Integració Hardware Software en una plataforma Low-Code

Roger López Puigbò

26 de febrer de 2024

Resum– Aquest projecte busca integrar de manera fluida el Gestor de Processos PM2 en una plataforma de low-code (*Low-Code*), oferint als usuaris una manera intuïtiva i eficient de gestionar i supervisar diversos processos informàtics. La solució integrada inclou una interfície *front-end* dedicada dins la plataforma de low-code, que destaca diverses funcions de PM2, incloent-hi el monitoratge en temps real amb paràmetres com la CPU i la memòria utilitzada. Els usuaris podran explorar i interactuar visualment amb les funcionalitats del PM2 mitjançant una interfície gràfica fàcil d'usar. A més, el projecte incorpora una demostració *Hardware* per il·lustrar la versatilitat del PM2 en la gestió de projectes diversos en diferents entorns informàtics. Una Raspberry Pi, equipada amb un motor servo i tres LEDS com a components representatius del *Hardware*, serà utilitzada per demostrar la capacitat del PM2 en gestionar i coordinar diverses tasques en dispositius heterogenis.

Paraules clau– PM2, plataforma de low-code, gestor de processos, interfície *front-end*, monitoratge en temps real, dispositius heterogenis.

Abstract– This project aims to seamlessly integrate the PM2 Process Manager into a *Low-Code* platform, offering users an intuitive and efficient way to manage and monitor diverse computing processes. The integrated solution features a dedicated *front-end* interface on the *Low-Code* platform, showcasing PM2's diverse functions, including real-time monitoring of parameters like CPU and memory usage in running processes. Users will be able to visually explore and interact with the PM2 functionalities through an easy-to-use graphical interface.
Additionally, the project incorporates a *Hardware* demonstration to illustrate PM2's versatility in managing diverse projects across different computing environments. A Raspberry Pi, equipped with a servo motor and three LED as a representative *Hardware* components, will be utilized to showcase PM2's ability to handle and coordinate various tasks across heterogeneous devices.

Keywords– PM2, *Low-Code* platform, process management, *front-end* interface, real-time monitoring, heterogeneous devices.

1 Introducció - Context del treball

En el món digital actual, el monitoratge i la gestió dels processos de computació són crucials per garantir un rendiment i una eficiència òptims. PM2 Process Manager és una eina popular entre desenvolupadors i administradors de sistemes a causa de la seva capacitat de gestionar diversos projectes en diferents entorns. Aquest treball se centra a integrar a la perfecció el gestor de processos PM2 en una plataforma *Low-Code*, proporcionant als usuaris una manera intuïtiva i eficient de gestionar i monitorar els seus processos informàtics.

La solució integrada ofereix un *front-end* dedicat que mostra les diverses funcions de PM2, incloent-hi el monitoratge en temps real de paràmetres com CPU i l'ús de memòria en processos en execució. Els usuaris poden explorar i interactuar visualment amb les funcionalitats de PM2 a través d'aquesta interfície intuïtiva sense haver de navegar a través de complexes línies d'ordres o fitxers de configuració.

El projecte també incorpora un component back-end que permet una comunicació perfecta entre la plataforma de low-code i el gestor de processos PM2, permetent als usu-

• E-mail de contacte: roger.lopezp@autonoma.cat
• Mención realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Remo Suppi Boldrito (Departament d'Arquitectura dels Computadors i Sistemes Operatius)
• Curs 2023/24

aris gestionar els seus processos de computació des de la mateixa plataforma.

En destaquem que la implementació de les parts de programari (SW)[1] i maquinari (HW)[2] es troben en diferents repositoris, permetent una gestió més independent i una millor escalabilitat en el desenvolupament i manteniment. En conjunt, aquest treball de final de grau presenta una solució pràctica que cobreix la bretxa entre la gestió de *Software* i *Hardware*. Aquesta plataforma open-source[3], desenvolupada per l'empresa Protofy.xyz, és un sistema Full-Stack i low-code per web/apps i IoT amb un a API i missatgeria en temps real que es pot fer servir com base per crear prototips ràpids d'aplicacions, webs, sistemes IoT, automatitzacions o APIs.

2 OBJECTIUS

Tenint en compte que els objectius generals del projecte s'involucra la integració coherent del PM2 a la plataforma de low-code existent, assegurant adaptabilitat, millora de l'experiència d'usuari i una relació cohesionada entre els nous components afegits i els ja establerts, es pot procedir a catalogar els objectius més específics:

1. **Integració amb components existents:**
   - Modificar i adaptar els mòduls actuals dins de la plataforma de low-code per integrar sense problemes el Gestor de Processos PM2.
   - Aconseguir arrancar la plataforma low-code en un microcontrolador Raspberry per a projectes futurs de l'empresa.
2. **Creació de la interfície d'usuari:**
   - Crear una interfície gràfica que exhibeixi de manera clara els detalls de les funcionalitats integrades de PM2.
   - Desenvolupar una interfície gràfica que possibiliti la modificació en temps real dels components físics, executant els programes amb PM2.
3. **Adaptació al disseny recent i mòdul de plantilla:**
   - Adaptar els mòduls i components existents dins de la plataforma de low-code per alinear-se amb els principis de disseny del Gestor de Processos PM2 integrat.
   - Establir una comunicació eficaç entre les dades al *back-end* i les sol·licituds enviades des del front-end.
4. **Resolució de problemes existents i adaptació als canvis:**
   - Identificar i abordar problemes específics ja existents dins de la startup, proporcionant solucions efectives i millorant l'eficiència operativa.
   - Ajustar-se de manera dinàmica als canvis constants que pugui experimentar la plataforma *Low-Code* en desenvolupament, els quals poden impactar directament en el projecte.
5. **Verificació de la capacitat multiplataforma de PM2:**
   - Provar i verificar la capacitat de PM2 per ser implementat i utilitzat amb èxit en diferents plataformes, com ara sistemes operatius diversos.
6. **Sistema d'interacció Software-Hardware:**
   - Desenvolupar un sistema que permeti la interacció entre la part *Software*, que utilitza PM2, i el *Hardware*, assegurant l'execució correcta dels programes.
   - Verificar la cohesió del sistema mitjançant la correcta comunicació entre el *Software* i el *Hardware*.

3 METODOLOGIA I PLANIFICACIÓ

La metodologia i la planificació implementades en aquest projecte es van dissenyar amb cura per garantir eficiència, transparència i adaptabilitat. En adoptar la metodologia Kanban, Trello[4] va servir com a hub central per a una col·laboració sense costures. Kanban és un mètode visual de gestió de projectes que destaca el lliurament continu i l'optimització del flux de treball. Utilitza taulells visuals amb targetes que representen tasques, permetent als equips visualitzar la feina, limitar la feina en curs i millorar l'eficiència global. Cada aspecte del projecte, des de les històries d'usuari fins a les tasques associades, es va elaborar i organitzar minuciosament dins de les targetes de Trello, proporcionant un desglossament exhaustiu de les característiques.

El tauler visual de Kanban a Trello (vegeu figura 1) va jugar un paper fonamental en la supervisió del progrés de les tasques, fomentant la comunicació oberta i identificant possibles embussos. Aquest enfocament tenia com a objectiu obtenir una comprensió profunda dels requisits dels usuaris, assegurant que les característiques s'alineaven estretament amb els seus objectius premeditats.

Per establir una estructura clara i un calendari, es va desenvolupar amb detall un diagrama de Gantt, que delineava les fases estimades i les seves durades respectives. Actuant com a mapa del projecte, el diagrama de Gantt facilitava la gestió efectiva de recursos i oferia una guia visual del flux seqüencial de tasques.

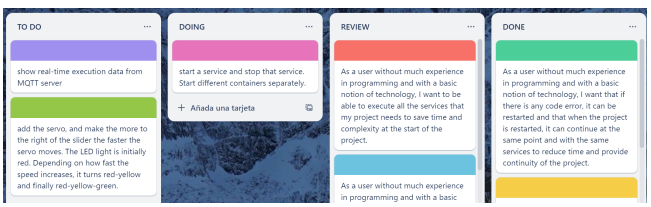

Fig. 1: Taulell Trello del projecte

El monitoratge continu i l'adaptació van ser elements de gran importància en l'estratègia global. La recerca diligent a través de diversos recursos en línia i les consultes amb diferents membres de la startup Protofy.xyz van jugar un paper crucial. Aquest procés iteratiu, enriquit amb idees recopilades mitjançant una recerca en línia extensa i debats col·laboratius, va permetre una resolució àgil de problemes

i va facilitar una presa de decisions ben informada, garantint l'èxit de les diferents parts del projecte.

4 BASE TEÒRICA

En el primer mes d'aquest projecte, es va dedicar a realitzar una formació teòrica profunda que va proporcionar una comprensió més completa de diversos conceptes i llenguatges relacionats. Aquesta etapa inicial, que va constar d'una part d'aprenentatge autònom i una altra impartida per membres de la startup, va ser fonamental per familiaritzar-se amb la plataforma i facilitar el desenvolupament del projecte.

Durant la formació inicial, es van cobrir temes clau com TypeScript, Express, LevelDB, React.js, a més d'analitzar diferents casos d'estudi.

4.1 TypeScript

TypeScript[5], un dels llenguatges escollits per la startup, requereix una formació específica a causa de les seves característiques úniques. Com a un llenguatge de programació basat en JavaScript amb tipat estàtic, ofereix un entorn de desenvolupament estructurat i rígid. Tot i que pot semblar restrictiu al principi, la rigidesa inicial de TypeScript és beneficiosa en l'entorn canviant d'una startup. En les primeres fases del projecte, ajuda a detectar errors quan els conjunts de classes i funcions no estan totalment definits.

A mesura que el projecte avança, el tipat estàtic de TypeScript es converteix en un avantatge, assegurant la seguretat del tipus, reduint els errors en temps d'execució i millorant la qualitat del codi. La seva previsibilitat fomenta la sostenibilitat, facilitant la col·laboració dels desenvolupadors. El suport de TypeScript per a la programació tant del costat del client com del servidor s'alinea amb els requisits de la plataforma, proporcionant una experiència de desenvolupament cohesionada.

Es pot dir, doncs, que les restriccions inicials de TypeScript es converteixen gradualment en flexibilitat, inversament lligades a les incerteses del projecte. Trobant l'equilibri entre l'estructura i l'adaptabilitat, es demostra adequat per a la naturalesa dinàmica dels projectes de startup.

4.2 Express.js

Express.js[6] és una llibreria àmpliament adoptada per a Node.js, que proporciona un conjunt robust de funcionalitats per facilitar el desenvolupament d'aplicacions web escalables i eficients. El seu disseny minimalista i la seva flexibilitat el converteixen en una opció popular per a la construcció del *back-end* d'aplicacions basades en el protocol *HTTP*.

La inclusió d'Express.js a la formació del projecte està justificada pel seu paper integral en la infraestructura del *back-end* de la plataforma de low-code. Aprofitant Express.js, la plataforma pot gestionar de manera eficient la definició de rutes, l'execució de *middleware* i altres aspectes essencials de la lògica del costat del servidor. Cal destacar que Express.js utilitza el protocol *HTTP* per a la comunicació, afegint una capa addicional de rendiment i compatibilitat amb els estàndards web.

4.3 LevelDB

LevelDB[7] és una comunitat i col·lecció de mòduls Node.js per a bases de dades transparents, oferint primitives sòlides per construir bases de dades potents. Aquest sistema de gestió de bases de dades clau-valor és àmpliament utilitzat en el desenvolupament web modern, destacant-se per la seva eficàcia en el tractament de dades no necessàriament estructurades.

La seva integració en el desenvolupament del projecte es fonamenta en el seu paper essencial com a sistema de gestió de bases de dades per a la plataforma de low-code. Amb un enfocament clau-valor, LevelDB facilita l'emmagatzematge de dades, alineant-se perfectament amb l'estructura dinàmica i evolutiva de la plataforma. Aquest enfocament possibilita un desenvolupament àgil, adaptant-se als models de dades evolutius propis d'un entorn de low-code. La capacitat de LevelDB per gestionar eficientment grans volums de dades i proporcionar una escalabilitat eficaç garanteix un rendiment òptim en el context d'aplicacions diverses i dinàmiques.

4.4 React.js

React.js[8], una llibreria de JavaScript per construir interfícies d'usuari, és un component fonamental en el desenvolupament web modern gràcies a la seva aproximació declarativa i basada en components. La seva inclusió a la formació del projecte està fonamentada pel seu paper central com a tecnologia *front-end* principal de la plataforma de low-code. L'arquitectura basada en components de React permet la creació d'elements d'interfície d'usuari reutilitzables, promocionant la modularitat del codi i millorant l'eficiència del desenvolupament dins del paradigma de low-code.

La implementació del DOM virtual de la llibreria garanteix una representació eficient, optimitzant el rendiment en aplicacions dinàmiques amb actualitzacions freqüents. El flux de dades unidireccional de React simplifica la gestió d'estats, proporcionant als desenvolupadors una aproximació estructurada i eficient per manipular interfícies d'usuari complexes. A més, React incorpora la funcionalitat de recàrrega en calent (hot reload), permetent als desenvolupadors veure actualitzacions instantànies a l'aplicació mentre realitzen canvis al codi. Aquesta capacitat accelera encara més el procés de desenvolupament, oferint una experiència fluida i eficient dins de la plataforma de low-code.

4.5 Next.js

La formació en Next.js[9] ha jugat un paper fonamental en l'enteniment i contribució al desenvolupament de projectes a la plataforma *Low-Code*. Next.js, com a marc de desenvolupament web basat en React, es converteix en una eina clau per a la creació d'interfícies d'usuari dinàmiques i eficients.

Amb una estructura modular i orientada a components, Next.js proporciona una abstracció potent que simplifica la complexitat del desenvolupament del costat del client. A través d'aquesta formació, s'adquireixen habilitats per gestionar rutes, controlar l'estat global de l'aplicació i optimitzar la càrrega de pàgines.

Next.js ofereix dues modalitats d'aplicació, que són essencials de comprendre: el Renderitzat del Costat del Ser-

vidor (SSR) i l'Aplicació d'una sola pàgina (SPA). Amb el SSR, les pàgines es renderitzen al servidor, mentre que amb l'SPA, la renderització es realitza al costat del client. Aquesta dualitat permet als desenvolupadors triar la millor estratègia segons les necessitats específiques de cada pàgina o aplicació.

4.6 Casos d'estudi

A continuació, es presenten dos casos d'estudi rellevants que s'han abordat en el marc del projecte. Aquests casos d'estudi representen reptes específics identificats en els projectes desenvolupats per l'empresa i il·lustren les solucions implementades per abordar-los.

4.6.1 CAS D'ESTUDI 1: Realització de Script per arrancar serveis específics amb Docker

Problema: Per poder detectar la problemàtica inicial, es va observar diferents projectes que l'empresa havia desenvolupat i es van realitzar una sèrie d'històries d'usuari[10] amb els conflictes comuns. La història d'usuari que resumeix els problemes és la següent: 'Jo com a usuari sense molta experiència en programació i amb una noció bàsica de la tecnologia, vull poder arrancar un o diversos serveis específics sense haver d'obrir totes les disponibles, per estalviar temps i permetre un correcte funcionament del projecte.'

Possible solució: La solució que es va pensar i dissenyar va ser la de desenvolupar un script que li passaven per paràmetre el nom de les diferents imatges que es volien executar, juntament amb el port que es volien executar. Si no s'especifica un port, s'assigna un lògicament diferent per a cada imatge. En acabar de crear les imatges, en aquest fitxer es demana si es vol interrompre l'execució d'aquestes i, si és el cas, si es volen eliminar, tot això de forma individualitzada per a cada imatge.

Resultats: El resultat és el que es tenia com a objectiu, s'aconsegueix executar, parar i eliminar les imatges que es desitgen sense haver de fer-ho amb serveis no desitjats. L'arxiu resultant es mostra en aquest[11] repositori de Github.

4.6.2 CAS D'ESTUDI 2: Realització de Script per mantenir arrancat un servei amb PM2

Problema: En aquest següent cas d'estudi es tenia una problemàtica diferent de l'anterior. Aquí se situa una història d'usuari en un projecte que desenvolupa un company de la startup el que l'interessava que si hi hagués algun error en el sistema, després de reiniciar-se el sistema, es vol que pugui continuar el servei des del mateix punt d'abans que passi l'errada del sistema.

Possible solució: La solució implementada en aquest cas és utilitzar el gestor de processos PM2. PM2 és un administrador de processos per a aplicacions Node.js que s'executen en un entorn de servidor. Proporciona diverses característiques útils per a gestionar aplicacions, com la capacitat de mantenir-les en execució contínuament, administrar múltiples instàncies de l'aplicació, equilibrar la càrrega, i gestionar el reinici automàtic en cas de fallades. PM2 també permet gestionar l'estat de les aplicacions, veure registres de sortida i monitorar l'ús de recursos com la CPU i la memòria. Aquest administrador s'empra en aquest cas per mantenir en constant execució un fitxer, guardant el seu estat d'execució amb una de les funcions de PM2, per així poder continuar executant-se després que el sistema es reinicia.

Resultats: El resultat és el desitjat; el fitxer continua executant-se inclús quan s'ha reiniciat el sistema, fent que estigui a prova de qualsevol error en el sistema. El projecte resultant es mostra en aquest[12] repositori de Github.

5 PROPOSTA SOFTWARE

Aquesta secció aborda tres àmbits clau de la proposta *Software*: l'estudi de mercat, la proposta inicial i la proposta final. L'estudi de mercat es basa en històries d'usuari que analitzen els reptes presents en diversos projectes de la startup. Aquest enfocament proporciona una visió detallada dels problemes pràctics, identificant punts dèbils comuns i requisits específics dels usuaris.

La proposta inicial es va gestar després d'identificar els desafiaments comuns als diferents projectes de la startup. Amb una estreta relació amb el primer cas d'estudi, aquesta proposta va ser plantejada com un punt de partida, reconeixent que la solució més senzilla no sempre és la més eficaç per abordar reptes complexos.

La proposta final destaca la connexió amb el segon cas d'estudi, especialment en l'elecció de PM2 com a gestor de processos preferit enfront de Docker. Aquesta elecció es basa en les característiques específiques de PM2 que s'ajusten millor a les necessitats del projecte, com la capacitat de mantenir en execució contínua un fitxer i gestionar eficientment els reinicis automàtics.

5.1 Estudi de mercat

En realitzar un estudi de mercat per aquest projecte, es van utilitzar històries d'usuari per analitzar els reptes presents en diversos projectes de la startup. Aquest enfocament va revelar punts dèbils comuns i requisits específics dels usuaris, proporcionant una visió detallada dels problemes pràctics. La correlació d'aquestes històries va identificar reptes recurrents, obrint camí a possibles solucions. Aquesta exploració centrada en l'usuari té com a objectiu comprendre de manera exhaustiva les necessitats dels usuaris i les tendències del mercat, sent la base per a un enfocament informat i estratègic en aquest treball.

5.1.1 Històries d'usuari

- Jo, com a usuari amb poca experiència en programació i una noció bàsica de la tecnologia, vull que, en crear un nou projecte, s'obrin tots els fitxers que necessito per estalviar temps i complexitat en l'inici del projecte.

- Jo, com a usuari amb poca experiència en programació i una noció bàsica de la tecnologia, vull poder aturar només el contingut que m'interessi i no tot el projecte, per estalviar temps i permetre un funcionament correcte del projecte.

- Jo, com a usuari amb poca experiència en programació i una noció bàsica de la tecnologia, vull que si es

produeix un error d'alimentació o error de codi, en reiniciar el projecte, es pugui continuar des del mateix punt, per reduir temps i aportar continuïtat al projecte.

5.2 Proposta inicial

Després d'identificar la problemàtica comuna en diversos projectes de la startup, la proposta inicial del *Software* es va concebre amb una estreta relació amb el primer cas d'estudi. Aquesta elecció va ser plantejada com una proposta inicial, ja que es va considerar com el mètode més simple i directe per abordar els desafiaments presentats pel cas d'estudi. No obstant això, és important subratllar que, malgrat la seva aparent simplicitat, la solució més senzilla no sempre resulta ser la més eficaç o òptima per abordar els reptes complexos que es plantegen en els casos d'estudi. Aquesta primera proposta, tot i servir com a punt de partida, va ser objecte d'una revisió crítica per garantir que les solucions futures fossin més sòlides i s'adeqüessin millor a les necessitats específiques dels usuaris.

5.3 Proposta final

En el context del desenvolupament del projecte, s'ha considerat la relació entre la proposta final del *Software* i el segon cas d'estudi. Aquesta connexió és fonamental per comprendre la decisió d'optar per PM2 com a solució preferida enfront de Docker. Tot i que Docker és àmpliament utilitzat i reconegut per a la gestió de contenidors, la proposta final ha dirigit la seva preferència cap a PM2, i això es deu a diversos factors.

PM2, com a gestor de processos de propòsit general, ofereix característiques específiques que s'alineen de manera més precisa amb les necessitats del segon cas d'estudi. La capacitat de mantenir en execució contínua un fitxer, guardar l'estat d'execució amb una funció específica de PM2 i gestionar de manera eficient els reinicis automàtics en cas de fallades, converteixen PM2 en una opció més adequada per a les especificitats del projecte.

Malgrat que Docker és una eina versàtil i amplament usada en diversos contextos, la proposta final ha optat per PM2 per optimitzar el rendiment i assegurar la continuïtat dels serveis en el context del segon cas d'estudi. Aquesta elecció destaca la importància d'avaluar detingudament les necessitats i els requisits específics de cada cas per prendre decisions informades sobre les tecnologies i eines a emprar en el desenvolupament del *Software*.

Pel que fa a la interfície d'usuari, es va prendre com a font d'inspiració un projecte[13] relacionat amb el gestor de recursos PM2, tot i que l'actual projecte està totalment personalitzat i adaptat per a les necessitats i requisits específics del projecte, assegurant així la singularitat del desenvolupament.

6 Implementació Software

En aquesta secció dedicada a la implementació del *Software*, s'abordaran diversos aspectes clau que defineixen la funcionalitat i l'arquitectura del sistema. Com a fonament del desenvolupament, es presenta un esquema lògic que representa la relació i la interacció entre els components clau del sistema. Aquesta representació visual proporcionarà una visió global de com les diferents parts del *Software* s'integren per assolir els objectius del projecte.

Un aspecte crític de la implementació és l'ús del gestor de processos PM2. Aquesta eina és essencial per mantenir en execució contínua els diferents components de l'aplicació, assegurant un rendiment estable i gestionant eficientment els reinicis automàtics en cas de fallades. En l'apartat 6.2 es descriuran les característiques i funcionalitats específiques de PM2 que contribueixen al bon funcionament del sistema. En l'arquitectura lògica que es mostra més endavant (vegeu figura 2), es podrà observar amb més claredat com és el funcionament de PM2 dins la plataforma, i com actua amb els diferents programes que consta aquesta.

La connexió *MQTT* (Message Queuing Telemetry Transport) és un element central per a la comunicació eficient entre els diversos dispositius i components del sistema. Es detallarà com s'estableix i gestiona aquesta connexió, destacant la seva importància per a la transmissió de dades en temps real i la coherència del sistema.

S'introduirà la plataforma *Low-Code* anomenada "Protofy", que ha estat fonamental en el desenvolupament del projecte. Aquesta plataforma ofereix eines i funcionalitats específiques que simplifiquen la creació i gestió de codi, accelerant així el procés de desenvolupament. Es descriuran les característiques destacades de Protofy i com s'integra amb els altres components del sistema.

Funcionament de la interfície web

S'utilitzen *endpoints*[1] per mostrar els serveis disponibles i dur a terme diverses funcions amb PM2. S'empra 'localhost:8080/' pel fet que l'execució es realitza localment al port 8080. La comunicació amb la interfície es du a terme mitjançant peticions *HTTP REQUEST* de tipus *POST* i *GET*, per actualitzar i llegir dades respectivament.

A continuació es mostra una llista de les diferents peticions que es duen a terme:

- *GET* http://localhost:8080/api/v1/services → Serveix per obtenir els serveis inicials que es mostren a la interfície.
- *GET* http://localhost:8080/api/v1/PM2Services/describe/:id → S'utilitza per mostrar les dades de monitoratge inicial per cada servei segons el seu id.
- *GET* http://localhost:8080/api/v1/PM2Services/start/:id → Inicialitza un servei segons el seu id.
- *GET* http://localhost:8080/api/v1/PM2Services/restart/:id → Reinicialitza un servei segons el seu id.
- *GET* http://localhost:8080/api/v1/PM2Services/stop/:id → Permet parar un servei segons el seu id.
- *GET* http://localhost:8080/api/v1/PM2Services/delete/:id → Elimina un servei segons el seu id.
- *POST* http://localhost:8080/api/v1/services/ + service.id, service → Actualitza els serveis quan es fa algun canvi.

[1] Punt d'accés específic en un servei *HTTP* que pot ser utilitzat per efectuar operacions o obtenir informació

6.1 Arquitectura lògica plataforma Low-code

En la següent figura es mostra un exemple lògic de com està estructurada la plataforma low-code. En la part del servidor està ubicada la plataforma, on el gestor de processos PM2 és qui executa tots els programes que componen la plataforma, incloent-hi el reverse proxy. Aquest últim actua com a intermediari depenent de les peticions realitzades pel client, subministrant el servei.

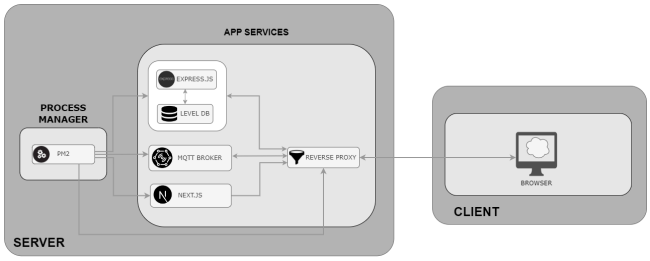

Fig. 2: Diagrama de l'arquitectura lògica de la plataforma

6.2 Gestor de processos PM2

PM2 (Process Manager 2)[14] destaca com un gestor de processos robust i versàtil dissenyat originàriament per a aplicacions Node.js, proporcionant funcionalitats essencials per a un rendiment òptim i fiabilitat en entorns de producció. Amb el temps, PM2 ha evolucionat per donar suport a una gamma més àmplia d'aplicacions i llenguatges de programació, convertint-se en un gestor de processos més de propòsit general.

6.2.1 Funcionalitat i característiques

PM2, com a gestor de processos, sobresurt no només per la gestió eficaç dels processos, sinó també com a gestor de dimonis. PM2 opera com un dimoni, garantint la disponibilitat i la continuïtat dels serveis de l'aplicació en tot moment. La seva funció de reinici automàtic millora la solidesa del sistema, assegurant que les aplicacions es reiniciïn automàticament en cas de fallades o bloquejos.

Anant més enllà de la simple gestió de processos, PM2 ofereix suport integrat per al balanceig de càrrega, distribuint de manera eficient el trànsit entre diverses instàncies d'una aplicació per optimitzar l'ús dels recursos i millorar el rendiment global del sistema.

Amb capacitats completes de registre i monitoratge, els desenvolupadors poden seguir el comportament de l'aplicació, diagnosticar problemes de manera efectiva i assegurar una experiència d'usuari sense interrupcions. PM2 simplifica els processos de desplegament, permetent als desenvolupadors llançar aplicacions amb facilitat mitjançant comandes senzilles.

A més, cal destacar que PM2 és multiplataforma, oferint la seva funcionalitat en diferents sistemes operatius, la qual cosa contribueix a la seva versatilitat i adaptabilitat a entorns diversos.

6.2.2 Comparació amb altres gestors de processos

PM2 destaca en comparació amb els gestors de processos tradicionals per la seva simplicitat, facilitat d'ús i funcionalitats específicament adaptades per a aplicacions. Superant alternatives en termes de robustesa, proporciona una interfície amigable i funcionalitats extenses. PM2 no està limitat exclusivament a aplicacions Node.js, sinó que també ofereix suport a una àmplia gamma d'aplicacions i llenguatges de programació, augmentant la seva versatilitat.

6.2.3 Aplicació en aquest projecte

En aquest projecte, la implicació de PM2 ha estat crucial per abordar les problemàtiques associades a l'execució i gestió dels serveis. Abans de la seva implementació, es presentaven dificultats com la incapacitat d'executar serveis de manera individual o en grups específics. A més, els serveis en execució es veien afectats quan el sistema es reiniciava. PM2 ha resolt aquestes qüestions amb eficàcia. La seva utilització en el projecte es materialitza a través d'una interfície *front-end* (vegeu figura 3) que ofereix un conjunt de serveis predefinits. En iniciar-los, s'utilitza la comanda *pm2.start*, i en aturar-los, la comanda *pm2.stop*. Aquesta interfície disposa de característiques com un botó per desar l'estat dels serveis, permetent que, si l'estat es guarda mentre un procés s'està executant, el servei continuï després de refrescar la pàgina. En cas contrari, el procés romandrà aturat. Una funcionalitat destacada és la capacitat de reiniciar el procés mitjançant la comanda *pm2.restart*, la qual pot ser útil en certes situacions. Per afegir nous processos, només cal clicar a la icona 'més' i assignar un nom al procés desitjat, el qual s'afegeix en estat aturat per defecte. Dins d'aquesta interfície gràfica, es mostra en temps real el registre d'execució dels diferents serveis a la part superior de la pantalla. Aquesta visualització detallada dels registres, que s'actualitza cada minut, proporciona informació sobre l'execució dels processos. Per obtenir tota la informació d'un procés, PM2 ofereix la funció *pm2.describe*, la qual serà important en el següent apartat com a part de la justificació de la connexió *MQTT*. Finalment, hi ha dues opcions disponibles que es mostren en clicar la icona el·lipsis vertical: guardar l'estat de tots els serveis mitjançant la icona de guardar tots els serveis, i la d'esborrar un servei mitjançant la icona d'esborrar. L'opció d'esborrar s'executa amb la funció *pm2.delete*.

6.3 Connexió MQTT

En aquest projecte, la connexió *MQTT*[15](Message Queuing Telemetry Transport) juga un paper crucial en facilitar l'intercanvi de dades en temps real i la comunicació entre diversos components. *MQTT* s'utilitza en dues situacions diferents per millorar la funcionalitat global del sistema.

En primer lloc, *MQTT* s'empra per establir una connexió en temps real per a la informació de monitoratge. La comunicació en temps real és possible gràcies a l'execució de MQTT sobre *websockets*, fent possible una comunicació *full-duplex* (bidireccional) i facilitant la comunicació real en aplicacions basades en el protocol HTTP. S'aprofita la funcionalitat PM2 (Process Manager 2), concretament *pm2.describe*, per obtenir detalls exhaustius sobre els processos en execució. Aquesta informació inclou mètriques crítiques com l'ús de la CPU i la memòria. Mitjançant la integració de *MQTT*, aquestes dades de monitoratge es transmeten de manera transparent en temps real, permetent als

usuaris observar i analitzar les mètriques de rendiment del sistema mentre es produeixen.

A més, *MQTT* s'usa per gestionar la visualització dels registres dels processos. A intervals regulars d'un minut, el sistema fa servir *MQTT* per difondre els registres generats pels diferents processos. Això garanteix que els usuaris tinguin accés a informació actualitzada i detallada sobre l'execució dels diferents processos. La connexió *MQTT* facilita la difusió eficient i oportuna d'aquests registres, proporcionant als usuaris una visió completa i dinàmica del comportament i l'estat de cada procés.

6.4 Plataforma Protofy

Protofy és una plataforma CMS (Content Management System) i *framework Low-Code* potenciada per la intel·ligència artificial (IA) que ofereix una àmplia gamma de funcionalitats per al desenvolupament d'aplicacions web i mòbils, així com sistemes IoT.

Característiques clau de Protofy:

- Potenciat per la IA: Protofy fa ús de les tecnologies d'IA per millorar el procés de desenvolupament i proporcionar funcionalitats intel·ligents.
- Plataforma *Low-Code*: Protofy ofereix una aproximació *Low-Code*, permetent als desenvolupadors construir aplicacions de manera ràpida i eficient mitjançant editors visuals, operacions CRUD automàtiques (Crear, Llegir, Actualitzar, Esborrar) i panells de control.
- CMS i *framework*: Protofy combina les funcionalitats d'un Sistema de Gestió de Continguts (CMS) i un *framework*, oferint una solució integral per a la construcció de sistemes digitals.
- Pila Integrada: Protofy està construït sobre tecnologies populars com React, NextJS, Expo, Tamagui i EspHome, proporcionant una pila robusta i integrada per al desenvolupament.
- Extensibilitat: Protofy ofereix un sistema altament extensible, permetent als desenvolupadors ampliar i modificar el *framework* i el sistema de panells de control subjacents.
- Sistema d'Objectes amb Validació: Protofy inclou un sistema d'objectes amb capacitats de validació incorporades, facilitant el maneig de dades i l'autogeneració de formularis basada en l'entitat.
- Assistent Integrat ChatGPT: Protofy incorpora un assistent ChatGPT que suporta la transferència automàtica de context, millorant l'experiència de l'usuari.

6.5 Resultat Software

A continuació es mostra la interfície d'usuari que ha acabat sent la solució de la part Software, la qual consta de les diferents funcionalitats del gestor de recursos PM2 integrades en la plataforma low-code Protofy.

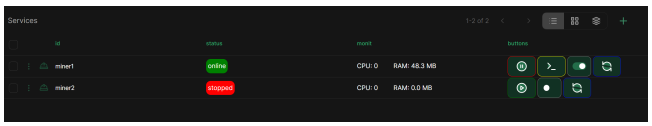

Fig. 3: Gestor de recursos en la plataforma Protofy

7 Proposta Hardware

En la següent secció es parlarà sobre l'estudi de mercat realitzat per donar suport a la proposta Hardware. Es menciona que aquest estudi es va dur a terme tenint en compte les necessitats que el projecte requeria. Així mateix, s'explorarà la proposta inicial, que, a causa del seu enfocament ambiciós i optimista, no va poder concretar-se i va evolucionar cap a la proposta final, que també es tractarà a continuació.

7.1 Estudi de mercat

L'estudi actual té com a finalitat avaluar diverses controladores al mercat per determinar la més idònia en un projecte de gestió de recursos. Aquest consisteix a executar un programa amb PM2 que activa tres leds i un servomotor en rebre el senyal del slider a través d'una connexió *MQTT* des de la plataforma *Low-Code*.

7.1.1 Requisits del projecte:

- Capacitat per executar programes de qualsevol extensió i format.
- Aptitud per executar el gestor de recursos en qualsevol plataforma.
- Connectivitat amb diferents components mitjançant *MQTT*.
- Control de LED i un servomotor.

7.1.2 Comparació de plaques

Raspberry Pi 4 Model B

1. **Avantatges:**
   - Alta potència de càlcul adequada per a una àmplia gamma d'aplicacions.
   - Gran suport comunitari i abundant documentació.
   - Compatibilitat amb Linux, oferint una gran flexibilitat en el desenvolupament de programari.
   - Suport per a múltiples llenguatges de programació.
   - Inclou pins GPIO per a la interconnexió amb components *hardware*.
   - Ampliament disponible en el mercat.
2. **Desavantatges:**
   - Preu superior en comparació amb altres microcontroladors.
   - Major consum d'energia respecte a microcontroladors més simples.

Odroid-XU4

1. **Avantatges:**
    - Rendiment superior en termes de CPU i RAM respecte a altres microcontroladors, incloent el Raspberry Pi.
    - Incorpora ports USB 3.0 per a una transferència de dades més ràpida.
    - Suport per a mòduls eMMC, oferint una opció d'emmagatzematge més ràpida i fiable que les targetes microSD.
2. **Desavantatges:**
    - Comunitat d'usuaris i suport més limitats.
    - Cost més elevat comparat amb altres microcontroladors.
    - Consum d'energia més alt.

Libre Computer Board AML-S905X-CC (Le Potato)

1. **Avantatges:**
    - Processador quad-core potent.
    - Capacitat de reproducció de vídeo en 4K.
    - Inclou pins GPIO per a la interconnexió amb components *hardware*.
    - Compatible amb Linux, proporcionant flexibilitat en el desenvolupament de programari.
    - Preu competitiu respecte al Raspberry Pi, oferint capacitats similars.
2. **Desavantatges:**
    - Menor suport comunitari en comparació amb el Raspberry Pi.
    - Menys accessoris disponibles en el mercat.
    - El consum d'energia pot ser superior al de microcontroladors més senzills com l'Arduino Uno.

Consideracions generals

Si la potència de càlcul, el suport comunitari extens i una àmplia gamma d'aplicacions són essencials, Raspberry Pi 4 Model B és una opció sòlida. Per a aplicacions integrades que requereixen més rendiment i connectivitat, com ara projectes més intensius en còmput o aplicacions que es beneficien de ports USB 3.0 i suport eMMC, l'Odroid-XU4 emergeix com una alternativa potent. Libre Computer Board AML-S905X-CC (Le Potato) se situa en un punt intermedi, oferint una potència de càlcul substancial a un preu més competitiu, tot i que amb una mica menys de suport comunitari que Raspberry Pi. Tenint en compte la necessitat d'executar una plataforma *Low-Code* des del dispositiu, la qual cosa exigeix una capacitat de còmput considerable, així com la necessitat de gestionar eficientment l'arxiu per als components *Hardware* i tenint en compte la disponibilitat del dispositiu a la startup, s'ha optat per utilitzar el Raspberry Pi 4 Model B com a dispositiu per a la part *Hardware* del projecte.

7.2 Proposta inicial

En les fases inicials del projecte, es va plantejar una proposta ambiciosa que buscava implementar un sistema de reconeixement facial i veu utilitzant una càmera i un altaveu connectats a una Raspberry Pi. L'objectiu era desenvolupar un programa capaç d'identificar una persona mitjançant el seu rostre i pronunciar-ne el nom a través dels altaveus, proporcionant una experiència d'interacció única i personalitzada.

No obstant això, al llarg del desenvolupament, es va fer evident que les estimacions inicials del temps de planificació eren més optimistes del que haurien de ser. Aquesta desviació es va atribuir principalment als nombrosos canvis i actualitzacions continuades a la plataforma de low-code que estava en procés de creació. Aquests canvis, tot i ser beneficiosos per millorar la plataforma, van generar conflictes constants amb la part de programari del projecte.

La necessitat de readaptar constantment la part de programari del projecte a causa de les actualitzacions de la plataforma va provocar un retard significatiu en el cronograma previst. Aquesta adaptació contínua als canvis es va convertir en un desafiament, ja que la base del projecte estava íntimament vinculada a la plataforma en evolució.

7.3 Proposta final

Després de les diverses adaptacions necessàries durant el desenvolupament del projecte, la proposta final se centra en una implementació més simplificada i específica. Consisteix en un *front-end* que presenta un control lliscant (slider) mitjançant el qual l'usuari pot moure el servo de 0 a 180 graus, depenent de si es llisca cap a l'esquerra o la dreta, respectivament. A més, cada vegada que es mou el slider, s'activen de dreta a esquerra o d'esquerra a dreta els tres leds connectats (igual que el servo) a una Raspberry Pi, segons si el servo es mou cap a 0 o cap a 180 graus. Aquesta elecció respon a la necessitat de reduir la complexitat del projecte i simplificar-ne els components, com es va explicar anteriorment respecte a les demores experimentades. Tot i això, els objectius principals d'aquesta part del projecte es mantenen centrats en l'execució de PM2 des de plataformes diferents i en la capacitat de gestionar arxius amb extensions diferents de '.ts' (extensió de TypeScript, la qual s'utilitzava exclusivament a la plataforma).

8 Implementació Hardware

Com s'ha esmentat amb anterioritat, aquesta secció del projecte ha vist reduïda la seva envergadura a causa del temps addicional requerit per la fase anterior, motivada principalment pel constant ajust continu en el desenvolupament de la plataforma *Low-Code*, el qual ha afectat el meu projecte en diverses ocasions. En la línia de la proposta final de *Hardware*, el principal objectiu és demostrar la compatibilitat multiplataforma de PM2 i la capacitat d'execució de fitxers amb extensions diverses.

La funcionalitat final consisteix en la implementació d'un *front-end*, gestionat per la plataforma *Low-Code*, on es presenta un slider. En moure's cap a la dreta, el valor dels angles que es transmet cap al servo augmenta fins al límit de 180 graus, i els tres leds (vermell, groc i verd) s'encenen en

l'ordre mencionat anteriorment. En el cas de moure's cap a l'esquerra, passa la inversa del cas descrit. Aquest resultat és possible gràcies a l'execució de dos fitxers diferents. Un d'aquests fitxers conté la part dels pins i la lògica de la Raspberry Pi, implementat per comoditat i per demostrar que es poden córrer més d'un fitxer amb el gestor de processos PM2, com un fitxer Python. L'altre fitxer se centra en la visualització de les dades del *front-end*, realitzat mitjançant la plataforma *Low-Code* i, per tant, desenvolupat amb React.js.

De manera similar a la secció de *Software*, aquesta part també utilitza la comunicació *MQTT*. L'arxiu TypeScript és responsable de publicar les dades, en aquest cas, els valors corresponents a la barra lliscant, que varien de 0 a 180. Aquests valors representen els 180 graus que admet com a màxim el servomotor; com més a la dreta, més gran és el valor. Segons aquests valors, l'arxiu Python, que s'ha subscrit a la connexió *MQTT*, controla en quin sentit s'activen els leds. La forma que s'ha implementat perquè s'executi el moviment del servomotor i els leds simultàniament ha estat mitjançant l'ús de threading. El seu funcionament consisteix a cridar dos fils d'execució per a les funcions que actuen sobre el servo i els leds, i després realitzar un join per esperar que ambdós fils finalitzin l'execució de les seves funcions. Les seccions següents proporcionaran una llista dels diferents components (vegeu taula 1) utilitzats per fer possible aquesta secció del projecte i un esquema lògic (vegeu figura 4) per oferir una visió més gràfica del sistema. En la secció A.5 del apèndix s'ha adjuntat el resultat del montatge dels diferents components (vegeu figura 10).

8.1 Components

A continuació es mostra una taula dels diferents components que s'han utilitzat en aquesta part del projecte i que han estat facilitats per la startup Protofy.xyz:

Identificador	Component
C1	LED Vermell
C2	LED Groc
C3	LED Verd
C4	3 x Resistències 65Ω [2]
C5	Cables Jumper Mascle-Femella (M-F)
C6	Cables Jumper Mascle-Mascle (M-M)
C7	Servomotor DS3225
C8	Font d'alimentació AC/DC IRM-30-5ST
C9	Raspberry Pi 4 model B

TAULA 1: IDENTIFICACIÓ DE COMPONENTS

8.2 Esquema lògic

El diagrama lògic següent il·lustra els components, tant *Hardware* com de *Software*, implicats en aquesta secció del projecte. D'una banda, trobem la plataforma i la seva interfície gràfica, on el gestor de processos PM2 s'encarrega de gestionar les diverses aplicacions que integren la plataforma low-code, incloent-hi l'execució d'un arxiu Python en el Raspberry Pi 4. La interacció entre aquest arxiu Python i la interfície gràfica es realitza mitjançant una connexió MQTT. A través d'aquesta, l'arxiu Python se subscriu a un tema (topic) específic, mentre que la interfície gràfica, fent ús d'un control lliscant (slider), publica valors basats en la posició en què es desplaça el control. Respecte a l'aspecte físic del projecte, inclou un servomotor i tres LEDs que s'activen de manera sincronitzada en resposta als senyals rebuts.

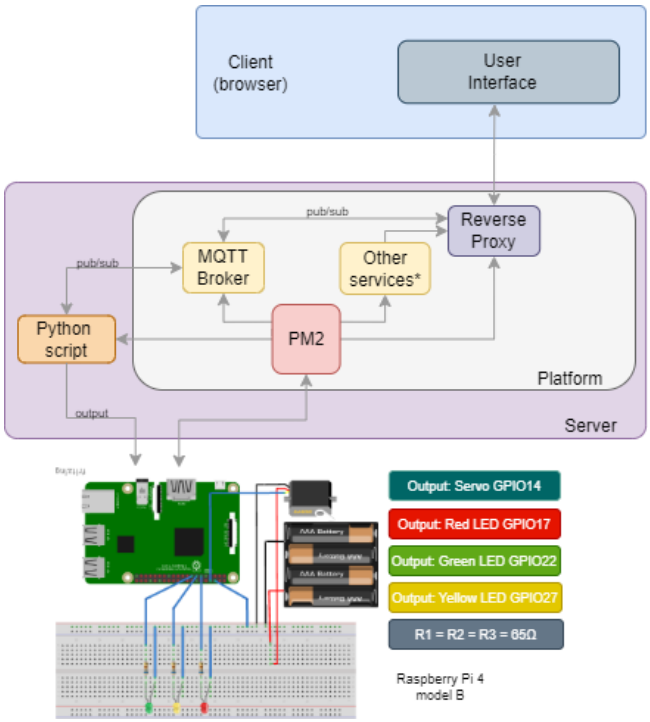

Fig. 4: Esquema lògic de l'arquitectura *Hardware*

***Other services:**

- LevelDB, Express.js i Next.js

9 CONCLUSIONS

9.1 Conclusions del treball

La integració exitosa de les funcionalitats de PM2 dins d'una plataforma low-code ha aportat un valor considerable a l'empresa. Aquest èxit prové d'un detallat procés d'investigació i desenvolupament que ha permès complir amb tots els objectius preestablerts. La proposta inicial s'ha validat eficaçment a través de la implementació de solucions tant en *software* com en *hardware*, demostrant la viabilitat del projecte. La utilització de PM2 com a gestor de processos ha millorat significativament la gestió de recursos, oferint una solució flexible i adaptable apta per a diferents entorns. Aquesta experiència ha recalcat la importància de ser adaptable i flexible, especialment quan es treballa amb tecnologies en evolució, i ha subratllat la necessitat d'ajustar-se ràpidament als canvis per superar els desafiaments. Com a resultat, el projecte no només ha enriquit el coneixement i les habilitats en gestió de recursos i integració de plataformes, sinó que també ha establert un sòlid precedent per a futurs projectes, destacant el valor de l'execució meticulosa i l'adaptabilitat en entorns dinàmics.

[2] El valor de la resistència és definit per la fórmula $R = \frac{3.3V - 2.0V}{0.02A} = 65\Omega$, on $0.02A$ és la intensitat nominal del LED, $2.0V$ és la caiguda de tensió directa del LED i $3.3V$ és la tensió de sortida de la Raspberry Pi 4.

9.2 Conclusions personals

En l'àmbit personal, aquest projecte ha representat un repte estimulant i formatiu. La complexitat de treballar amb una plataforma en constant canvi ha millorat la meva capacitat per adaptar-me a situacions imprevistes i aprendre sobre la marxa. La resolució de problemes continus ha reforçat la meva habilitat per abordar obstacles de manera efectiva. A més, l'experiència d'integrar *Hardware* i *Software* ha proporcionat una comprensió més profunda de la interconnexió entre ambdós i com aconseguir una sinergia eficient. La capacitat per superar els contratemps i assolir els objectius marcats ha reforçat la meva confiança per abordar projectes futurs de manera proactiva. En última instància, aquest projecte ha estat més que una tasca tècnica; ha estat un viatge educatiu i de desenvolupament personal que ha contribuït significativament al meu creixement professional i habilitats tècniques.

9.3 Propostes de millora a futur

Com a continuació natural d'aquest projecte, es podrien realitzar diverses millores significatives en diferents àrees. Per començar, seria beneficiós centrar-se a millorar l'aspecte visual de la interfície d'usuari, optimitzant les icones i ajustant algunes funcionalitats per proporcionar una experiència més agradable, ja que a causa del temps que es disposava s'-ha optat per obtenir una versió més funcional que estètica. Dins del desenvolupament de programari, una àrea crítica de millora consistiria a revisar i optimitzar el codi font per reduir els temps de resposta i millorar l'eficiència general del sistema. Quant a la part *Hardware*, s'obririen un seguit d'oportunitats de millora. Una proposta inicial podria ser implementar la proposta *Hardware* original, amb reptes més complexos per posar a prova les capacitats del sistema. Altres possibles millores podrien incloure l'exploració de nous protocols de comunicació, la incorporació de funcionalitats addicionals segons les necessitats dels usuaris i la integració de tecnologies emergents.

AGRAÏMENTS

En primer lloc, vull expressar el meu agraïment a Protofy.xyz per proporcionar-me l'entorn propici per al desenvolupament d'aquest projecte, oferint-me totes les eines necessàries. Vull dedicar un agraïment especial al meu tutor a l'empresa, Lluís Rovira, pel seu continu suport, ajudant-me a resoldre els dubtes que sorgien i guiant-me quan m'adonava que estava perdut. També vull agrair al meu tutor a la universitat, Remo Suppi, per la seva disponibilitat constant i la seva habilitat per respondre a les meves inquietuds. Finalment, un agraïment sincer a la meva família, que ha estat un suport fonamental a la meva vida i ha contribuït al meu creixement personal.

REFERÈNCIES

[1] Repositori del projecte amb la part *Software*: [Online] https://github.com/RogerOrRobert/PM2-interface-SW

[2] Repositori del projecte amb la part Hardware:[Online] https://github.com/RogerOrRobert/PM2-interface-HW

[3] Repositori de la plataforma *Low-Code* Protofy: [Online] https://github.com/protofy-xyz/protofy

[4] Trello inc.. Trello. [Online] https://trello.com/

[5] Article amb raons per utilitzar TypeScript:[Online] https://brandontle.com/writing/why-typescript/

[6] Documentació de les funcionalitats d'Express.js: [Online] https://expressjs.com/

[7] Informació de la base de dades levelDB: [Online] https://leveljs.org/

[8] Documentació React.js: [Online] https://react.dev/learn

[9] Tutorial inicial a next.js: [Online] https://nextjs.org/learn/dashboard-app/getting-started

[10] - Llibre utilitzat per entendre les històries d'usuari: [Llibre] Mike Cohn. *User Stories Applied: For Agile* Software *Development*. Addison-Wesley, 2004.

[11] Repositori cas d'estudi script serveis Docker: [Online] https://github.com/RogerOrRobert/Docker-project

[12] Repositori cas d'estudi mantenir serveis arrancats: [Online] https://github.com/RogerOrRobert/Keep-Alive-Project

[13] Projecte gestor de recursos: [Online] https://javascript.plainenglish.io/how-to-create-a-simple-web-ui-to-manage-node

[14] Tutorial sobre el gestor de recursos PM2: [Online] https://PM2.keymetrics.io/docs/usage/quick-start/

[15] Tutorial ús de la connexió *MQTT*: [Online] https://www.hivemq.com/mqtt/

APÈNDIX

A.1 Diagrama de Gantt

En aquesta secció de l'apèndix es presenten dos diagrames de Gantt, un representa el pla inicial i l'altre reflecteix els ajustos realitzats a conseqüència dels retards:

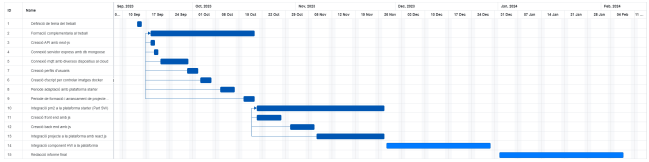

Fig. 5: Diagrama de Gantt inicial

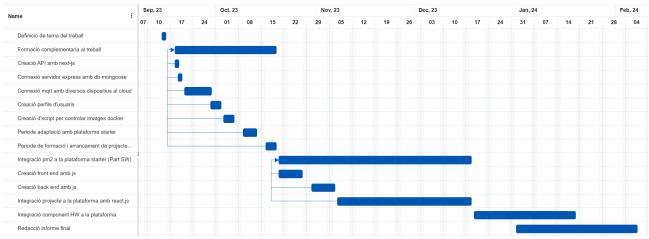

Fig. 6: Diagrama de Gantt final

A.2 UI part Software

En aquest apartat es pot observar el *front-end* del gestor de serveis a la plataforma *Low-Code*

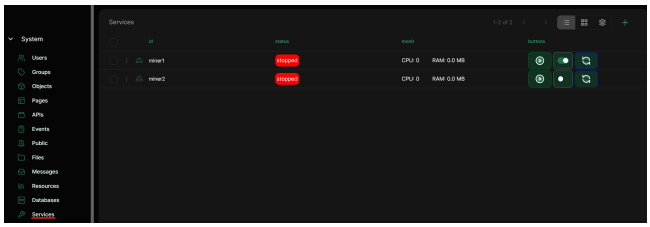

Fig. 7: Interfície *Software* completa

A.3 UI part Hardware

Aquesta secció mostra el *front-end* de la part *Hardware* del projecte.

Fig. 8: Interfície Hardware completa

A.4 Esquemàtic part Hardware

Aquí es pot observar un esquemàtic realitzat amb el programari fritzing de les diferents connexions i components de la part Hardware del treball.

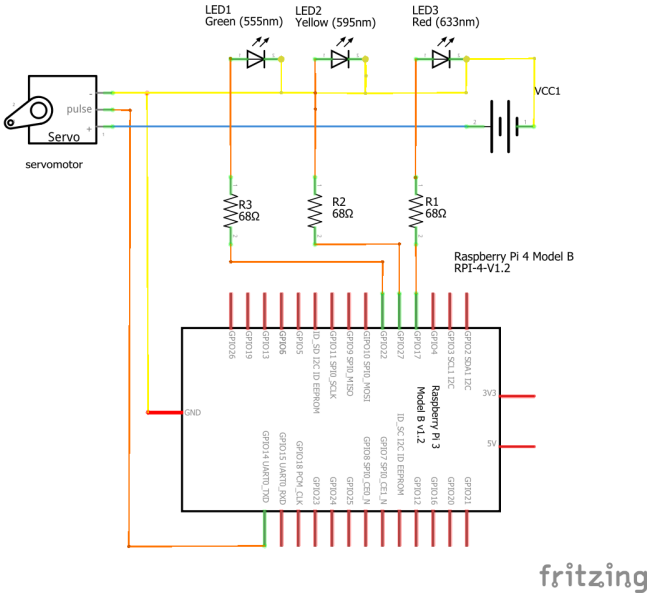

Fig. 9: Esquemàtic part Hardware

A.5 Demostració components Hardware

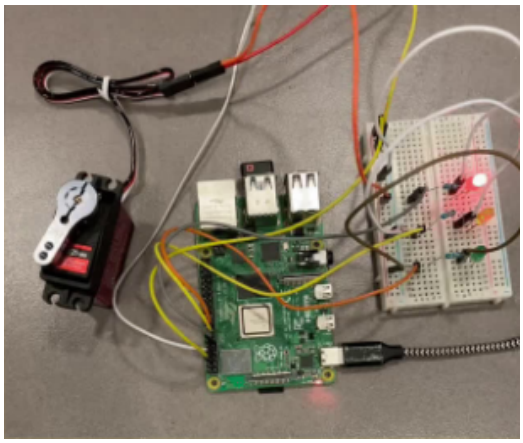

Fig. 10: Muntatge del circuit de la placa Raspberry Pi amb un servomotor i 3 LED