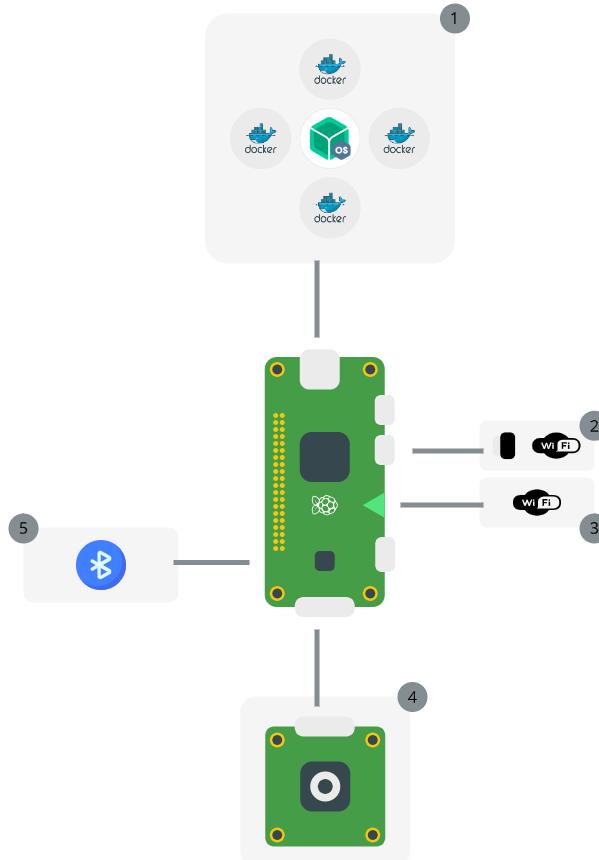
Lo anterior se plantea en el entendido de generar una evaluación conjunta y robusta que enaltezca los principios rectores de las materias, posibilitando el alcanzar objetivos más haya de los planteados por el programa de las materias.

De forma particular se busca generar un proyecto que proponga el prototipo de un producto de consumo final enfocado en la satisfacción de necesidades domóticas cuyo interior este integrado por una arquitectura de microservicios. De igual forma se desarrollara un sistema basado en la microservicios y la nube que sirve de interfaz entre el usuario final y los dispositivos embebidos.

Arquitectura

El dispositivo domótico esta basado en la tarjeta raspberry pi zero w, la cual fungirá como plataforma de control y comunicación. Esta posee características tales como:

- Dimensiones 66.0mm x 30.5mm x 5.0mm
- 1GHz, single-core CPU
- 512MB RAM
- Mini HDMI
- 2 Micro usb (datos y energía)
- 802.11 b/g/n wireless LAN
- Bluetooth 4.1
- CSI camera connector



Arquitectura física del dispositivo

Una vez conocida la plataforma se debe detallar que no se utilizaran el total de los periféricos del dispositivo por lo cual se listan los elementos a utilizar y la forma en que se usaran.

1. Sistema operativo

Puesto que se desea utilizar una arquitectura basada en microservicio de forma interna se opto por utilizar el SO Balena OS, el cual monta un kernel linux y un gestor de contenedores.

- El kernel controla el funcionamiento a nivel lógico gestionando los periféricos y respondiendo las solicitudes lógicas del gestor de contenedores.
- El gestor de contenedores gestiona las aplicaciones, donde básicamente es posible diseñar aplicaciones como microservicios y el sistema los controla mediante docker compose

Cabe destacar que a diferencia de otros sistemas Balena OS emplea muy pocos recursos, pues es posible montarlo en una tarjeta micro sd de como mínimo 4GB ocupando únicamente 1GB, dejando el resto de espacio para las aplicaciones.

2. Dongle wifi

Entiéndase como un dispositivo wifi-usb externo el cual servirá como punto de acceso para los dispositivos a la raspberry pi.

3. Wifi integrado

Dispositivo wifi interno el cual se conectara a al router central para proveer acceso a internet.

4. Cámara

Cámara raspberry la cual provee imagenes o video en 720p, su conexión es nativa mediante el puesto CSI de las raspberry pi.

5. Bluetooth

Radio bluetooth de baja energía con integración nativa raspberry pi el cual facilita la conexión de dispositivos tales como smartphones, sistemas de sonido, balizas, etc.

Aplicaciones

Una vez que se presento la arquitectura es posible presentar las aplicaciones que justifican la integración de ambas materias mediante el presente proyecto.

La plataforma integra componente de software y hardware y ha sido nombrada zerolink, definiéndose de la siguiente forma:



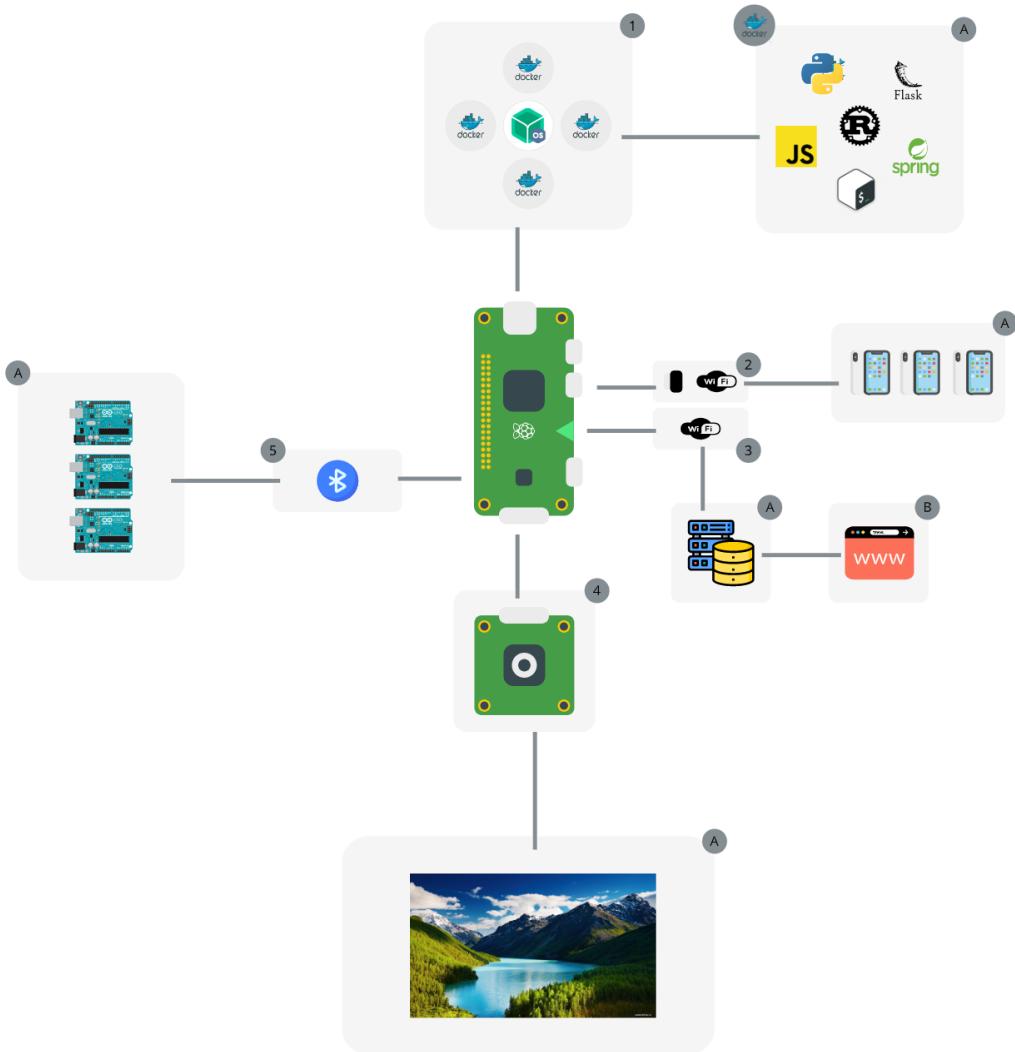
Zerolink es un dispositivo que combina una cámara de seguridad en tiempo real dotada de reconocimiento facial e integraciones de aprendizaje automático que además funge como access point el cual al ser puesto en función junto con más dispositivos zerolink creará una red mesh que proveera de internet estable a toda la casa así como de seguridad.

Como se puede ver en la definición zerolink es un dispositivo que puede integrarse a las casas facilitando la seguridad y posibilitando la integración de dispositivos que aumenten las funciones originales de zerolink.

Por ejemplo será posible integrar dispositivos como focos, enchufes, electrodomésticos o un sinfín de sensores y actuadores, sin embargo, no hay que perder de vista que zerolink es un dispositivo domótico en sí.

Y si a esto le agregamos que zerolink interna y externamente se basa en microservicios esto hace que la plataforma sea más interesante.

Arquitectura de las aplicaciones



Arquitectura lógica del dispositivo zerolink

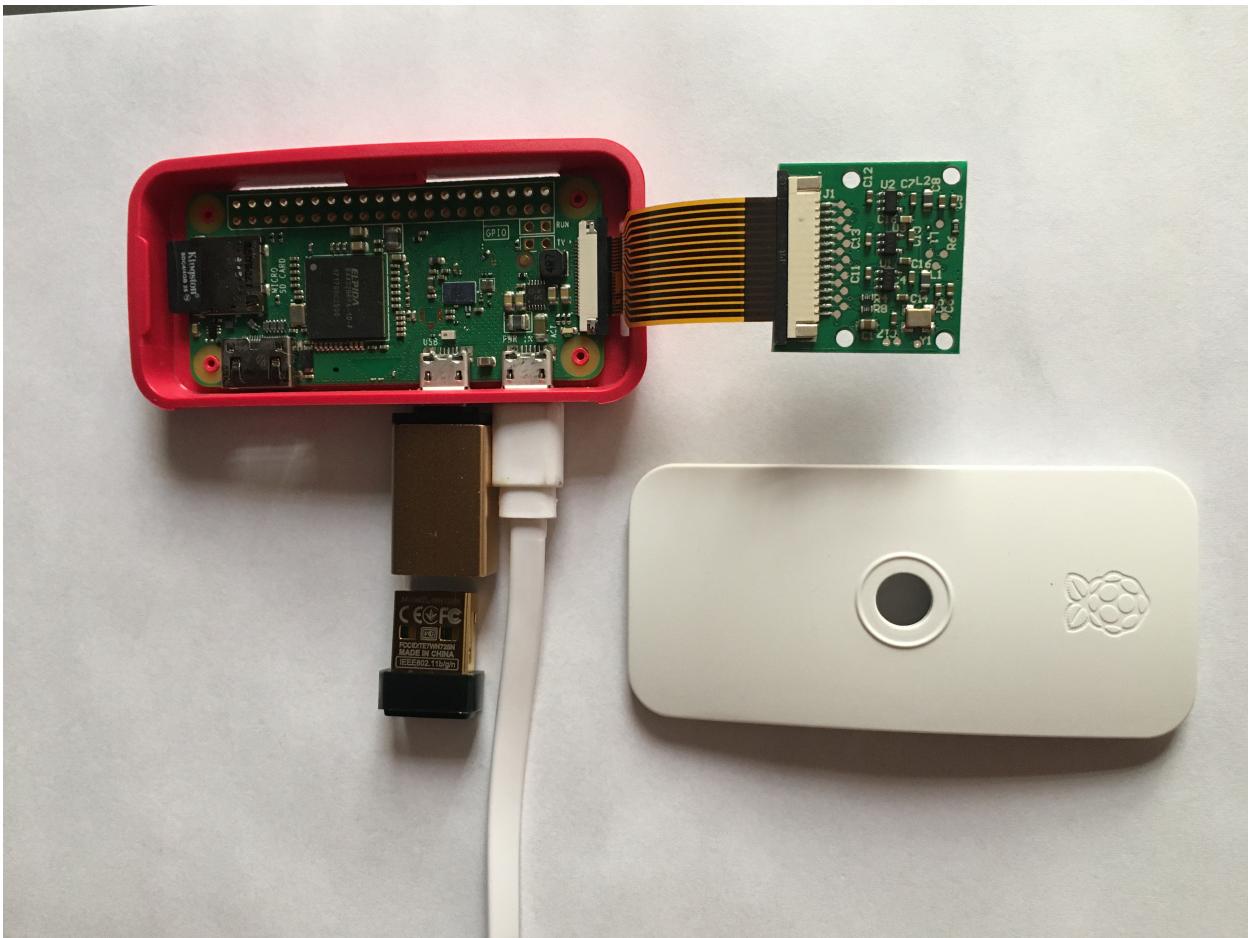
1. Sistema operativos
 - A. Aplicaciones
2. LAN
 - A. Dispositivos móviles
3. WLAN
 - A. Servidor
 - B. Cliente
4. Cámara
 - A. Objeto de captura
5. Bluetooth
 - A. Dispositivos IoT
 - Arduino
 - Focos

- Enchufes
- Balizas

Desarrollo

Dispositivo

Al momento el dispositivo se encuentra construido en un 80%, esto ya que el dispositivo tiene dos elementos, por un lado la electrónica (la cual se refleja en el diagrama de arquitectura) y por otro lado la estructura física que contendrá a los componentes electrónicos.



Estructura del circuito con componentes y enclosure rpi



Estructura general del circuito



Camara RPI Sony de 5 mpx

- Ya se cuenta con la estructura que albergara al los componentes electrónicos, sin embargo, aún es necesario hacer adaptaciones para su correcto montaje.



Vista superior del enclosure del circuito



Vista frontal del enclosure del circuito



Vista isometrica del enclosure ensamblado

Sistema operativo

Al momento el sistema operativo ha sido instalado y configurado al 100%

The screenshot shows the balenaCloud dashboard with the Zerolink application selected. The left sidebar lists organizations, applications, and the current application 'zerolink'. The main area has three main sections: 'Devices' (1 device), 'Releases' (2 releases), and a detailed view of the 'zerolink' application. The application summary includes a Raspberry Pi icon, architecture (rpi), creation date (May 11th 2021, 10:10 am), and a 'Starter' button. Below this is a table of devices, showing one entry: 'zerow' (Offline, Raspberry Pi v1 / Zero / Zero W, last seen 4 days ago, created on May 11th 2021, 1:40 pm, UUID 1b9f87f, OS version balenaOS 2.54.2+rev1). The bottom of the screen shows a footer with 'Dashboard del sistema operativo'.

Microservicios domóticos

Hasta el momento se tienen contempladas 4 microservicios domóticos los cuales puede variar de acuerdo a si el proyecto es o no en equipo o inclusive al número de integrantes de este.

Microservicio 1: API de usuarios

Es una utilería destinada a la gestión de usuarios dentro de una red LAN de tal forma que una red domestica posea perfiles de uso donde se controlen accesos e inclusive un control parental.

La presente API puede estar implementada en cualquier framework web considerando como buenas opciones flask (python) o spring boot (java).

The screenshot shows a browser window with multiple tabs. The active tab displays a Python Flask route for adding a user:

```

@app.route('/add<user>/<email>/<password>', methods=['GET'])
def insertOne(user, email, password):
    queryObject = {
        'User': user,
        'Email': email,
        'Password': password
    }
    query = zeroUsers.insert_one(queryObject)
    return "query insertado"

```

To the right of the code, there is a message: "Comienza a construir con la API de ZeroLink".

```

1  from flask import Flask, render_template, jsonify, request
2  from flask_cors import CORS
3  import pymongo
4
5  app = Flask(__name__)
6
7  client = pymongo.MongoClient('mongodb+srv://raul:3312@cluster0.pemn.mongodb.net/zero?retryWrites=true&w=majority')
8  db = client.zero
9  zeroUsers = db.zeroUsers
10
11 @app.route('/')
12 def holaMundo():
13     return jsonify(render_template("index.html"))
14

```

https://api-1.raulcrush.replit.co

Comienza a construir con la API de Zerolink

Console Shell

```

172.18.0.1 - - [11/Jun/2021 08:22:48] "GET /add/Raul/raul@mail.com/FbVK4E35/Admin/ HTTP/1.1" 404 -
172.18.0.1 - - [11/Jun/2021 08:23:42] "GET /add/raul/raul@mail.com/FbVK4E35/admin/ HTTP/1.1" 404 -
172.18.0.1 - - [11/Jun/2021 08:23:43] "GET /add/raul/raul@mail.com/FbVK4E35/admin/ HTTP/1.1" 404 -
172.18.0.1 - - [11/Jun/2021 08:27:06] "GET /get/ HTTP/1.1" 200 -

```

```
{"0": {"email": "raul@mail.com", "password": "FbVK4E35", "profile": "admin", "user": "raul"}, "1": {"email": "roy@mail.com", "password": "ZnSt6MgP", "profile": "usuario", "user": "roy"}, "2": {"email": "karly@mail.com", "password": "Y4UsJAKr", "profile": "invitado", "user": "karly"}}
```

Microservicio 2: Wifi Access Point

WiFi Access Point es una utilidad para extender dinámicamente una red cableada o inalámbrica existente. Para lograr esto, la utilidad utiliza el chip WiFi integrado para crear un punto de acceso al que puede conectar sus dispositivos. Para habilitar el acceso a Internet, puede conectar un cable Ethernet (modo AP) o utilizar una interfaz WiFi secundaria (mediante un dispositivo USB WiFi) para conectarse a una red habilitada para Internet.

El repetidor WiFi puede funcionar en los siguientes modos:

- Punto de acceso: amplíe una conexión Ethernet existente con una red de punto de acceso
- Repetidor: extiende una conexión inalámbrica existente con una red de punto de acceso. Requiere el uso de un dongle USB WiFi

Microservicio 3: Cámara web RTC con inteligencia artificial

Este proyecto de Raspberry Pi utiliza WebRTC (más sobre eso más adelante) para la comunicación entre la cámara y el navegador, y le permite acceder a una transmisión de video desde su dispositivo desde cualquier parte del mundo utilizando la función de URL pública de balenaCloud.

```

import io
import picamera
import logging
import socketserver
from threading import Condition
from http import server

PAGE="""\
<html>
<head>
<title>Raspberry Pi - Surveillance Camera</title>
</head>
<body>
<center><h1>Raspberry Pi - Surveillance Camera</h1></center>
<center></center>
</body>
</html>
"""

class StreamingOutput(object):
    def __init__(self):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            # New frame, copy the existing buffer's content and notify all
            # clients it's available
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)

class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        if self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace; boundary=FRAME')
            self.end_headers()
            try:
                while True:
                    with output.condition:
                        output.condition.wait()
                        frame = output.frame
                    self.wfile.write(b'--FRAME\r\n')
                    self.send_header('Content-Type', 'image/jpeg')
                    self.send_header('Content-Length', len(frame))
                    self.end_headers()
                    self.wfile.write(frame)
                    self.wfile.write(b'\r\n')
            except Exception as e:
                logging.warning(
                    'Removed streaming client %s: %s',
                    self.client_address, str(e))
            else:
                self.send_error(404)
                self.end_headers()
        else:
            self.send_error(404)
            self.end_headers()

class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):
    allow_reuse_address = True
    daemon_threads = True

with picamera.PiCamera(resolution='640x480', framerate=24) as camera:
    output = StreamingOutput()
    # Uncomment the next line to change your Pi's Camera rotation (in degrees)
    #camera.rotation = 90
    camera.start_recording(output, format='mjpeg')
    try:
        address = ('', 8000)
        server = StreamingServer(address, StreamingHandler)
        server.serve_forever()
    finally:
        camera.stop_recording()

```

Microservicios web

Microservicio 1: Base de datos

Este servicio refiere a una base de datos o conjunto de bases de datos alojados en la nube. En este apartado se considera una diferenciación al tipo de base de datos necesaria para la función requerida.

Por ejemplo para datos de usuario o conjuntos enriquecidos se requiere Mongo DB, sin embargo, si los datos a almacenar provienen de sensores o dispositivos domóticos se utilizará Influx DB.

The screenshot shows the MongoDB Atlas Data Explorer interface. The top navigation bar includes tabs for 'zerodb', 'Atlas', 'Realm', and 'Charts'. The main area displays 'QUERY RESULTS 1-3 OF 3' with three user documents:

```

_id: ObjectId("60c31e1150f40926db84ed76")
email: "raul@mail.com"
user: "raul"
password: "FbVK4E35"
profile: "admin"

_id: ObjectId("60c31e6d50f40926db84ed77")
email: "roy@mail.com"
user: "roy"
password: "ZnSt6MgP"
profile: "usuario"

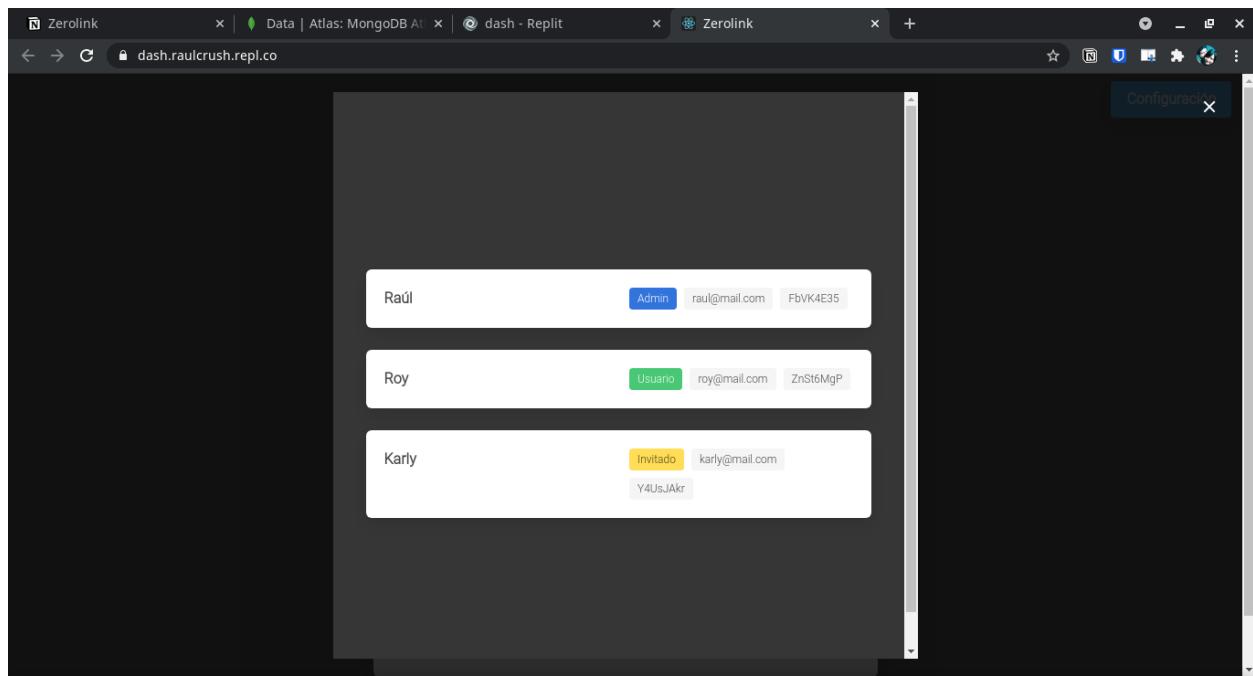
_id: ObjectId("60c31e9e50f40926db84ed78")
email: "karly@mail.com"
user: "karly"
password: "Y4UsJAkz"
profile: "invitado"

```

Microservicio 2: Dashboard domótico

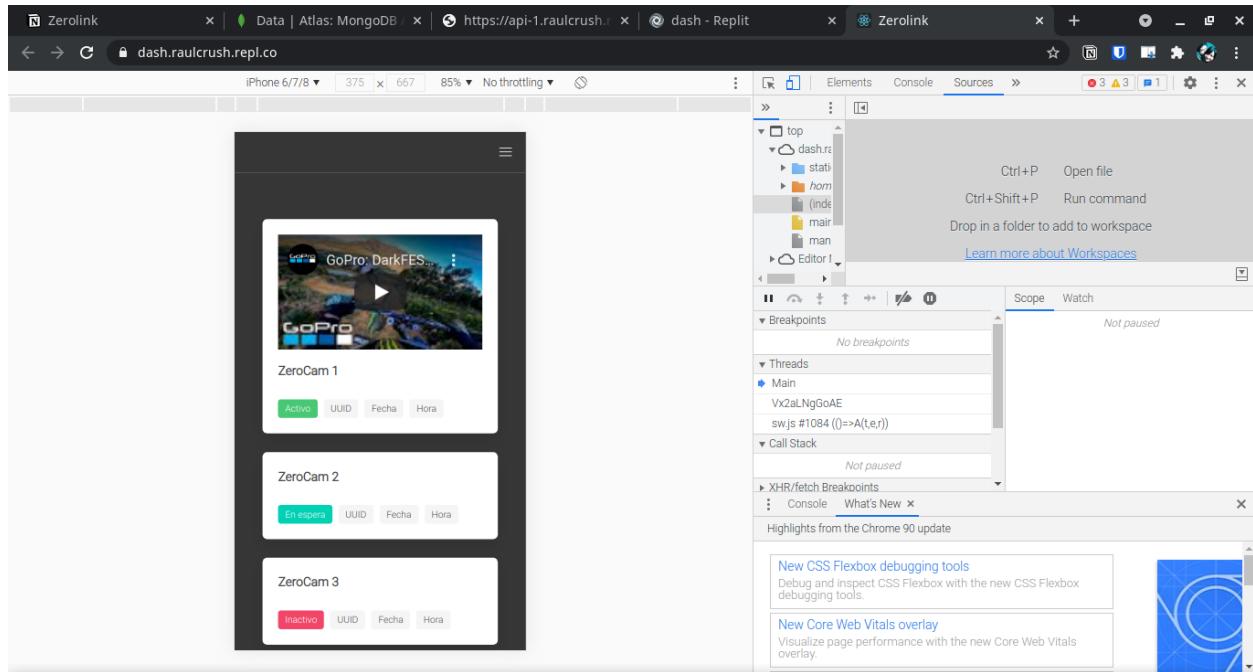
Este servicio se compone de un conjunto de microservicios los cuales serán componentes React JS o Gatsby JS para la presentación de datos y control de los nodos zerolink.

Además de que se considera la versión offline o PWA la cual facilitara el soporte de un frontend cuasi nativo.



Microservicio 3: Frontend cámara RTC

Refiere al componente React JS necesario para presentar la visualización en tiempo real de las camaras en los nodos zerolink



Video

A continuación se presenta la evidencia audiovisual del funcionamiento del proyecto.

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c94e3649-3f74-4569-8d2f-3da7874ad82b/zerolink_funcionamiento.mp4

Repositorio

En la siguiente liga se encuentra accesible el código fuente, las configuraciones, diagramas e inclusive el presente documento.

raulszvz/zerolink
Area: Escolar Estado: En proceso Fin: May 31, 2021 Inicio: Apr 30, 2021 Participantes: Alejandra Perez Guerrero, Fernando Yael Ortega Guizar, Karla Valeria Perez Perez, Raúl Sánchez Vázquez, Rogelio Robledo Moreno URL: <https://zerolink.webml.studio> El presente documento plantea el desarrollo de un proyecto conjunto de las materias de domótica y temas de ingeniería de software de tal forma que sea posible integrar los

<https://github.com/raulszvz/zerolink>

Conclusiones

Cuando se habla de IoT se piensa en dispositivos pequeños con baterías de larga duración y que hacen más fácil la vida, sin embargo, cuando se habla de IoT mediante sistemas embedidos (como raspberry pi o arduino) es común que se asocie con prototipos cuya funcionalidad queda legada a actividades recreativas.

Es por ello que se planteo el desarrollo de zerolink como un dispositivo que aunque tiene por base un sistema embedido posibilita el desarrollo de aplicaciones comerciales.

Por tanto, zerolink se cataloga como un dispositivo que favorece el intercambio de información y la seguridad de forma eficiente y a un bajo costo, compitiendo con desarrollo costosos con plataformas complejas. Ademas que este dispositivo cuenta con dos grandes funcionalidades lo que lo hace aun mejor, ya que se puede utilizar para mantener seguro un sitio, ya sea lugar de trabajo o en el mismo hogar por su sistema de reconocimiento facial con inteligencia artificial, y por otra parte el poder servir como red mesh que nos ampliará las posibilidades.

Con esto se pudo ver que con ambas materias usadas a la par se pueden crear sin fin de proyectos, siendo este uno de ellos, ya que al optar por usar microservicios su implementación resultó ser más fácil y rápida entonces obtuvimos un producto del cual se puede sacar provecho, y ayudara a mejorar los hogares en el ámbito de seguridad y conectividad.

Referencias

Get started with Raspberry Pi 3 and Node.js - Balena Documentation

In this guide, we will build a simple Node.js web server project on a Raspberry Pi 3. At its most basic, the process for deploying code to a Raspberry Pi 3 consists of two major steps: Setting up your Raspberry Pi 3 with balenaOS, the host OS that manages communication with balenaCloud and runs the core device operations.

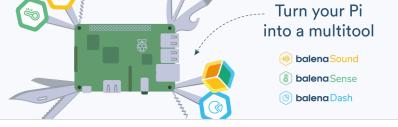
<https://www.balena.io/docs/learn/getting-started/raspberrypi3/nodejs/>



Two projects, one device: turn your Raspberry Pi into a multitool

Built one of our projects, and want to expand upon it? Perhaps you want to build a second one, but lack the hardware? Fear not! One of the great benefits of balenaCloud is that you can combine different projects and run everything on one device.

<https://www.balena.io/blog/two-projects-one-device-turn-your-raspberry-pi-into-a-multitool/>



Turn a Raspberry Pi into a Wi-Fi access point or repeater

Remote work brings more opportunity to tackle projects from different parts of the home. However, without strong enough Wi-Fi strength, your productivity might sink. Use this project to create a Wi-Fi access point or repeater out of a Raspberry Pi so you can improve signal coverage while working from home or remotely on the road.

<https://www.balena.io/blog/turn-a-raspberry-pi-into-a-wi-fi-access-point-or-repeater/>



Build a Raspberry Pi-based network camera using WebRTC

Do you want to be able to monitor your house, workplace, husband, wife, kids, pets, etc. when you are not there? Do you have a Raspberry Pi and a camera laying around? If you answered yes to at least one of these questions, keep reading! Updated 9th January 2020 -

<https://www.balena.io/blog/build-a-raspberry-pi-based-network-camera/>



balenalabs-incubator/wifi-repeater

WiFi Repeater project is a utility to dynamically extend an existing wired or wireless network. To achieve this the utility uses the onboard WiFi chip to create an access point you can connect your devices to.

<https://github.com/balenalabs-incubator/wifi-repeater>



balena-io-examples/balena-python-hello-world

This is a simple skeleton Flask server project that works on any of the devices supported by balena. This project simply serves up "Hello World!" on port :80 of your balena device. To get this project up and running, you will need to signup for a balena account here and set up a device, have a look at our Getting Started tutorial.

<https://github.com/balena-io-examples/balena-python-hello-world>



Video Streaming Raspberry Pi Camera | Random Nerd Tutorials

In this post we're going to show you how you can do video streaming with a Raspberry Pi and a Raspberry Pi Camera - how to stream live video into a web page that you can access in any device that has a browser and is connected to the same network the Pi is.

<https://randomnerdtutorials.com/video-streaming-with-raspberry-pi-camera/>



Instalar y configurar el servicio DNS dinámico Duck DNS con Docker

A continuación veremos como instalar y configurar el servicio DNS dinámico Duck DNS con Docker. Pero antes de todo veremos de forma breve que es un DNS dinámico y porque Duck DNS es una opción a tener muy en cuenta. La función de un servicio dinámico DNS es facilitar el acceso a un servidor que tiene una IP pública dinámica.

<https://geekland.eu/instalar-y-configurar-duck-dns-con-docker/>



Make Python API to access Mongo Atlas Database - GeeksforGeeks

Prerequisite: Python | Build a REST API using Flask RESTful APIs are a very helpful tool to manage the backend of an application. It can be used to define the architecture as well as manage the dependencies of the application using routes managed by the API.

<https://www.geeksforgeeks.org/make-python-api-to-access-mongo-atlas-database/>



Anexos

[Things as a Service](#)

[Integración de un actuador](#)

[Integración de un sensor](#)

[Automatización](#)

[Otras formas de interacción](#)