Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

KONTROL INGENIARITZA, AUTOMATIZAZIOA ETA ROBOTIKA UNIBERTSITATE MASTERRA

MASTER AMAIERAKO LANA

FOG APLIKAZIOEN GARAPENA ETA OPERAZIOA BATERATZEKO ETA AUTOMATIZATZEKO EREDUETAN OINARRITUTAKO IKUSPEGIA

Ikaslea: Hurtado Pando, Ekaitz

Zuzendaria: Casquero Oyarzabal, Oscar

Zuzendarikidea: Armentia Díaz de Tuesta, Aintzane

Ikasturtea: 2022-2023

Data: Bilbo, 2023ko uztailaren 17a

Bilboko Ingeniaritza Eskola, Universidad del País Vasco/Euskal Herriko Unibertsitatea

Fog aplikazioen garapena eta operazioa bateratzeko eta automatizatzeko ereduetan oinarritutako ikuspegia

Ikaslea: Hurtado Pando, Ekaitz

Zuzendaria: Casquero Oyarzabal, Oscar

Zuzendarikidea: Armentia Díaz de Tuesta, Aintzane

Saila: Sistemen Ingeniaritza eta Automatika

Ikasturtea: 2022/2023

Masterra: Kontrol Ingeniaritza, Automatizazioa eta Robotika Masterra

Gako-hitzak: Industria 4.0, laino konputazioa, MDE, mikrozerbitzuak, lan-fluxua, Node-RED, Kubernetes, Docker

Laburpena

Industria 4.0k hainbat ikerketa-lerro berri sortu ditu, abantaila garrantzitsuak eskainiz. Ingurune industrialean egoera bera eman da teknologia berritzaileen integrazioarekin. Teknologia horiei esker Fog edo laino konputazioaren aitzinamendua gertatu da, hodei konputazioaren ahultasunak gainditzen dituena, datuen jatorritik gertuago baitago.

Paradigma horretan aplikazio konplexuak hedatzen dira, Fog aplikazioak deiturikoak, mikrozerbitzuen kontzeptua erabiliz funtzionalitate mota anizkoitz abiarazi ditzaketenak. Estandarizazio esfortzuak gauzatu diren arren, Fog aplikazioen garapen eta operaziorako ad-hoc irtenbideak proposatu ohi dira, aplikazioen osagaiak faktore nagusi bezala kontuan eduki barik.

Lan honek Fog aplikazioen garapena eta operazioa bateratzeko eta automatizatzeko euskarria izatea du helburu. Betiere aplikazio berrerabilgarriak eta pertsona orok erabil dezaketenak lortzeko, estandarizazio proposamen garrantzitsuenak jarraitzen dira. Gainera, ereduetan oinarritutako ikuspegia izanez, prozesu osoa era zuzenean beteko dela ziurtatzen da, proposatutako ereduak integra ditzaketen teknologiak hautatu baitira eta erabiltzaileei jarraibide eta tresna egokiak eskaini baitzaizkie.

Bilboko Ingeniaritza Eskola, Universidad del País Vasco/Euskal Herriko Unibertsitatea

Un enfoque basado en modelos para agrupar y automatizar el desarrollo y la operación de aplicaciones Fog

Autor: Hurtado Pando, Ekaitz

Tutor: Casquero Oyarzabal, Oscar

Cotutor: Armentia Díaz de Tuesta, Aintzane

Departamento: Ingeniería de Sistemas y Automática

Curso Académico: 2022/2023

Titulación: Máster en Ingeniería de Control, Automatización y Robótica

Palabras clave: Industria 4.0, computación en la niebla, MDE, microservicios, flujo de trabajo, Node-RED, Kubernetes, Docker

Resumen

La Industria 4.0 ha abierto varias nuevas líneas de investigación, ofreciendo importantes ventajas. En el ámbito industrial se da la misma situación debido a la integración de las tecnologías innovadoras. Gracias a estas tecnologías ha surgido el denominado Fog Computing, el cual supera las debilidades del Cloud Computing al estar más cercano al origen de los datos.

En este paradigma se despliegan aplicaciones complejas denominadas aplicaciones Fog, las cuales son capaces de poner en marcha múltiples tipos de funcionalidades utilizando el concepto de microservicio. A pesar de que se han realizado esfuerzos por estandarizar, para el desarrollo y operación de las aplicaciones Fog se suelen proponer soluciones ad-hoc, sin tener en cuenta los componentes de la aplicación como factor clave.

Este trabajo se propone dar soporte para el desarrollo y operación de aplicaciones Fog. Con el fin de conseguir aplicaciones reutilizables y que cualquier persona las pueda desarrollar, siempre se han seguido las propuestas de estandarización principales. Además, como se hace uso de técnicas de Ingeniería Conducida por Modelos, se asegura que el proceso completo se ejecute correctamente, ya que se han seleccionado tecnologías capaces de integrar dichos modelos, y se han proporcionado pautas y herramientas adecuadas a los usuarios.

Bilboko Ingeniaritza Eskola, Universidad del País Vasco/Euskal Herriko Unibertsitatea

A model-based approach for integrating and automating fog application development and operation

Author: Hurtado Pando, Ekaitz

Supervisor: Casquero Oyarzabal, Oscar

Co-supervisor: Armentia Díaz de Tuesta, Aintzane

Department: Systems Engineering and Automatic Control

Academic course: 2022/2023

Master's degree: Master's Degree in Control, Automation and Robotics Engineering

Keywords: Industry 4.0, Fog Computing, MDE, microservices, workflow, Node-RED, Kubernetes, Docker

Abstract

The Industry 4.0 has created several new lines of research, offering significant advantages. The same situation occurs in the industrial field due to the integration of innovative technologies. Thanks to these technologies, the so-called Fog Computing has emerged, which overcomes the weaknesses of Cloud Computing by being closer to the source of the data.

In this paradigm, complex applications called Fog applications are deployed, which are able to implement multiple types of functionalities using the concept of microservices. Although standardization efforts have been made, ad-hoc solutions are often proposed for the development and operation of Fog applications, without taking into account the components of the application as a key issue.

This work aims to support the development and operation of Fog applications. In order to achieve reusable applications that can be developed by anyone, the main standardisation proposals have always been followed. Furthermore, as Model Driven Engineering techniques are used, it is ensured that the whole process is executed correctly, as technologies capable of integrating such models have been selected and appropriate guidelines and tools have been provided to the users.

Aurkibidea

Irudien Zerrenda

Taulen Zerrenda

Akronimoak

INCAR	Ingeniería de Control, Automatización y Robótica
UPV/EHU	Universidad del País Vasco/Euskal Herriko Unibertsitatea
IKT	Informazio eta Komunikazio Teknologiak
IoT	Internet Of Things
TOSCA	Topology and Orchestration Specification for Cloud Applications
MAL	Master Amaierako Lana
IDE	Integrated Development Environment
MDE	Model Driven Engineering
DDF	Distributed Dataflow
M2T	Model-to-Text
OAM	Open Application Model
CBSE	Component Based Software Engineering
HTTP	Hypertext Transfer Protocol
XSLT	Extensible Stylesheet Language Transformations
GCIS	Grupo de Control e Integración de Sistemas
API	Application Programming Interface
GUI	Graphical User Interface
YAML	YAML Ain't Markup Language
CLI	Command-line interface

Atala I

SARRERA

1 Sarrera eta testuingurua

Edukia

Kapitulu honetan zehar garatutako master amaierako lanaren sarrera eta kokatzen den testuingurua zehaztuko da.

1.1 Sarrera

Gizadiak antzinatik, baliabide teoriko eta praktikoaz baliatuz, problema konplexuak konpontzeko gai izan da, soluzioak aurkituz. Erromatar inperioak, adibidez, uraren garraiorako kanalizazio sistemak eraiki zituen, garai hartan handitzen hasi zen populazioaren betebeharrak asetzeko [1].

Soluzio horiek lortzeko beharrezko teknologiaren azterketa eta aplikazioaz arduratzen den pertsonari ingeniari deritzo, latinezko "*ingenium*" hitzetik datorrena. Kontzeptu horretatik abiatuta, historian zehar espezializazioa edo lanaren banaketa gauzatu da. Hasieran espezializatu ziren langileak artisauak izan ziren, garrantzi handia hartu zutenak XVIII. mendean Industria Iraultza gertatu zen arte.

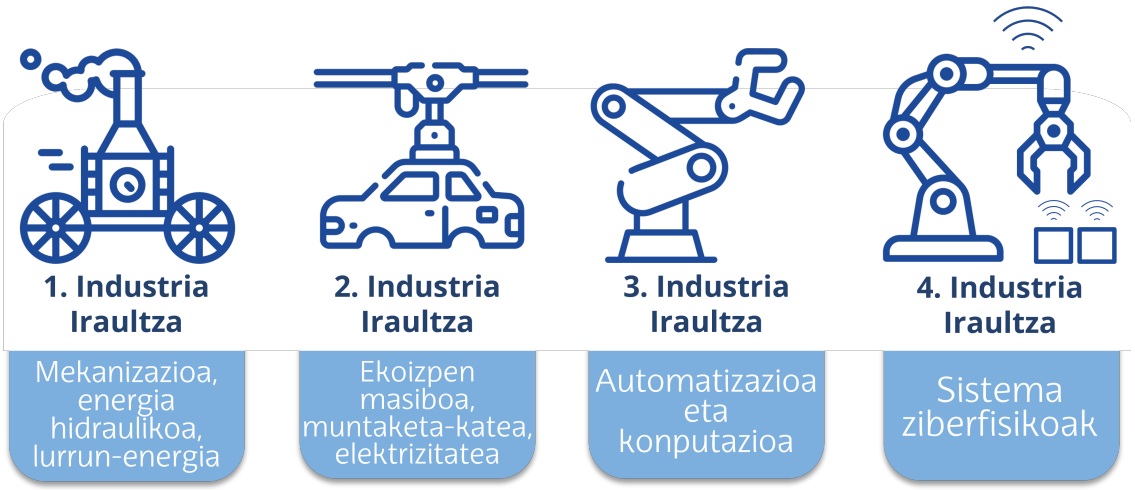
Industria Iraultza Ingalaterran sortu zen aldaketa teknologiko, sozioekonomiko eta kulturala izan zen. Gertaera honi esker, gizarte nekazaria gizarte industrial bilakatu zen, berrikuntza teknikoak eta makinariaren garapena ahalbidetuz. Aldaketa eta berrikuntza hauek garapen-fase gehiago sortu zituzten. Horrela, XIX. mendearen amaieran Bigarren Industria Iraultza bizi izan zen. Ordura arte, energia-iturri gisa erabiltzen ziren ikatza eta lurruna petrolioa eta elektrizitateaz aldatu ziren [2].

XIX. mendearen bukaeran, batez ere argindarraren sorkuntza zela eta, garapen teknologiko eta ekonomiko hori gehiago sakondu zen. Ondoren, XX. mendearen azkenengo erdian agertu zen Hirugarren Industria Iraultzan aipatutako energia sistema berriek informazio eta komunikazio teknologia berriekin (IKTak) bat egin zuten, hau da, Internetekin batera sortu ziren teknologiak. Horrek, aukera-leiho berriak zabaldu zituen, hala nola, konputazioa eta automatizazioa [3].

Internetaren sorkuntzak eta IKTek informazioa prozesatzeko eta konpartitzeko era berriak ahalbidetuz, murgilduta gauden Laugarren Industria Iraultza agertu zen. Kontzeptu

hau, edo Industria 4.0 ere deitua, 2011ko Hannoverreko makina-erremintaren azokan aurkeztu zen lehen aldiz. Ekoizpen prozesuei moldagarritasun handiagoa eskaintzeko eta baliabideen esleipena optimizatzeko teknologia berriak garatu dira, batez ere, informazioaren prozesamenduan eta sistema ziberfisikoetan [4].

Industria 4.0rekin smart factory edo fabrika adimentsuen kontzeptua agertu da. Fabrika horien bereizgarritasuna integratzen dituzten teknologiak dira. Hainbat ikerketa-lerro berri agertu dira: Gauzen Internet (IoT), Big Data, hodei konputazioa, robotika, etab. Hala ere, azken joerek teknologien integrazioan dute ardatza.



Irudia 1.1: Industriaren eboluzio historikoa.

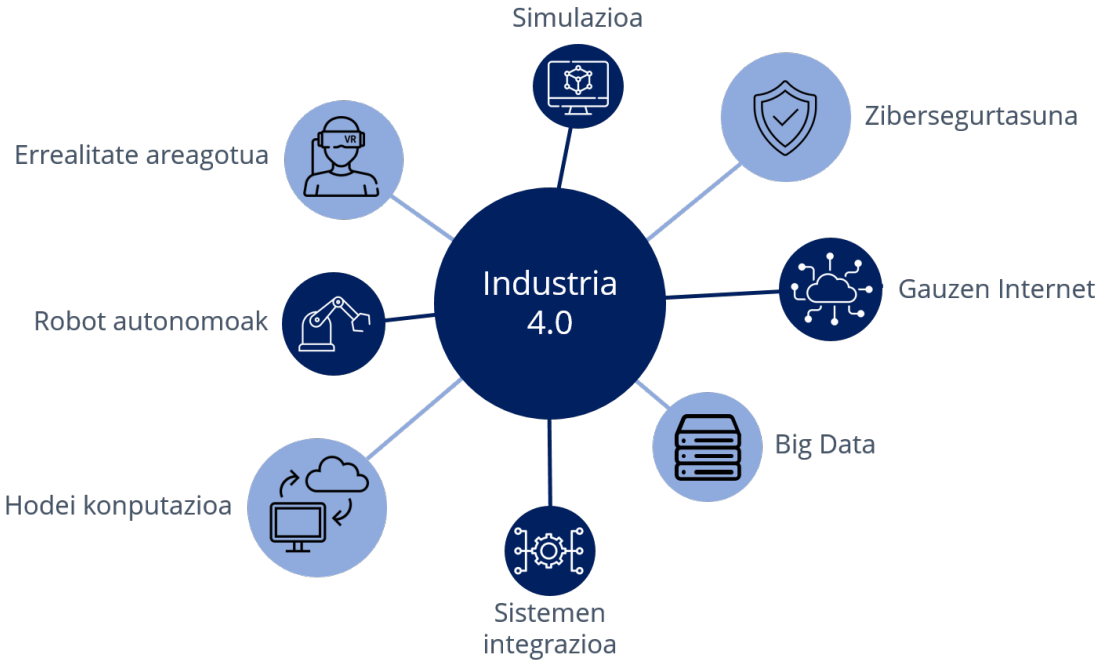
1.1. Irudian aipatutako industriaren eboluzioa erakutsi da. Aztertu daitekeen bezala, garai bakoitzean eskuragarri zeuden teknologiak prozesuak automatizatzeko eta optimizatzeko erabili izan dira, eta ideologia hori gaur egunera arte iraun du. Azken hamarkadetan komunikazio eta informazio teknologien hazkundeak hainbat esparruen garapena bultzatu du.

Industria 4.0an hainbat arlok garrantzi handia hartu dute. Horietako bat hodei konputazioa edo Cloud Computing deritzona da, Interneten globalizazioagatik eta konputazio kostu txikiagotzearekin hazkunde nabarmena jaso duena. Gainera, prozesuen eraginkortasun eta optimizazioan laguntzen duen teknologia da, azken joeretako bat dena [4].

Paradigma hori konputazio arkitektura banatu batean oinarritzen da. Komunikazio-sareen eboluzioarekin, aplikazio konplexuak sortzeko aukera agertu da eta teknologia horiek aplikazioak exekutatzeko erabili izan dira. Ondoren, birtualizazio teknologiak aplikazio horiei ezaugarri berriak gehitzeko garatu izan dira: planifikazio eta birkonfigurazio dinamikoa, elkarreragingarritasuna, etab.

1.2 Testuingurua

Azken urteetan Industria 4.0 merkatuko eskakizunei erantzuna emateko agertu da, kalitate eta optimizazioari dagokionez [5]. Kontzeptu honen euskarrietako bat fabriketako aktiboek sortutako datuetatik fabrikazio prozesuan balio erantsia lortzea da. Aktibo horiek fabrikako behe mailan kokatzen diren baliabideen multzoa da [6]. Horrela, industriaren aurrerapenarekin hainbat teknologia paradigma honetara integratu dira: hodei konputazioa, Big Data, sistemen integrazioa... (ikus 1.2. Irudia) [7, 8].



Irudia 1.2: Industria 4.0n integratutako teknologiak.

Teknologia horiek aplikazio konplexuen garapena ahalbidetzen dute, hainbat esparrurekin erlazionatuta daudenak, hala nola, analitika adimentsua, plangintza dinamikoa edo prebentziozko mantentzea [8, 9, 10]. Hori dela eta, aplikazio-eremu desberdinetan ezarri daitezkeen aplikazioak dira, domeinuaren araberako eskakizunak bete behar dituztenak. Esaterako, bideragarritasuna eta errentagarritasuna aplikazio hauen ezaugarri garrantzitsua da [11].

Bestetik, fabrikazio sistemen hierarkia interkonektibitate eredu global batean bihurtu da. Ikuspuntu estruktural batetik hierarkia hori hiru mailetan egituratzen da: 1) fabrika edo edge deiturikoa, aktibo fisikoak fabrikazio eta garraio zerbitzuak eskaintzen dituztenak; 2) lainoa edo fog, konputazio eta memoria baliabideak datuen jatorritik gertu eskaintzen duena; eta 3) hodeia edo cloud, lainoaren antzeko ezaugarriak dituena, baina baliabideak kanpoko zerbitzarietan kokatuta [8, 12].

Fabrikako aktiboek aplikazio horiek jasateko biltegiratze eta konputazio ahalmen nahikorik ez daukatenez, hasiera batean aplikazioak hodeian hedatu ziren [11]. Hala ere,

inplementazio honek zenbait arazo dakar, hala nola, latentzien agerpenak eta datuen atzipenean kalteberatasuna. Horregatik, aplikazioak aipatutako maila zehazki batean hedatu dira: lainoan [13].

Fog aplikazioetarako eskuragarria dagoen softwarea bi taldetan bana daiteke: framework eta software plataforma. Lehenengoa aplikazioen garapenerako beharrezkoak diren metodologia eta tresnen multzoa da, eta bigarrenak, berriz, aplikazioak orkestratzeko funtzionaltasunak eta mekanismoak eskaintzen dituen inguruneari egiten dio erreferentzia [12, 14].

Plataforma horiek ezaugarri jakin batzuk bete behar dituzte: malgutasuna, birkonfigurazioa, egokigarritasuna, segurtasuna, hedagarritasuna, elkarreragingarritasuna eta eramangarritasuna, besteak beste [8, 11]. Horregatik, birtualizazio arinezko teknologiek garrantzi handia hartu dute, ezaugarri horiek betetzeko aukera eskaintzen baitute [15].

Erakundeen aldetik, Fog eta Cloud Computing paradigmaren estandarizazioa sustatu dute, esaterako, OpenFog [16] edo TOSCA [17]. OpenFogek Fog aplikazioak elkartutako mikrozerbitzuen bilduma gisa definitzen ditu, eta TOSCAk aplikazioak zerbitzu bilduma gisa adierazten dituzten lan-kargak definitzeko topologia-txantiloien erabilera deskribatzen du.

Mikrozerbitzuak osagai txiki bezala ulertu dira, horien mugen definizioa diseinu erronka kritikoa izanda [18]. Osagai bakoitza funtzionalitate bakarra betetzen du eta beste osagai batzuekin elkarreragiteko sarrerak eta/edo irteerak ditu [19]. Egile batzuek ikuspegi hori datu-fluxu (edo lan-fluxu) zuzendu gisa mugatzen dute [9, 20, 21]. Ikuspegi hori erabiliz, sinpletasuna, hedagarritasuna eta malgutasuna eskaintzen dio aplikazioen garapenari.

Horrela, aplikazioaren logika osagai independente eta hedagarritan banatzen da, pertsonalizatutako eskaerei erantzuteko elkarrekin lan egiten dutenak [18, 22]. Horrek, hardware, protokoloak eta funtzionalitatea fluxuaren nodo bezala abstraitzea ekartzen du. Gainera, mikrozerbitzuak sortzeko birtualizazio teknikak erabiltzen dira, aipatutako ezaugarriak betetzeko [11, 15].

Planteamendu hori jarraituz, hainbat lanek modelatze teknikak erabili dituzte Fog aplikazioen definiziorako. Hala ere, horietako inork ez ditu ikusmolde, diseinu eta hedapenerako tresnak eskaintzen [20, 21]. Nahiz eta estandarizazio saiakerak egin diren, aplikazioaren diseinua eta osaerarako ad-hoc irtenbideak proposatu ohi dira, aplikazio-domeinu ezberdinetarako aplikagarriak ez direnak.

Testuinguru honetan, master amaierako lan (aurrerantzean MAL) honetan Fog aplikazioen garapena eta operazioa bateratzeko eta automatizatzeko ereduetan oinarritutako ikuspegia aurkezten da. Horrela, Fog aplikazioaren fase guztiak kontuan hartuko dira: diseinua, garapena eta inplementazioa edo operazioa. Lehenengo bi faseetarako automatizaziorako garapen-ingurune integratu bat (ingelesez, Integrated Development

Environment edo IDE), eta azkenengorako, Fog aplikazioen hedapena ahalbidetzen duen software plataforma bat proposatu da. Atal guztiek ereduetan funtsatutako tekniketan (ingelesez Model Driven Engineering edo MDE techniques) eta mikrozerbitzuen lan-fluxuen kontzeptuan oinarritu dira.

Helburuak eta irismena

2

Edukia

Atal honetan proiektuan zehaztutako helburuak eta lanaren irismena aipatuko dira.

2.1 Helburuak

MAL honen helburu nagusia Fog aplikazioak diseinatzeko, garatzeko eta inplementatzeko ikuspegi generiko bat proposatzea da, estandarren kontzeptuak jarraituz. Metodologia horrek, lan-fluxuaren kontzeptua barneratu beharko du, eta ereduetan funtsatutako teknikak erabili beharko dira prozesu osoan zehar.

Lortutako aplikazioek hainbat ezaugarri bete beharko dituzte: hedagarritasuna, elkarreragingarritasuna, eramangarritasuna, etab. Gainera, prozesu osoa automatizatu beharko da, fase guztiak zuzen betetzen direla egiaztatzeko.

Aplikazioen osagaiak mikrozerbitzuak izango dira, beraz, funtzionalitate bakarra exekutatzeko eta beste osagai batzuekin komunikatzeko gai diren elementuak lortu beharko dira. Bestalde, elementuen berrerabiltzea eta arintasuna bermatu beharko da, horretarako sorkuntza-prozesuan beharrezko mekanismoak eskainiz (adibidez, kode-iturria garatzeko programazio-lengoaia desberdinak ahalbidetzea).

Helburu nagusia lortzeko, helburu partzial hauek zehazten dira, atal bakoitza ondo betetzen dela frogatzeko:

- Fog aplikazioentzako meta-eredu bat garatzea, aplikazio mota horien ezaugarriak kontuan hartzen duena.
    - Aplikazioen osagaientzako meta-eredua garatzea, beharrezko informazio guztia kontuan hartzen duena.
    - Aplikazioentzako meta-eredua garatzea, osagaien meta-ereduarekin integra daitekeena eta estandarrekin bat datorrena.
- Aplikazioen osagaiak eraikitzeko software egokiena lortu, mikrozerbitzuak sortzeko balio behar duena.

- Fog aplikazioak diseinatzeko eta garatzeko software ingurune bat garatzea. Proposatutako meta-ereduak integratzeko ahalmena izan beharko du.
    - Software inguruneen aukera desberdinak aztertu, egokiena hautatzeko.
    - Software ingurunean meta-ereduak integratu.
    - Software ingurunea moldatu, meta-ereduekin bat datozen aplikazioen definizioak lortzeko.
- Fog aplikazioen inplementazioa edo operazioa, bai hedapena, bai kudeaketa, ahalbidetzen duen software plataforma bat garatzea.
    - Inplementazio hori ahalbidetzen dituzten plataforma desberdinak aztertu, egokiena hautatzeko.
    - Plataforma horretan aplikazioa lehen mailako entitate bezala integra daitekeen aztertu.
    - Plataforma horretan aplikazioak hedatzeko baliabideak aztertu.
    - Plataforma moldatu, meta-ereduak integratuz eta aplikazioak erraz heda daitezkeen zerbitzuak gehituz.

2.2 Irismena

Aurreko lanei esker, Fog aplikazioak garatzeko teknologien ezagutza lortu da. Beraz, birtualizazio teknologietan oinarritutako osagai anitzeko aplikazioak eraikitzeko ikerketa burututa dago. Lan honek ezagutza hori erabiliko du, aplikazioen garapen eta operazio faseenetarako.

Proiektu honen irismena ikuspegi generiko bat lortzea da, mikrozerbitzuetan oinarritutako Fog aplikazioak lortzeko aukera eskaintzen duena. Proposatutako ikuspegia domeinu desberdinerako balio beharko du.

MAL honen bitartez, Fog aplikazioak diseinatzeko eta garatzeko meta-ereduak definitzen dira, eta horiek integratu ditzaketen teknologiak aurkezten dira. Bestalde, aplikazio horiek inplementatzeko software plataforma bat proposatzen da.

Definitutako kontzeptu-proba garapen-ingurune integratuan diseinatu eta erabilitako plataforman hedatu delarik, lan honetan proposatu den ikuspegiaren funtzionamenduaren zuzentasuna frogatu da, eta beren abantailak eta desabantailak aztertu dira.

Lanak dakartzan onurak

3

Edukia

Ataza honetan proiektu honek dakartzan onurak aztertuko dira. Esan beharra dago, onura gehienak Industria 4.0k aurkezten dituenekin bat datoz. Analisia burutzeko, bi zatitan banatuko da, lehenik eta behin, onura teknikoak aztertuz, eta, ondoren, onura ekonomikoen analisia betez.

3.1 Onura teknikoak

Hasteko, mikrozerbitzuetan oinarritutako Fog aplikazio konplexuak eraikitzeko aukera eskainiz, elkarreragingarritasuna betetzen da, hau da, garatzaile bakoitzak era desberdinean sortutako softwareak komunikatu eta elkar lan egin ahalko dute. Gainera, aplikazio konplexuen eraikuntza era eraginkorrean gauzatuko da.

Beste alde batetik, jarraitutako plangintzagatik erabilitako teknologiak hedagarriak izango dira, etorkizunean funtzionalitate berriak gehitzeko aukera eskainiz. Horrek, bai aplikazioak diseinatzeko eta garatzeko teknologiak, bai aplikazioak inplementatzeko plataformak bete egingo dute.

Gainera, Industria 4.0an oso garrantzitsua den birkonfigurazio dinamikoa eskaintzen du, hau da, aplikazioak eta beren osagaiak birdiseinatu daitezke, eta erabilitako teknologiek ezaugarri hori betetzeko zerbitzuak eskainiko dituzte.

Lan honen ekarpenaren zati handi bat software garapena izanda, segurtasuna kontuan hartu behar da. Horregatik, proposatutako ikuspegiak segurtasuna areagotzen du, ezin baitira edozelako aplikazioak hedatu, proposatutako meta-ereduarekin bat etorri beharko baitira.

3.2 Onura ekonomikoak

Onura ekonomikoak proiektu guztien atal garrantzitsua da, azken urteetako ziurgabetasun ekonomikoa eta lehiakortasun handiko merkatuak direla eta. Hori dela eta, proiektu honen kontribuzioa hainbat mozkin ekonomiko eskaintzen ditu.

Aplikazio eraginkorragoak eginez, eta beraien osagaien berrerabilpena ahalbidetuz, kostuen txikitze nabarmena betetzen da. Honekin batera, energiaren optimizazioa dator, aplikazioen garapen-denborak murriztu ahalko baitira.

Bestalde, aukeratutako teknologiak erabiliz aplikazioak monitorizatu ahalko dira. Horrek, hainbat arazo ekiditeko aukera eskain dezake, mantentze kostuak txikituz eta produktuaren kalitatea handituz. Horrekin erlazionaturik, aurreztutako dirua sisteman berrinbertitzeko eta eraginkortasuna areagotzeko balio dezake.

Bukatzeko, birtualizazio teknologien erabilerak kostu konputazionala murriztea dakar, aplikazio oso arinak sortzeko aukera eskaintzen baitute. Gainera, sorkuntza prozesurako malgutasun handiagoa edukiz, garatzaileek etekin gehiago lortzeko ahalmena izango dute.

4 Gaiaren egoeraren azterketa

Gaiaren egoeraren azterketa burutzeko Fog aplikazioen diseinu, garapen eta operazioaren inguruko hainbat artikulu desberdin identifikatu dira, beraien ekarpenak eta mugak aztertuz. Literaturako lan batzuek Fog Computing paradigma eta bertan hedatzen diren aplikazioak era orokor batean aztertzen dituzte [23, 24, 25], baina eskuarki baliabide eta aplikazioen kudeaketaren ikuspegitik.

[26] artikuluan datu-fluxu banatuan (ingelesez Distributed Dataflow edo DDF) oinarritutako programazio eredua erabiltzen dute IoT aplikazioak garatzeko, lainoko edo hodeiko gailuetan heda daitezkeenak. Aplikazioak grafo zuzendu bezala definitzen dira, grafoaren nodo bakoitza sarrera eta irteerak dituen prozesatzeko unitate independentea dena. Hortaz, bi garatzaile bereizten dira: nodo eta aplikazio garatzaileak. Lehenengorako ez da eredurik proposatzen, baina bigarrenerako Node-RED tresnaren hedapen bat aurkezten da.

Antzera, [27]-an DDF programazio eredua erabiltzen dute baita ere, baina denbora errealeko beharrekin. Ondorioz, IoT aplikazioak kontzeptu berdinean ardazten dira, hau da, grafo zuzendu gisa. Lan honetan, berezko interfaze grafikoa proposatzen da, aplikazioen eta sare-topologia diseinatzeko aukera eskaintzen duena. Gainera, aplikazioak inplementatzeko eta probatzeko Java klase batzuk proposatzen dituzte.

IoT sistemak ere [28]-an aurkeztutako lanaren ardatza dira, Domain Specific Language edo DSL bat proposatzen dena. Honek, maila anitzeko aplikazioen diseinua ahalbidetzen du, IoT gailuak eta Edge, Fog eta Cloud nodoak barne hartzen dituena. Bi meta-eredu proposatzen dituzte, batetik IoT sistemak definitzeko, eta bestetik, egoera desberdinei aurre egiteko egokitzen diren legeak definitzeko. Lan honek, aplikazioa hedatzeko beharrezko fitxategiak lortzeko M2T (Model-to-Text) transformadore bat aurkezten du. Hala ere, ez du aplikaziorako eredu bat proposatzen, osagai guztiak edukiontzi berdinean kokatzen direla suposatzen baitu.

[21]-ean egileek aplikazioak diseinatzeko meta-ereduak ere definitzen dituzte, prozesatzeko zeregin erreaktiboen lan-fluxu gisa. Zereginak bananduta diseinatutako gaien bidez lotuta egonda, lan honetan begizta gabeko grafoak erabiltzen dira, gaien erabilera zehazten duen kontrol-fluxua definitzeko. Meta-ereduek aplikazioaren hedapena automatizatzen duten arren, lan honek ez du aplikazioak grafikoki diseinatzeko tresnarik aurkezten. Bestalde, meta-ereduak ez dira inplementazio teknologietatik guztiz independenteak.

[29] artikuluan osagaiak publish/subscribe mekanismoak erabiltzen dituzte elkarren artean komunikatzeko, aplikazioak “plug-and-play” izendatutako osagaiekin sortuz. Hainbat

osagai mota desberdin bereizten dituzte, hala nola, kontrolatzaileak, sentsoreak, eragingailuak, web erabiltzaile-interfazeak, etab. Dena den, egileek ez dute aplikazioen edo haien osagaien modelizazio formalik aurkezten.

[30]-eko egileak TOSCA [17] estandarrean eta Docker [31] teknologian oinarritzen dira osagai anitzeko aplikazioen hedapena gaitzeko. Aplikazioaren zehaztapenetarako, egileek TOSCAn oinarritutako irudikapen bat proposatzen dute, aplikazioen software osagaiak eta beharrezko Docker baliabideak deskribatzeko aukera ematen duena. Aplikazioa diseinatzeko interfaze grafikorik aurkezten ez duten arren, konfigurazio fitxategiak automatikoki lor daitezke. Horrela, osagaien kudeaketa lortzen da, horien bizi-zikloa exekutatzen ari diren edukiontziengandik banatzen baita. Planteamendu honekin, aplikazio-diseinatzaileak erabilitako teknologien, hau da, aplikazioen modelatzearen eta osagaien garapenerako softwarearen ezagutza izan behar du. Ondorioz, ez dago programatzaileen eta diseinatzaileen arteko interes-banaketarik.

Bestetik, [32]-an Kubernetes [33] plataformaren hedapen bat aurkezten da, Application-Centric Orchestration Architecture (ACOA) izendatuta. Helburua Edge-Fog-Cloud continuumean hedatutako aplikazioen zerbitzuaren kalitatea optimizatzea da. Hori lortzeko, aplikazio-eredu bat proposatzen da, lan-fluxu eta grafo zuzenduetan oinarrituta, osagaiak eta erlazioak barne hartzen dituena. Aipatutako osagaiak bananduta hedatzen dira, baina aplikazioa osatzerakoan lotuta daude, ereduak aplikazioa elementu txikien multzo bezala hartzen baitu. Hortaz, osagaien modelizazioa ez da lan honetan aurkezten. Gainera, Kubernetes plataformaren menpeko soluzio bat da.

[34] argitalpenean egileek Kubernetes [33] plataformaren hedapen bat aurkezten dute, mikrozerbitzuetan oinarritutako aplikazioen eskuragarritasuna kudeatzeko. Horretarako, egileek Kubernetesen integra daitekeen kontroladore bat proposatzen dute, aplikazioen egoera erreplika eta zerbitzuen birbideratzea eskaintzen duena. Lan honetan ez da aplikazioak diseinatzeko eta garatzeko eredurik erabiltzen, osagaiak exekuzio-fasean daudela suposatzen baita. Hori dela eta, modelizazio eredu orokor bat erabiliz, soluzio honek ez luke funtzionatuko.

Beste alde batetik, [35] bezalako lan batzuek TOSCA estandarra erabiltzen dute ereduetan oinarritutako aplikazioak sortzeko. Kasu honetan, Open Application Model edo OAM eredua erabiltzen dute aplikazioak definitzeko, roletan oinarritutako orkestrazio proposamen bat aurkeztuz. Lan honetan egileek ez dira aplikazioen eta beraien osagaien garapenean ardazten, baina kontzeptu-proba garatzeko software plataforma bezala Kubernetes hautatzen dute. Beraz, ez dute aplikazioak diseinatzeko eta garatzeko tresna grafikorik adierazten. Hori dela eta, eredu estandarrak erabili arren, diseinatzaile eta garatzaileek teknologiaren ezagutza eduki behar dute, kasu honetan, Kubernetesko konfigurazio eta hedapen fitxategiena.

[36] artikuluak TOSCA estandarrean oinarritutako Cloud aplikazioak modelatzeko eta beraien arteko erlazioak zehazteko ekarpena egiten du. Kubernetes plataforma hedatzen

duen “orcon” deituriko orkestratzailea aurkezten dute kontzeptu-proba gisa, zerbitzuen arteko erlazioen kudeaketa eskaintzen duena, bizi-zikloaren sinkronizazioa eta konfigurazio aldaketak ahalbidetuz. Eredu estandar bat erabili arren, beste lan batzuk bezala, ez du tresna grafikorik erabiltzen, eta, hortaz, aplikazioaren diseinatzaileak teknologiaren ezagutza izan behar du. Gainera, erabilitako plataformaren dependentzia handiko soluzioa da.

Ondorioz, ez dago aplikazioaren modelizazio generikoa biltzen duen proposamenik, barne hartzen dituena, besteak beste: 1) faktore nagusi gisa hartzen duen osagaien modelizazioa; 2) edukiontziak sortuko dituzten irudien kudeaketa eta pertsonalizatzea, mikrozerbitzuen oinarria izango direna; 3) tresnen erabilera, interfaze grafikoak, esaterako, aplikazioaren diseinatzailea lotutako teknologietatik abstraitzeko; eta 4) proposamena inplementazio teknologiatik independentea izateaz gain estandarrak jarraitzea, orokortasuna betetzeko.

Aukeren analisia

5

Edukia

Kapitulu honetan proiektuan zehar erabilitako teknologia garrantzitsuen analisia beteko da. Lanaren zati garrantzitsu bat garapen-ingurune integratu baten erabileran datza, aplikazioak diseinatzeko balio beharko duena. Gainera, edukiontziek garrantzi handia dute, horiek eskaintzen dituzten onura guztiengatik aplikazioek osatzen dituzten osagaien oinarria izango baitira. Hain zuzen ere, mikrozerbitzuak eraikitzeko aukera egokiena dira. Azkenik, edukiontzi horiek hedatzeko eta kudeatzeko plataforma bat beharrezkoa da.

Hori dela eta, hiru ataletan bananduko da erabiliko diren teknologien aukeren analisia. Batetik, garapen-ingurune integratuen analisia beteko da, ereduetan oinarrituta egongo dela kontuan hartuz; bestetik, mikrozerbitzuak izango diren edukiontziak eraikitzeko teknologien ikerketa burutuko da, eta, amaitzeko, mikrozerbitzu hauek hedatzeko eta kudeatzeko beharrezkoak diren teknologiak aztertuko dira.

5.1 Garapen-ingurune integratuen aukeren analisia

Garapen-ingurune integratuen bilaketan hainbat gauza kontuan izan dira. Lehenik eta behin, ereduetan oinarritutako prozesua izanda, aukerek eredu hauek integratzeko aukera eskaini behar dute. Bestalde, osagai berriak sortzeko gai izateko beharrezko pertsonalizazio maila eduki behar dute. Aplikazioak diseinatzeko tresna izango denez, aukerek diseinurako interfaze egoki, sinple eta argia izatea kontuan edukiko da, interfaze hori grafikoa izanez balorazioa handituz.

Node-RED

Node-RED IBMk garatutako fluxuetan oinarritutako garapen tresna bat da. 2016an OpenJS Foundation erakundeko kode-irekiko proiektua bihurtu zen. Bere helburu nagusia hardware gailuak, APIak eta online zerbitzuak Gauzen Internetaren barruan elkarkonektatzea da eta ikusmen-programaziorako diseinatua izan da.

Horretarako, web-nabigatzailean oinarritutako fluxu editore grafikoa eskaintzen du. Sortutako aplikazioen osagaiak gorde eta berrerabil daitezke, bai osagai pertsonalizatuak,

bai plataformaren berezkoak direnak. Gainera, plataforma hau IoT sarearen ertzean koka daiteke, nahiko arina baita. Zerbitzariaren aldetik, berriz, gertaeretan oinarritutako JavaScript kodea garatzeko aukera eskaintzen du.

Apache Airflow

Apache Airflow kode irekiko lan-fluxuen kudeaketarako plataforma bat da, 2014an sortu zena. Pythonen garatuta dago, eta lan-fluxuak programazio lengoaia horretan diseinatzen dira. Plataforma hau konfigurazioa kodearen bitartez betetzeko ideiarekin sortu zen.

Beste soluzio batzuek XML lengoaia erabiltzen dute lan-fluxuak modelatzeko. Honek, Python erabilita, kanpoko liburutegiak inportatzeko aukera eskaintzen du. Bestetik, kudeaketarako, DAGak edo grafo zuzendu aziklikoak erabiltzen ditu, beraz, horrek, plataforma hedagarria eta dinamikoa izatea ahalbidetzen du.

ThingsBoard

ThingsBoard kode-irekiko Gauzen Interneteko plataforma bat da, IoT proiektuak era azkarrean garatzea eta kudeatzea ahalbidetzen duena. Hainbat ekintza bete daitezke plataforma honekin: gailu eta aktiboetatik datuak lortu eta bistaratu, lan-fluxuak eraiki, panel dinamikoak diseinatu, etab.

ThingsBoard soluzioaren barruan Rule Engine deritzon softwarea aurki daiteke, gertaeretan oinarritutako lan-fluxuak eraikitzeko tresna dena. Aukera pertsonalizagarria eta konfiguragarria izanda, gertaera konplexuak sor daitezke eta osagaiek elkarren artean bidaltzen diren mezuekin lan egin (filtratu, eraldatu...).

Taula 5.1: Garapen-ingurune integratuen konparaketarako taula.

	Teknologiak		
Ezaugarriak	**Node-RED**	**Apache Airflow**	**ThingsBoard**
Kode irekia	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)
Komunitatea	Handia	Handia	Ertaina
Konfigurazioa	Erraza	Erraza	Konplexua
Dokumentazioa	Handia	Ertaina	Handia
Interfaze grafikoa	Bai	Ez	Bai
Hedagarria	Bai	Bai	Bai

Aukera guztiak aztertu eta gero, lortutako ondorioak aurkezteko, 5.1. Taula erakutsi da euskarri gisa. Node-RED aukera errazena eta pertsonalizazio eta komunitate maila altuena duen aukera da, Apache Airflow aukerak ezaugarri interesgarriak eskaintzen ditu, baina irudi interfazea ez duena; ThingsBoard, azkenik, plataforma osoena eta konplexuena da.

Karakteristika guztiak kontuan hartuz, aukera optimoena Node-RED dela ondorioztatu da, pertsonalizazio maila altua eskaintzen baitu interfaze grafiko sinple batekin. Bestalde, berezko nodoak sortzeko aukera eskaintzen du, Gauzen Internetarako plataforma batetik, Fog aplikazioak diseinatzeko plataforma batera bihurtzeko ahalerarekin. Alabaina, beste teknologiak oso baliagarriak dira, zenbait puntu kontuan hartuz: esaterako, Apache Airflowren kasuan Pythonen ezaguerak eduki behar direla.

5.2 Mikrozerbitzuak sortzeko aukeren analisia

Komentatu den bezala, mikrozerbitzuak izango diren edukiontziak sortzeko hainbat birtualizazio arinezko teknologia bilatu dira. Guztien abantailak eta desabantailak aztertu dira, proiektu honetarako egokiena dena hautatzeko.

Docker

Docker 2013an sortu zen kode-irekiko proiektua da, edukiontziak sortzeko eta kudeatzeko tresnak eskaintzen dituena. Dockerren jaurtiketak edukiontzien teknologiari bultzada garrantzitsua eman zion, kontainerrak sortzeko, kudeatzeko eta exekutatzeko ikuspegi orokorra aurkeztu baitzuen. Harrezkero, garatzaile eta enpresa askok Docker bereganatu dute eta informazio teknologien industrian edukiontzien teknologiaren harrera zabala bultzatu du.

Teknologiak honek edukiontzi independenteak Linux instantzia bakarrean exekutatzeko aukera ematen du, makina birtualen abiaraztea eta mantentze-lanen gainkarga saihestuz. Edukiontzi horien barruan aplikazioa exekutatzeko baliabide guztiak daude: tresnak, plataformak, liburutegiak, etab. Sormen prozesurako aukera desberdin asko eskaintzen dituen plataforma da, konfiguragarritasun handia suposatzen duena.

Bestalde, ondoren azalduko diren orkestratzaile askorekin integrazio erraza ahalbidetzen du. Docker Swarmekin, adibidez, integrazioa ezin hobea da, konpainia berdinak sortutako teknologiak baitira. Honek, abantaila nabarmena suposatzen du, gaur egungo industrian softwarearen integrazioa atal garrantzitsua baita.

RKT

RKT, edo *Rocket* ere izendatua, CoreOSek 2014an edukiontziak garatzeko jaurtitako motorra da. Kasu praktiko askorentzako erabili izan da eta Linux distribuzio gehienetan

eskuragarri dago. Red Hat konpainiak proiektua Cloud Native Computing Foundation (CNCF) fundazioari eman zion, OpenSource bihurtuz.

Edukiontzien plataforma honek segurtasunari ematen dio arreta, bereziki. RKT kontainerrek ez dute zalantzazko funtzionaltasunik onartzen, erabiltzaileak esplizituki funtzio ez hain seguruak gaitu ezean. RKTren arkitekturan exekuzio unitate bakoitza Unix prozesuentzako eredu klasikoan exekutatzen da, ingurune autonomo eta isolatua baita.

Tresna honek instalazio eta konfiguraziorako aukera desberdinak eskaintzen ditu, baina hori, abantaila bat izan ezik, bere erabilerarako konplexuegia bihurtzen du. Hala ere, Dockerrekin integratu daiteke, beraz, ezaugarri guztiak kontuan hartuta, alternatiba ona da.

PodMan

PodMan kode-irekiko Linuxen tresna natibo bat da, edukiontzien sorkuntza, exekuzioa edo partekatzea errazteko diseinatua izan dena. Tresna honek CLI (*Command Line Interface*) bat hornitzen du era erraz batean erabiltzeko. CLIa Dockerrenaren antzekoa da, beraz, teknologia honenganako integrazioa handia da.

Beste kontainer motore teknologiek bezala, PodManek sistema eragilearekin elkarreragiten du edukiontziak sortu eta exekutatzeko. Ondorioz, PodManekin edo beste teknologiekin sortutako edukiontziak oso antzekoak dira, tresna honen eramangarritasuna handituz.

Beste teknologiekin duen desberdintasun nabariena aurkeztutakoa da, hau da, eskaintzen duen eramangarritasuna. Dockerren kasuan, adibidez, berezko zerbitzuak erabiltzen ditu edukiontzien kudeaketarako. PodManek, horiez gain, beste teknologien zerbitzuak erabil ditzake. Hala ere, beste softwareek baino aukera gutxiago eskaintzen ditu kudeaketa eta konfigurazio atalean, tresna sinpleagoa baita.

Buildah

Bestetik, Buildah proiektuak, beste aukerek bezala, edukiontziak kudeatzeko aukera eskaintzen du. Linuxen oinarritutako kode-irekiko tresna da, Open Container Iniciativekin (OCI) bateragarri diren edukiontziak diseinatzeko erabil daitekeena. Horrek, beste teknologiekin sortutako edukiontziekin integrazioa errazten du.

Tresna honek edukiontziak sortzeko atalean ardaztu da batez ere. Horregatik, sorkuntza prozesurako aukera desberdinak eskaintzen ditu. Esaterako, kontainerrak sortzeko kodea garatzeko hainbat lengoaia ahalbidetzen ditu.

Buildah eta PodMan tresnek integrazio ezin hobea daukate, biek kode zati bat konpartitzen baitute. Hala ere, puntu desberdinetan ardazten dira: PodMan eramangarritasunean

eta Buildah sorkuntza prozesuan. Beste teknologiekin konparatuz, tresna sinpleagoa izanez, proiektu honek dokumentazio eta komunitate txikiagoa du.

Taula 5.2: Mikrozerbitzuak sortzeko teknologien konparaketarako taula.

	Teknologiak			
Ezaugarriak	**Docker**	**RKT**	**PodMan**	**Buildah**
Kode irekia	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)
Komunitatea	Oso handia	Handia	Ertaina	Txikia
Konfigurazioa	Erraza eta aukera anitzekoa	Konplexua	Erraza	Erraza
Dokumentazioa	Oso handia	Ertaina	Handia	Txikia
Segurtasuna	Ertaina	Oso altua	Altua	Ertaina
Eramangarritasuna	Oso altua	Altua	Oso altua	Ertaina

Edukiontziak sortzeko eta kudeatzeko aukera guztien analisian lortutako ondorioak zehazteko, 5.2. Taula aurkeztu da. Lortutako ondorioak honakoak izan dira: Docker tresnarik ezagunena eta osoena da, RKT segurtasunean eta elkarreragingarritasunean ardazten da, PodMan eramangarritasun handia eskaintzen du eta Buildah edukiontziak sortzeko tresna independentea da.

Beraz, ezaugarri guztiak kontuan hartuz, aukera optimoena Docker dela ondorioztatu da, euskarri gehien duen teknologia baita. Gainera, eramangarritasun eta konfiguraziorako hainbeste aukera eskainiz, lan honen helbururako hautagai ezin hobea bihurtzen du. Hala ere, beste teknologiak baliagarriak dira, eta, nahi izanez gero, bakarrik edukiontzien sorkuntzarako prozesua aldatu beharko litzateke, aukera berria erabiliz.

5.3 Mikrozerbitzuen orkestraziorako aukeren analisia

Atal honetan, mikrozerbitzuen oinarri izango diren edukiontziak hedatzeko eta kudeatzeko beharrezkoak diren plataformen aukeren analisia beteko da. Aurreko atalean edukiontziak sortzeko teknologiak ikertu egin dira, beraz, analisi hau garatzerakoan, teknologia horiekin bateragarri badiren kontuan eduki da.

Bestalde, proiektu hau aurrera eramateko, erabiliko den plataforma konfiguragarria eta hedagarria izatea eskatzen du, beste ezinbesteko eskakizun bat izanez.

Docker Swarm

Docker Swarm edukiontziak kluster batean elkartzeko eta hauek exekutatzeko plataforma bat da. Docker edukiontziekin bateragarritasun ezin hobea du, konpainia berdinak garatutako bi proiektu baitira. Esan bezala, edukiontzien orkestratzailea izanda, hainbat *host* makinetan hedatutako kontainerrak kudeatzeko aukera eskaintzen du.

Docker teknologiarekin integrazioa ezin hobea izanda, instalazioa ez da oso konplexua. Hala ere, edukiontziak sortzeko beste teknologia bat erabili nahi izanez gero, hainbat arazo sor daitezke. Gainera, Docker Swarm ez dauka berezko GUI (Graphical User Interface) bat, tresna sinplea bihurtuz.

Kubernetes

Kubernetes lan kargak eta zerbitzuak kudeatzeko kode irekiko plataforma eramangarri eta hedagarria da. Edukiontzien hedapena eta exekuzioa errazten du, eta plataforma konfiguratzeko aukera anitz eskaintzen du. Google konpainiak 2014an aurkeztutako proiektu bat izanda, komunitate handia du. 2015ean Google Linux fundazioarekin batu zen CNCF fundazioa sortuz, prozesuan Kubernetes kode irekiko proiektu bezala bihurtuz.

Kubernetes plataformak, edukiontziak hedatzeaz gain, horien bizi-zikloa kudeatzen du, arazoren bat agertuz gero, errekuperazio ekintzak betetzeko, esaterako, kontainerra berriro berrabiarazi. Bestetik, kode irekiko edo hodei publikoko azpiegiturak aprobetxatzeko askatasuna eskaintzen du, plataforma hedatzeko aukera emanez. Aukera horien barruan, berezko azpiegiturak sortu eta integratu daitezke Kubernetesen.

Apache Mesos

Apache Mesos Linuxeko kernel edo nukleoan oinarritutako OpenSource kluster kudeatzaile bat da. Ez dauka Kubernetes edo Docker Swarmen tamaina edo garrantzia, baina eramangarritasun egokia du, hainbat sistema eragileetan exekuta baitaiteke.

Bestalde, edukiontziak sortzeko teknologiekin integra daiteke, hain zuzen ere, Docker eta RKTrekin. Potentzia eta zerbitzu aldetik aukera oso egokia da, eskuragarritasuna eta akatsekiko tolerantzia behar duten proiektuetarako, batez ere. Hala ere, euskarri komunitatea nahiko mugatua da, Kubernetesekin konparatuz, besteak beste.

Nomad

HashiCorp konpainiak garatutako kode irekiko proiektu hau edukiontziak orkestratzeko plataforma bat eskaintzen du. Malgutasunean eta sinpletasunean oinarritzen da, beraz, edukiontzi arinak erabiltzen dituen proiektuetarako aukera egokia da.

Edukiontziak osatutako aplikazioak heda daitezke, osagaiak ezabatuz edo gehituz. Hala ere, ingurune txikietarako prestatua izan da. Euskarri aldetik, komunitate aktiboa dauka, baina Kubernetes edo Apache Mesosekin konparatuta, txikiagoa dena.

Taula 5.3: Edukiontziak kudeatzeko eta hedatzeko teknologien konparaketarako taula.

	Teknologiak			
Ezaugarriak	**Docker Swarm**	**Kubernetes**	**Apache Mesos**	**Nomad**
Kode irekia	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)	OpenSource (ApacheLicense 2.0)	OpenSource (Mozilla Public License 2.0)
Komunitatea	Handia	Oso handia	Ertaina	Ertaina
Eskalagarritasuna	Txikia	Oso altua	Altua	Ertaina
Malgutasuna	Txikia	Oso altua	Altua	Txikia
Eramangarritasuna	Ertaina	Altua	Oso altua	Ertaina
Hedagarria	Ez	Bai	Bai	Ez

Aukera guztien analisia bete ondoren lortutako ondorioak aurkezteko, 5.3. Taula osotu da. Ondorioak honakoak dira: Docker Swarm Dockerrekin integrazio oso ona dauka, teknologiak sortzeko aukera egokiena dena, baina ez da hedagarria; Kubernetes aukera osoena eta komunitate handikoena da; Apache Mesos eramangarritasunean oinarritzen da eta Nomad malgutasunean eta sinpletasunean.

Puntu guztiak kontuan hartuz, proiektu honetarako aukera egokiena Kubernetes dela ondorioztatu da. Komunitatea eta malgutasun handieneko aukera da, eta plataforma konfiguratzeko eta hedatzeko hainbat aukera desberdin eskaintzen ditu. Batez ere, azkenengo ezaugarri honek dauka garrantzi handiena garatu nahi den lanerako. Gainera, [37] bezalako artikuluak argitaratu dira hautapen hau bermatzen dutenak.

6 Arriskuen analisia

Kapitulu honetan zehar proiektuaren garapenean sor daitezkeen arriskuen analisia beteko da. Analisia osoena izan dadin, arriskuak motaka aztertuko dira, hala nola arrisku ekonomikoak, teknikoak eta bestelako arriskuak. Arrisku bakoitzeko prebentzioa eta kontingentzia plana adierazita, 6.1. Taulan egoera guztien probabilitate-eragin erlazioa erakutsi da, bakoitza behar den tokian kokatuta.

Arrisku ekonomikoak

1. Klusterra eraikitzeko finantzaketarik ez edukitzea.
    - *Prebentzioa*: MALa zehazten den momentuan, behar den ekipamendua zehaztea.
    - *Kontingentzia plana*: Makina birtualak erabili klusterra eraikitzeko, dena ordenagailu bakarrean eginez.
2. Programaziorako iturri-kodea garatzeko programen lizentziarik ez edukitzea.
    - *Prebentzioa*: Ordaintzeko softwareez gain, kode-irekiko softwareen aukerak aztertuta edukitzea, eta sortutako proiektuak edozein softwareentzako balio izatea.
    - *Kontingentzia plana*: Sortutako proiektuak edozein softwareetan funtzionatzen badute, kode-irekiko aukera egokienera igaro.

Arrisku teknikoak

3. Klusterraren konfigurazio ez egokia burutzea.
    - *Prebentzioa*: Konfigurazioan bete beharreko pauso guztiak ondo zehaztea eta bete eta gero berrikustea.
    - *Kontingentzia plana*: Lehenik eta behin, akatsa non eman den ikertu, eta klusterra birkonfiguratu.
4. Node-RED plataformaren konfigurazio ez egokia egitea.
    - *Prebentzioa*: Konfigurazioan bete beharreko pauso guztiak ondo zehaztea eta berrikuspen egokia burutzea.

- *Kontingentzia plana*: Lehenik eta behin, akatsa non eman den ikertu, eta plataforma birkonfiguratu.

5. Hardware gailuren bat arazo batengatik funtzionatzen uztea.

   - *Prebentzioa*: Gailuen mantentze-lan egokiak betetzea. Esaterako, pizte eta itzaltze prozedurak ondo betearaztea.
   - *Kontingentzia plana*: Matxura bat gertatu bada, analisi bat garatzea. Arazoa zein izan den zehaztuta, konponketa edo errekuperazio ekintzak aurrera eramatea.

Bestelako arriskuak

6. Proiektuaren antolakuntza egokia ez izatea.

   - *Prebentzioa*: Lana hasi baino lehen antolakuntza aproposa betetzea eta tutoreekin adostea.
   - *Kontingentzia plana*: Proiektua doan puntutik berrantolatu epemuga kontuan hartuz.

7. Proiektuaren garapenean zehar aldaketak agertzea, bai tutoreek adierazita, bai teknologien mugek behartuta. Aldaketak software zein hardware atalean eman daitezke.

   - *Prebentzioa*: Lana garatzerakoan birkonfigurazioak gerta daitezkeela kontuan hartu, eta edozein momentuan erraz alda daitekeen garapena egitea.
   - *Kontingentzia plana*: Bete beharreko aldaketak lanari nola eragiten dion aztertzea eta, horretatik abiatuz, proiektua birplanifikatzea.

Taula 6.1: Arriskuen probabilitate-eragin erlaziorako taula.

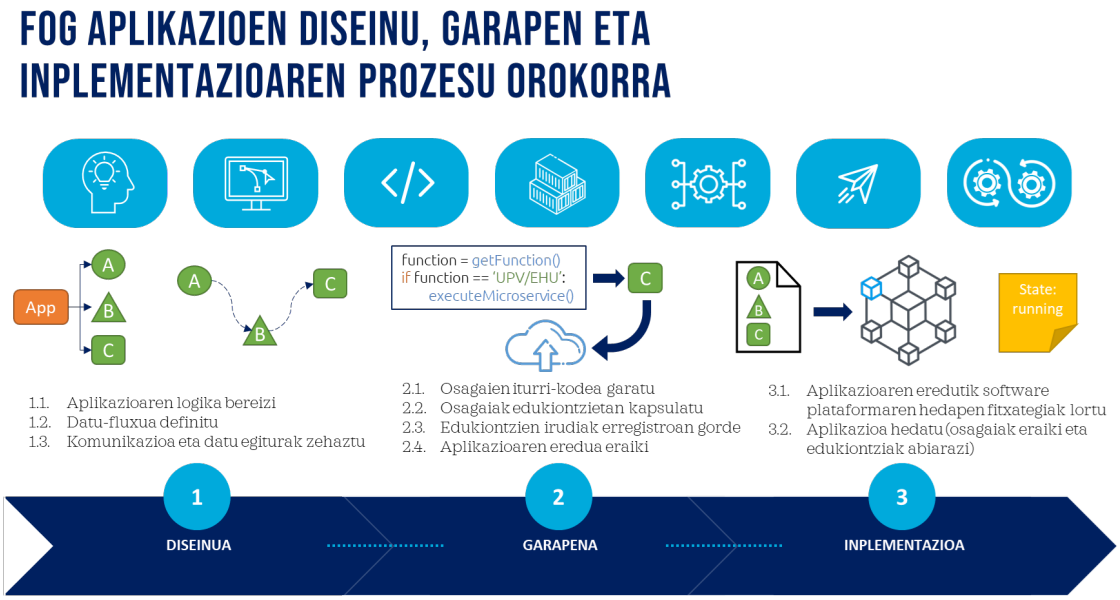
	Eragina				
Probabilitatea	**Oso txikia**	**Txikia**	**Ertaina**	**Altua**	**Oso altua**
Oso altua					
Altua				7	
Ertaina			1		
Baxua		2		4	3,6
Oso baxua					5

Atala II

GARAPENA

7 Soluzioaren deskribapena

Kapitulu honetan zehar lan honen ekarpena den soluzioaren deskribapen orokorra bete egingo da. MALaren helburu nagusia mikrozerbitzuetan oinarritutako Fog aplikazioen garapen eta operaziorako ikuspegi bat aurkeztea da eta, horretarako, prozesu horiek ondo ezagutu behar dira, baliabide egokiak hautatzeko, jarduerak zein diren eta nork bete behar dituen ondo definituz.



Irudia 7.1: Fog aplikazioen diseinu, garapen eta inplementazioaren deskribapen orokorra.

7.1. Irudian literaturan orokorrean proposatzen den Fog aplikazioen garapen-ziklo osoaren hiru faseak aurkeztu dira, ikuspegiaren atalekin bat datozenak. Lehenik eta behin, diseinu fasean aplikazioaren ideia zehaztu behar da (edo osagai zehatz baten ideia). Ondoren, ideia hori aurrera eramateko aplikazioaren logika ahal den heinean bereizi behar da. Horrela, aplikazioa hainbat osagaietan banatzen da, osagai bakoitzak logika osoaren zati bakar bat betetzen duena, hau da, funtzionalitate bakarra exekutatzen dute.

Osagaiak definituta eduki ondoren, beraien arteko elkarrekintza zehaztu behar da. Horretarako, datuen fluxua eta osagaien arteko komunikazioa nola gauzatuko den zehaztu behar da, datu egiturak eta komunikazio-protokoloak definituz.

Garapen fasean sartu aurretik, proposatutako soluzioarekin osagai berrerabilgarriak lortzen direnez, aurretik garatutako osagairen bat badagoen aztertu behar da. Osagai berriak garatu behar balira, bigarren faseko jarduerak jarraituko dira. Software programak izanik, iturri-kodearen eraikuntza atal garrantzitsua da. Birtualizazio teknologiak erabil-

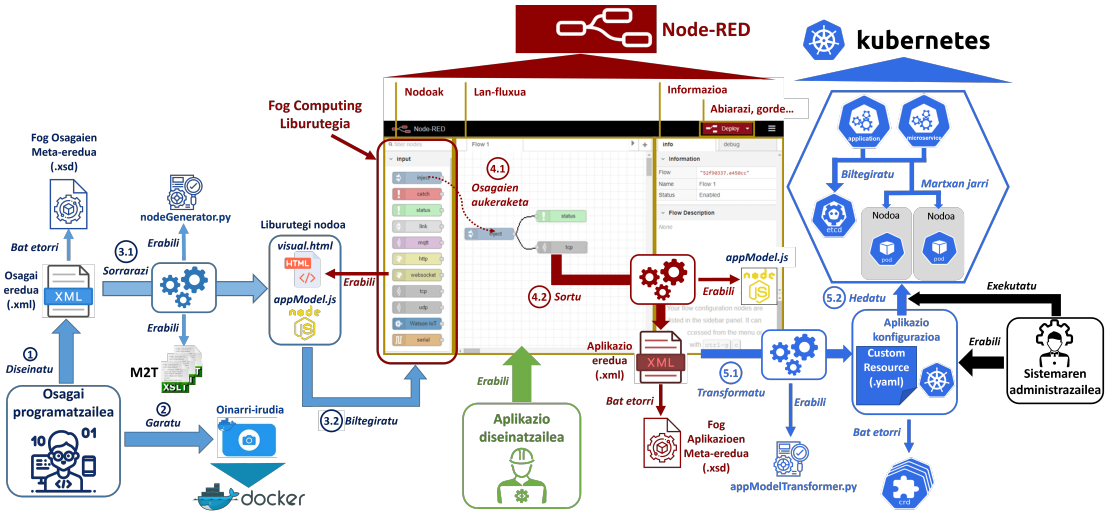
tzen direnez, kodea edozein programazio-lengoaian gara daiteke, baina kontuan hartuz funtzionalitate bat hautatuz, horri dagokion kode zatia exekutatu behar dela.

Osagaiaren funtzionalitateak exekutatzeko beharrezko baliabide guztiak Docker edukiontzi batean kapsulatu behar dira, eta horren emaitza den irudia erregistroan gorde, orkestrazio plataformarako eskuragarri egoteko. Azkenik, osagai guztien irudiak sortuta eta gordeta edukita, aplikazioaren eredua eraiki ahalko da, osagai bakoitzeko funtzionalitate bat aukeratuz. Hori lortzeko, ereduak automatikoki eraikitzeko ahalmena duen IDE bat erabili ahalko da.

Azkenik, inplementazio fasean sortutako aplikazioaren eredua erabiliko da, erabakitako plataformaren hedapen fitxategiak lortzeko. Transformazio hori lortutakoan, aplikazioa hedatu ahalko da, eta, software plataformak osagaien irudiak lortzeaz eta abiarazteaz arduratuko da, garapenean aukeratutako funtzionalitate zehatzak exekutatuz.

Aipatutako prozesua aplikazio berrietarako ematen da batez ere. Aurretik aipatu denez, aplikazioak behar dituen osagairen bat dagoeneko garatuta badago, berrerabil daiteke. Era berean, funtzionalitate berriak behar baditugu, eta lehendik dagoen osagai bat harekin erlazionatuta badago, osagaia egunera dezakegu, hutsetik sortu beharrean. Osagai guztiak jada existitzen badira, aplikazioa zuzenki diseinatu daiteke, osagai bakoitzeko funtzionalitateak aukeratuz, eta IDEaren laguntzaz aplikazio-eredua automatikoki eraikiko da.

7.2. Irudiak master amaierako lan honetan garatutako ikuspegirako soluzioa grafikoki aurkezten du, prozesuaren atal bakoitzean parte hartzen dituzten teknologiak zehaztuz. Literaturan proposatzen den metodologiaren moldaketa bat da, bete beharrezko zereginek ez baitute elkarrengandik hainbesteko menpekotasunik. 7.1. Irudiko prozesuarekin konparatuz Fog aplikazioen garapen-zikloaz gainera, erlazionatutako, baina beste alde batetik betetzen den Fog osagaien garapen-zikloa proposatzen da proiektu honetan.



Irudia 7.2: Proposatutako ikuspegiaren deskribapen orokorra, parte hartzen duten teknologiak barne.

Proposatutako kasuan, aplikazioaren diseinua ez da zertan lehengo jarduera izan behar. Fog osagaiak era independentean diseina eta gara daitezke, horren ardura osagai programatzaile deitutako pertsonak hartuko duena. Horrela, osagai berriak sor ditzake, edo jada existitzen diren osagaiak heda ditzake, funtzionalitate edo komunikazio ahalmen berriak gehituz.

Fog osagai horiek aplikazio diseinatzailea deituriko pertsonarentzako eskuragarri egongo dira. Honek, garapen-ingurune integratua izango den plataforma erabilita, Fog osagaiak lortu, pertsonalizatu eta Fog aplikazioak sor ditzake, aurretik aipatutako aplikazio-ereduak sortuz.

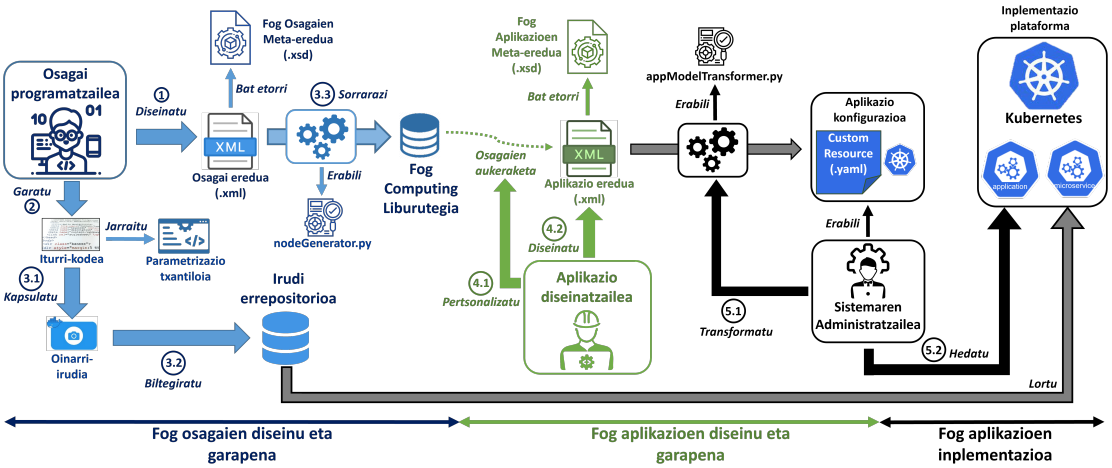
Azkenik, aplikazio-eredu horiek sistemaren administratzailea deritzon pertsonari helduko zaizkio. Honen ardura, ereduak inplementazio edo operazio plataformaren hedapen fitxategietan bihurtzea da. Fitxategi horiekin, Fog aplikazio software plataforma heda eta abiarazi ahalko da, osatzen dituzten mikrozerbitzuak sortuz eta exekuzio fasean sartuz.

Hurrengo kapituluetan lan honetan proposatzen soluzioa den ikuspegia sakonki aurkeztuko da. Jarduera bakoitzeko, parte hartuko dituzten teknologiak azalduko dira, eta jarraitutako prozedurak garapen zuzena aurrera eramateko.

8 Metodologiaren ikuspegi orokorra

Atal honetan zehar proposatutako metodologiaren ikuspegi orokorra aztertuko da. Esaterako, aurretik komentatu denez, Fog aplikazioak mikrozerbitzuez osatutako lan-fluxu zuzendu bezala kontuan hartzen dira. Mikrozerbitzuak inplementatzeko soluziorik erabiliena edukiontziak dira.

Bestetik, metodologiak MDE teknikak erabiltzen ditu eta bere helburu nagusia osagaien eta aplikazioen diseinu eta garapenaren banaketa lortzea da. Zehazki, lan honek XML teknologien erabilera proposatzen du, hain zuzen ere, XML fitxategiak (.xml luzapena dituztenak) ereduak sortzeko eta W3C XML Schema teknologia [38] (.xsd luzapeneko fitxategiak) meta-ereduak inplementatzeko. M2T transformazioetarako XSLT (eXtensible Stylesheet Language Transformations) estilo-orrien erabilera proposatzen da, XML dokumentuak beste formatuetan bihurtzeko aukera eskaintzen dituztenak. Teknologia horiekin, lortutako eredua generikoak izatea lortzen da, hau da, inplementazio-plataformatik menpekotasunik ez edukitzea.



Irudia 8.1: Proiektuan proposatutako metodologiaren ikuspegi orokorra.

8.1. Irudiak aipatutako ikuspegi orokorra aurkezten du. Aztertu daitekeenez, aurreko atalean komentatutako hiru interes-talde bereizi dira: osagai programatzaileak, aplikazio diseinatzaileak eta sistemaren administratzaileak (bakoitzaren jarduerak urdinez, berdez eta beltzez erakutsi direnak, hurrenez hurren).

Beraien artean interes-banaketa lortzeko bi kontzeptu definitu dira, Fog Osagaia eta Fog Aplikazioa, Component Based Software Engineering (CBSE) ikuspegian [39] oinarritzen direnak. CBSEk sistema konplexuen eraikuntza proposatzen du (Fog aplikazioak), osagai

sinpleen konposizioaren bidez (Fog osagaiak), aldez aurretik era independentean garatuak izan direnak.

CBSE ikuspegiak osagaien garapena sustatzen du aplikazio ezberdinetan berrerabil daitezkeen software-modulu gisa. Hori dela eta, lan honetan Fog osagaiak modulu horiek izango dira, aurretik osagaien programatzaileak aplikazioaren diseinutik era independentean sortutakoak. (8.1. irudiko 2. urratsa). Osagai hauek aplikazio-domeinuan identifikatzen diren zenbait funtzionalitate biltzen dituzte.

Ikuspegi hau jarraituz, Fog aplikazioaren diseinatzaileek funtzionaltasun konplexuak osatzeko erlazionatu daitezkeen kutxa beltz gisa ulertuko dituzte Fog osagaiak. Horrela, aplikazioaren diseinu-fasean erabiltzeko, ez da beharrezkoa haien barne xehetasunen berri izatea, edo lotutako teknologien ezagutza edukitzea; nahikoa da haien interfazea ezagutzea.

OpenFog erreferentzia-arkitekturak eskaintzen duen Fog aplikazioen definizioa betetzeko, Fog aplikazioak Fog osagaien instantziaz osatuta daudela proposatzen da, zeinen funtzionalitateetako bakarra hautatu den, hau da, mikrozerbitzuez osatuta. Beraz, Fog osagaiaren instantzia mikrozerbitzu bati dagokio, eta aukeratutako funtzionalitatea eskaintzen duen zerbitzua izango da.

Aurretik aipatutako deskribapen orokorrean bete beharreko jarduerak zehaztu dira. 8.1. Irudian jarduera horien arduradunak aurkeztu dira. Aztertu daitekeen bezala, kasu honetan osagaien eta aplikazioen prozesuak bananduta daude. Horregatik, Fog aplikazioak sortzeko prozesua ez da hain lineala. Aplikazio-diseinatzaileek osagai-programatzaileek sortutako osagaiak erabiltzen dituzte aplikazioa eraikitzeko. Osagai berriren bat beharrez gero, programatzaileari eskatu beharko dio. Betiere beraien arteko interes-banaketa eskaintzea da lan honen helburua, hau da, bakoitzak bere jarduerei buruzko ezagutza soilik izan behar duela.

9 Ereduetan oinarritutako Fog aplikazioen diseinu eta garapena

Edukia

Aurkeztutako proposamenak Fog osagai eta aplikazio berrerabilgarrien diseinu eta garapen zuzena ziurtatu behar du. Osagaiak aplikazioetatik modu independentean garatzen badira ere, ez dago benetako interes-banaketarik osagai-programatzaileen eta aplikazio-diseinatzaileen artean, osagaien garapenak aplikazioen garapenarekin harreman zuzena eta errepikakorra dakarrelako, ezinbestekoa izanik merkatuko beharrak jakitea.

Aipatutako taldeen arteko interes-banaketa lortu nahi denez, proposatutako metodologiak bi fase bereizten ditu: Fog osagaien diseinua eta garapena eta Fog aplikazioen diseinua eta garapena, prozesuaren fase honetan parte hartzen duten osagai-programatzaile eta aplikazio-diseinatzaileak kontuan harturik. Hain zuzen ere, aipatutako banaketa lortzeko, bakoitzak era independentean lan egingo du. Horregatik, hurrengo ataletan osagai-programatzaileek eta aplikazio-diseinatzaileek, beren zereginak bete ditzaten, bete behar dituzten jarraibideak eta mekanismoak zehazten dira, bakoitzarentzat desberdinak izango direnak.

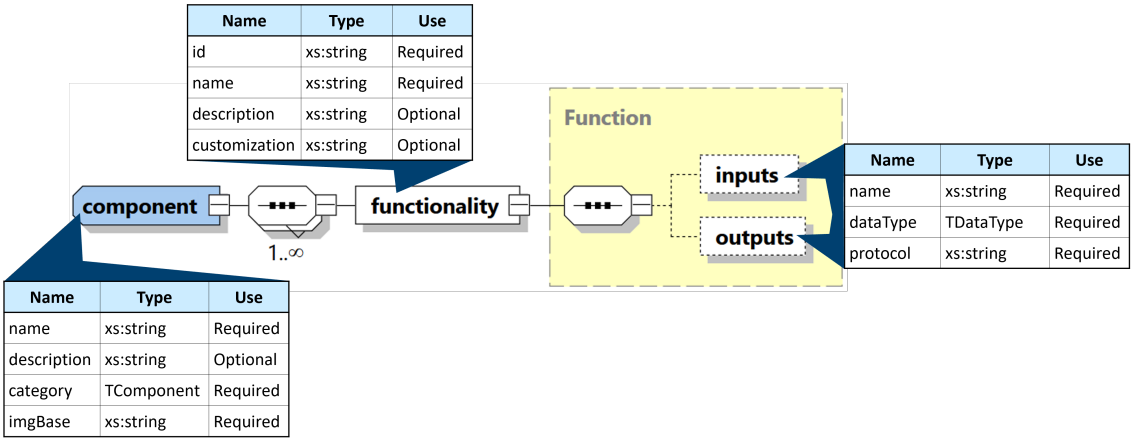
9.1 Fog osagaien diseinu eta garapena

Esan bezala, proposatutako metodologiak Fog osagaien zehaztapen abstraktua eskaintzen du (aplikaziotik independentea). Gainera, erabilitako teknologiei esker, zehaztapen generikoa (domeinutik independentea) lortzen da. Hori lortzeko, aurkeztutako zehaztapena Fog osagaiaren meta-ereduan oinarritzen da, Fog osagaien ezaugarriak biltzen dituena.

Fog osagaiak inplementatzeko Fog aplikazioen barruan egon behar direnez, hurrengo bi azpiataletan Fog osagaiak diseinatzeko eta garatzeko urratsak eta erabili beharreko baliabideak aztertuko dira. Azkenik, osagaiak sortutakoan beraien biltegiratze-prozesua aztertzeko azkenengo azpiatal bat aurkeztu da.

9.1.1 Fog osagaien diseinua

Fog osagaiak diseinatzeko lehen urrats gisa, osagai-programatzaileak osagai eredua osatu beharko du, osagai bakoitzeko. Eredu hori XML fitxategi batean garatuko da eta bertan Fog osagaiak kapsulatuko dituen funtzionalitateak ezartzen dira. Eredu honen zuzentasuna bermatzeko, Fog osagaien meta-ereduan (ikus 9.1. Irudia) zehaztutako arauak jarraitu behar ditu, XML Schema batean inplementatuta dagoena (XSD motako fitxategi batean).



Irudia 9.1: Proposatutako Fog osagaien meta-eredua XML Schema bezala inplementatuta.

Osagai bakoitzak (*component*) sisteman bakarra den izen (*name*) eta hautazko deskribapen zehatz bat (*description*) izango du. Mota edo kategoria (*category*) atributuak osagaien taldekatzea ahalbidetzen du, programatzailearen irizpidearen arabera zehazten dena eta aurretik definitutako datu-mota zerrendatu batetik lortzen dena. Azkenik, osagaiaren inplementazioarekin erlazionatutako atributu bat zehaztu da, oinarri-irudia (*imgBase*).

Berrerabilpena betetzeko, osagaiek hainbat funtzionalitate eskain ditzakete (*functionality*), identifikadore bakar batez (*id*) eta izen batez (*name*) bereizita daudenak. Gainera, deskribapen xehe bat (*description*) gehitu daiteke eta exekuzio-aldagaiak zehaztu *customization* atalean. Azkenengo aldagai horiek programatzaileari funtzionalitateekin erlazionatutako datuak zehazteko aukera eskaintzeko gehitu dira, betiere *CUSTOM_<aldagai izena>* formatua edukiko dutenak.

Bestalde, funtzionalitateek sarrera- eta/edo irteera-parametroak dituzte, *inputs* eta *outputs* elementuek irudikatuta. Horiek datu-egiturak adierazten dituzte, *dataType* atributuaren bitartez, programatzaileek aurretik definitutako *TDataType* mota enumeratu bat dena. Horiekin batera, datuak zein protokoloren (*protocol*) bitartez bidaltzen diren zehazten da.

Aipatu denez, programatzaileak meta-eredua erabiliko du Fog osagaiaren eredua osatzeko. Kontzeptu-proba bezala 9.1. XML fitxategiak diseinu ataleko emaitza den Fog osagai baten eredua erakusten du, zenbakiak prozesatzeko balio duena. Prozesatze osagaien

kategoriari dagokio eta hiru funtzionalitate eskaintzen ditu, balioa handitzeko, txikitzeko edo biderkatzeko. Bakoitzak balioekin lan egiteko datu-egitura egokia jasotzen du, eta emaitza HTTP protokoloaren bidez bidaltzen du.

Kodea 9.1: *ZenbakienProzesamendua* Fog osagaiaren eredua XML dokumentu bezala

```
<?xml version="1.1" encoding="UTF-8"?>
<component name="ZenbakienProzesamendua" category="prozesatze"
      description="Osagai honek zenbaki-prozesatze desberdinak burutzen
          ditu." imgBase="ekhurtado/gcis:zenbakien-prozesamendua">
  <functionality id="BalioaHanditu" name="Balioa handitu"
        description="Jasotako balioa aukeratutako urratsarekin gehitzen
            du." customization="custom_urratsa">
    <inputs name="JasotakoBalioa" dataType="TZenbaki" protocol="HTTP"/>
    <outputs name="BalioEmaitza" dataType="TZenbaki" protocol="HTTP"/>
  </functionality>
  <functionality id="BalioaTxikitu" name="Balioa txikitu"
        description="Jasotako balioa aukeratutako urratsarekin kentzen
            du." customization="custom_urratsa">
    <inputs name="JasotakoBalioa" dataType="TZenbaki" protocol="HTTP"/>
    <outputs name="BalioEmaitza" dataType="TZenbaki" protocol="HTTP"/>
  </functionality>
  <functionality id="BalioaBiderkatu" name="Balioa biderkatu"
        description="Jasotako balioa aukeratutako zenbakiarekin
            biderkatzen du." customization="custom_biderkatzailea">
    <inputs name="JasotakoBalioa" dataType="TZenbaki" protocol="HTTP"/>
    <outputs name="BalioEmaitza" dataType="TZenbaki" protocol="HTTP"/>
  </functionality>
</component>
```

Fog osagaien diseinu atala bukatzeko, programatzaileak ea osagaiaren eredua meta-ereduarekin bat datorren egiaztatuko du. Ereduen, meta-ereduen eta XML analizatzaile baten erabilerak diseinatutako osagaiak zuzenak direla eta behar bezala gara daitezkeela ziurtatzen du.

9.1.2 Fog osagaien garapena

Fog osagaien garapenean bete behar duten funtzioa era zuzen batean exekutatzeko beharrezkoak diren pausoak eta erabili beharreko baliabideak zehazten dira. Mota honetako osagai bat garatzeko lehen urratsa betiere iturri-kodea sortzea da. Kode horrek hainbat funtzionalitatetik bakar bat exekutatzeko ahalmena izan behar du, berarekin erlazionatuta dauden sarrera eta/edo irteerak kontuan hartuz.

Aipatu beharra dago Fog osagaien garapena beraien inplementazioarekin oso lotuta dagoen prozesu bat dela. Fog osagaien inplementaziorako birtualizazio teknologiak erabiltzea proposatzen da lan honetan. 5. kapituluan aztertu denez, horretarako teknologia egokiena Docker da. Hori dela eta, iturri-kodearen garapenerako inplementazio tresna kontuan hartu behar da.

Beraz, diseinu-fasean proposatutako Fog osagaien meta-ereduan zehaztutako funtzionalitate guztiak exekutatzeko logika izateaz gain, erabiltzaileak zehaztutako funtzionalitatea zehatzaren aukeraketa lortzeko gai izan behar da, osagaiaren exekuzio-fasea baino lehen.

Horregatik, lan honek Fog osagaiak parametrizazio-txantiloi bat jarraituz garatzea proposatzen du (8.1. irudiko 2. urratsa). Inplementazio teknologia bezala Docker hautatuta, Fog osagai bakoitza edukiontzi irudi batean kapsulatzea proposatzen da. Funtzionalitate guztiak barne edukiko dituen irudi horri *oinarri-irudia* (ingelesez *base image*) deitu diogu. Hain zuzen ere, Fog osagaien meta-ereduko *imgBase* atributua elementu honi egiten dio erreferentzia.

Bestetik, aipatu beharra dago Docker edukiontzia berehala abiarazten dela, sistema eragilea abiarazi beharrik gabe, birtualizazioa baita sistema eragile gonbidatua sistema eragilearen gainean eskaintzen duena. Beraz, edukiontzi baten barruan exekutatu beharreko funtzionalitatearen parametrizaziorako modurik egokiena ingurune-aldagaiak erabiltzea da, horiek dinamikoki izendatutako balioak baitira, prozesuak exekutatzeko moduan eragin ditzaketena. Horrela, kodearen exekuzioaren pertsonalizazioa bete daiteke Fog aplikazioen diseinu fasean.

Aldagai horiek bi taldetan bana daitezke. Batetik, osagaia Fog aplikazioan integrazioarekin erlazionatutako ingurune-aldagaiak, hala nola *SERVICE*, aukeratutako funtzionalitatea zehazteko eta horrekin erlazionatutako "*customization*" exekuzio-aldagaiak. Bestetik, mikrozerbitzuen lan-fluxuarekin erlazionatutako aldagaiak, esaterako, *INPORT_NUMBER* osagaiak besteekin komunikatzeko entzuten egongo den portu zenbakia adierazteko.

Docker erabilita iturri-kodea edozein programazio-lengoaian gara daiteke, osagai-programatzaileari askatasuna eskainiz. Adibide gisa, 9.2. Kodean *ZenbakienProzesamendua* osagaiaren kodearen zati bat erakutsi da. Ikus daitekeenez, osagai-ereduan zehaztutako aldagaiak lortzen dira, eta aplikazio-diseinatzailearen diseinu-faseko erabakiaren arabera, aukeratutako funtzionalitatearekin erlazionatutako kodea exekutatzen da.

Kodea 9.2: *ZenbakienProzesamendua* Fog osagaiaren iturri-kodearen zati bat, [40]-an eskuragarri dagoena

```
import os, requests

function = os.environ.get('SERVICE')
inPortNumber = os.environ.get('INPORT_NUMBER')
custom_step = os.environ.get('CUSTOM_URRATSA')
custom_multiplier = os.environ.get('CUSTOM_BIDERKATZAILEA')

def main():
  match function:
    case "BalioaHanditu":
      increaseValue()
    case "BalioaTxikitu":
      decreaseValue()
    case "BalioaBiderkatu":
      multiplyValue()
    case _:
      pass
```

9.1.3 Diseinatu eta garatutako Fog osagaien biltegiratzea

Esan bezala, Fog aplikazioak sortzeko prozesuan, iturri-kodea amaitu eta gero, Docker edukiontziaren oinarri-irudia sortu beharko da. Bertan, kodearen fitxategia eta bere exekuziorako beharrezkoak diren baliabideak kapsulatu beharko dira.

Docker irudiak sortzeko fitxategiari *Dockerfile* deritzo. 9.3-ean aztertutako adibidearen *Dockerfile* fitxategia aurkezten da. Python lengoaian garatutako kodea izanez, horren oinarri-irudia hautatu da. Kodearen fitxategia edukiontzira kopiatu da eta beharrezko paketeak instalatu dira: *Flask* HTTP mezuen harrerak kudeatzeko eta *requests* HTTP mezuen bidalketak ahalbidetzeko. Azkenik, kodea exekutatzeko komandoa zehazten da.

Kodea 9.3: *ZenbakienProzesamendua* Fog osagaiaren oinarri-irudiaren *Dockerfile* fitxategia

```
1 FROM python:3.10-alpine
2 RUN pip install Flask
3 RUN pip install requests
4 COPY zenbakienProzesamendua.py /
5 CMD ["python3", "-u", "zenbakienProzesamendua.py"]
```

Programatzaileak oinarri-irudia sortzeko komando-lerroko interfazean hurrengo komandoetatik lehenengoa exekutatu beharko du. Ondo sortu dela egiaztatzeko, bigarren komandoa erabili dezake, sortu diren irudi guztiak ikusteko. Ondoren, osagai-programatzaileak oinarri-irudi horiek irudi-errepositorio batean gorde behar ditu, Fog aplikazioen inplementazio faseetan eskuragarri egoteko (8.1. Irudiko 3.1. urratsa). Horretarako, hirugarren komandoa exekuta dezake, nahi duen irudia errepositoriora igotzeko. Kontuan eduki behar da igoera hori betetzeko baimenak eduki behar direla.

```
> docker build -f <Dockerfile fitxategia> -t <oinarri-irudiaren etiketa> .
> docker images
> docker push <oinarri-irudiaren etiketa>
```

Osagai-eredua inplementaziorako informazioarekin eguneratu eta gero, osagaien liburutegi batean gorde beharko du, aplikazioaren diseinu fasean erabili ahal izateko (8.1. Irudiko 3.2. urratsa). Adibidez, osagai-ereduan *imgBase* atributuan sortutako oinarri-irudiaren etiketa sartu beharko du. Fog osagaien liburutegiari Fog Computing Liburutegia deituko diogu.

9.2 Fog aplikazioen diseinu eta garapena

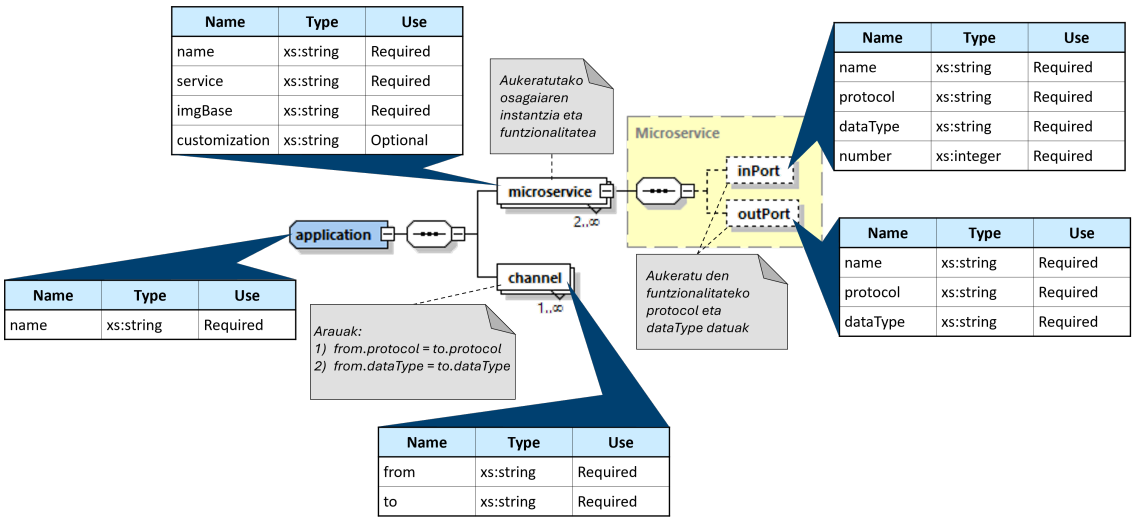
Fog osagaiak garatuta eta Fog Computing Liburutegian biltegiratuta egonda, Fog aplikazioen diseinu eta garapena ahalbidetzen da. Osagaien kasuan bezala, meta-ereduen erabilera funtsezkoa da, aplikazio-diseinatzailearentzako jarraibideak zehazten baitituzte.

Ezarritako ikuspegiarekin Fog osagaiak Fog aplikazio batean integratzeko eta zuzenean abiarazteko prest daudenez, kasu honetan, aplikazioaren diseinu eta garapen faseak batu daitezke. Bi fase horien helburua aplikazio-eredua garatzea izango da.

Fog osagaien garapenean beraien inplementazio fasea kontuan izan da. Proposatutako ikuspegiak Fog aplikazio-eredu generiko bat proposatzen duenez, aplikazioen kasuan inplementazioa kanpoan utz daiteke, horrela, aplikazio-diseinatzaile eta sistemaren administratzailearen arteko interes-banaketa lortuz. Diseinatzaileak aplikazio-eredua lortuko du eta administratzaileari emango dio. Honek, software-plataforma hedatzeko beharrezko transformazioak beteko ditu.

Horrela, aplikazioen diseinurako, kasu bakoitzerako, osatzen duten mikrozerbitzuak ezarri behar ditu. Mikrozerbitzuak zehazteko, Fog osagaiak instantziatzen ditu, Fog Computing Liburutegitik lortzen direnak, eskaintzen dituzten funtzionaltasunen artean beharrezkoa aukeratuz. Ondoren, lan-fluxua antolatzen du, datu-fluxua definitzen duten mikrozerbitzuen arteko erlazioak ezarriz.

Prozesu honen zuzentasuna bermatzeko, Fog aplikazioen meta-eredu bat ere zehaztu da, XML Scheman inplementatuta (ikus 9.2. Irudia). Aplikazio bakoitzerako, meta-eredu honek sistema bakarra den izen bat (*name*) eta bi mikrozerbitzu edo gehiagoz osatuta dagoela zehazten du. Mikrozerbitzu bakoitza, 9.1.2. azpiatalean deskribatzen den bezala, diseinatutako Fog osagaien instantziari dagokio. Beraz, datuak bertatik lortzen dira, hala nola mikrozerbitzuaren izena (*name*) instantziatutako Fog osagaien izenetik, eskaintzen duen zerbitzua (*service*) eta honekin erlazionatutako exekuzio-aldagaiak (*customization*) hautatutako funtzioaren identifikatzailetik, eta oinarri-irudia (*imgBase*) osagaiaren eredutik heredatzen da.



Irudia 9.2: Proposatutako Fog aplikazioen meta-eredua XML Schema bezala inplementatuta.

Hautatutako Fog osagaiak eta bere funtzionalitateak mikrozerbitzuaren komunikazioa zehazten du, sarrera eta/edo irteerak baititu (*inPort* eta *outPort*). Bakoitzak bere izen,

erabilitako protokolo eta datu-motarekin identifikatzen dira. Sarrerako atakak eskaerak entzungo dituen ataka-zenbakiarekin ere bereizten da (*number*).

Bestetik, mikrozerbitzuen arteko datu-fluxua kanalen (*channel*) bidez egituratzen da. Horiek bi atributu dituzte, jatorria (*from*) eta helmuga (*to*) zehazteko. Horrela, aplikazioaren fluxuaren hasiera eta amaiera sarrera ataka edo irteera ataka bakarrik dituzten mikrozerbitzuek mugatuko dituzte.

Aplikazioen diseinua bete ondoren, hau da, mikrozerbitzuen eta beraien arteko erlazioen zehaztapena osotuz, aplikazioaren garapena burutuko da. Garapen fasean, zehaztutako informazio guztia lortuko da, eta, proposatutako meta-eredua jarraituz, aplikazio-eredua eraikiko da.

Kontzeptu-proba bezala, 9.4. Kodeak hiru mikrozerbitzuz osatutako Fog aplikazio baten eredua aurkezten du. Hasierako mikrozerbitzuak (sarrera atakarik ez duenak) ausazko zenbakiak sortzen ditu eta zenbaki horiek bitan handitzen duen mikrozerbitzu batera bidaltzen ditu (sarrera eta irteera atakak dituena). Ondoren, emaitza azkenengo mikrozerbitzu batera bidaltzen da (irteera atakarik ez duena), balio finala pantailan erakusteko. Guztiak HTTP protokoloaren bitartez komunikatzen dira.

Kodea 9.4: *ZenbakienAplikazioa* Fog aplikazioaren eredua XML dokumentu bezala

```
<?xml version="1.0" encoding="UTF-8"?>
<application name="ZenbakienAplikazioa">
  <microservice name="ZenbakienSorkuntza" service="BalioNaturalak"
        imgBase="ekhurtado/gcis:zenbakien-sorkuntza"
        customization="{'custom_mota': 'ausazkoa', '
          custom_hasierakobalioa': 5}">
    <outPort name="ZenbakienSorkuntzaOPort"
        protocol="HTTP" dataType="TZenbaki"/>
  </microservice>
  <microservice name="ZenbakienProzesamendua" service="BalioaHanditu"
        imgBase="ekhurtado/gcis:zenbakien-prozesamendua"
        customization="{'custom_urratsa': 2}">
    <inPort name="ZenbakienProzesamenduaIPort"
        protocol="HTTP" dataType="TZenbaki" number="6000"/>
    <outPort name="ZenbakienProzesamenduaOPort"
        protocol="HTTP" dataType="TZenbaki"/>
  </microservice>
  <microservice name="ZenbakienAurkezpena" service="PantailaAurkezpen"
        imgBase="ekhurtado/gcis:zenbakien-aurkezpena">
    <inPort name="ZenbakienAurkezpenaIPort"
        protocol="HTTP" dataType="TZenbaki" number="7000"/>
  </microservice>

  <channel from="ZenbakienSorkuntzaOPort" to="
      ZenbakienProzesamenduaIPort"/>
  <channel from="ZenbakienProzesamenduaOPort" to="
      ZenbakienAurkezpenaIPort"/>
</application>
```

Aplikazio-eredua edukita, azkenik, aplikazio-diseinatzaileak egiaztatu beharko du eredua proposatutako Fog aplikazioen meta-ereduarekin bat datorrela. Beraz, bi meta-ereduen

erabilera konbinatuak aplikazioen definizio zuzena bermatzen du, ezarritako lan-fluxuaren koherentzia Fog osagaien interfaze ezaugarrien arabera egiaztatzea ahalbidetzen baitu.

10 Ereduetan oinarritutako garapen-ingurune integratua

Edukia

Aurreko atalean aztertutako prozesua era zuzen batean gauzatzeko, zenbait prozesu automatizatu behar dira, erabiltzaile bakoitzak jarduera desberdin asko bete behar baitituzte. Jarduerak automatizatzeko eta programatzaile zein diseinatzaileei laguntza eskaintzeko, proiektu honetan garapen-ingurune integratu edo IDE bat proposatzen da.

Osagai-programatzaileek eta aplikazio-diseinatzaileek bananduta lan egiten dutenez, lehenik eta behin proiektu honek osagaientzako eskaintzen dituen baliabideak aztertuko dira. Ondoren, IDEan integratutako teknologiak eta nola erabiltzen diren azalduko da.

10.1 Fog osagaien sorkuntzarako baliabideak

Lan honen helburuetako bat Fog osagai-programatzaileen eta aplikazio-diseinatzaileen arteko interes-banaketa lortzea dela-eta, bakoitzak bere garapen-ingurunea edukiko du. 9.1. atalean komentatu denez, Fog osagaiak osatzeko Docker birtualizazio teknologia erabilita, iturri-kodea garatzeko edozein programazio-lengoaia erabili daiteke.

Fog osagaien ezaugarriak kontuan izan behar dira (adibidez, hainbat funtzionalitatetik bakarra exekutatu beharra), baina programatzaile bakoitzak bere tresna erabili ahalko du iturri-kodea garatzeko eta oinarri-irudia osatzeko. Azkenengo honetarako Docker teknologia erabiltzea beharrezkoa da.

Horrela, programatzaileak osagaiaren oinarri-irudia sortzeko askatasuna du, baina beti Fog osagaien meta-eredua kontuan edukiz eta eskainitako txantiloiak erabiliz. Irudia sortuta eta errepositorioan gordeta egonda, osagaiaren-eredua garatu beharko du.

Programatzaileak osagai-eredua XML fitxategia batean gorde eta gero, Fog osagaien meta-ereduarekin egiaztapen automatikorako programa bat garatu da. Osagai-eredua eskainita, ea meta-ereduarekin bat datorren bueltatuko du. Osagai-programatzailearentzako baliabide guzti hauek proiektu honen errepositorioan [40] daude eskuragarri.

10.2 Fog Computing Liburutegiaren sorkuntza automatikoa

Fog aplikazioen diseinua eta garapena errazteko eta aipatutako interes-banaketa lortzeko helburuarekin, lan honek Node-RED plataformaren hedapen bat proposatzen du, programazio eta diseinu tresna bisual gisa. Fog aplikazioen lan-fluxua zehazteko, aplikazio-diseinatzaileek Fog osagaiak eskuragarri eta ondo antolatuta eduki behar ditu.

Node-RED tresnaren abantaila bat txantiloietatik nodo pertsonalizatuak garatzeko ahalmenean datza. Lan honek ezaugarri hori aprobetxatzea proposatzen du, Fog Computing Liburutegia Node-RED barruan integratuz, Fog osagaiak meta-ereduarekin bat datozen nodo gisa gorde daitezen.

Nodo horiek automatikoki sortu ahal izango dira, M2T transformazioak erabiliz. Horrela, osagai-ereduko informazioarekin, Node-RED zuzenean integra daitezkeen nodoak sor daitezke. Gainera, nodoak zeregin pertsonalizatuak exekutatu ditzakete, hala nola, Fog aplikazioaren eredua automatikoki sortzea.

Fog aplikazioak grafikoki diseinatzeko, Fog Computing Liburutegiko osagaiek irudikapen grafikoa izan behar dute. Bestalde, aplikazioak garatzeko behar den informazio zati bat osagai-ereduetan dago, eta osagaien instantziazioen araberakoa da. Horregatik, lan honek liburutegiko nodo bakoitzak bi zatiz osatuta egotea proposatzen du: 1) liburutegiko nodoaren edukia bistaratzeko aukera ematen duen atal bisual bat; eta 2) instantziatutako Fog osagaiari dagokion aplikazio-ereduaren zatia automatikoki sortzen duen zati funtzional bat.

Node-RED nodo-fluxuetan oinarrituta egonez, ikuspegi honekin aplikazio-eredua automatikoki sor daiteke, osagai bakoitzak bakarrik bere informazioa gehituz eta hurrengo osagaiari eredu eguneratua bidaliz.

Fog Computing Liburutegiaren sorkuntza automatikorako, kontuan hartu da Node-REDen exekuzio-denbora Node.js-en eraikita dagoela [41]. Izan ere, liburutegiko nodoak Node.js dira eta bertan JavaScript kodea idatz daiteke atal funtzionala osatzeko. Bestalde, Node-RED web-editore bat eskaintzen du, beraz, atal bisuala HTML lengoaian zehaztuko da.

Atal bisualerako *visual.html* fitxategia sortzen da. Nodoaren irudikapen bisualarekin erlazionatutako informazioa (osagaiaren web-ikuspegia) dauka, bai liburutegian (kolorea, izena, nodoaren sarrera edo irteerak, etab.), bai behin instantziatuta (aukeratutako funtzionalitatea, etab.). Horrez gain, erabiltzaileak parametrizatu beharreko aldagaien esleipen elementuak ezartzen ditu.

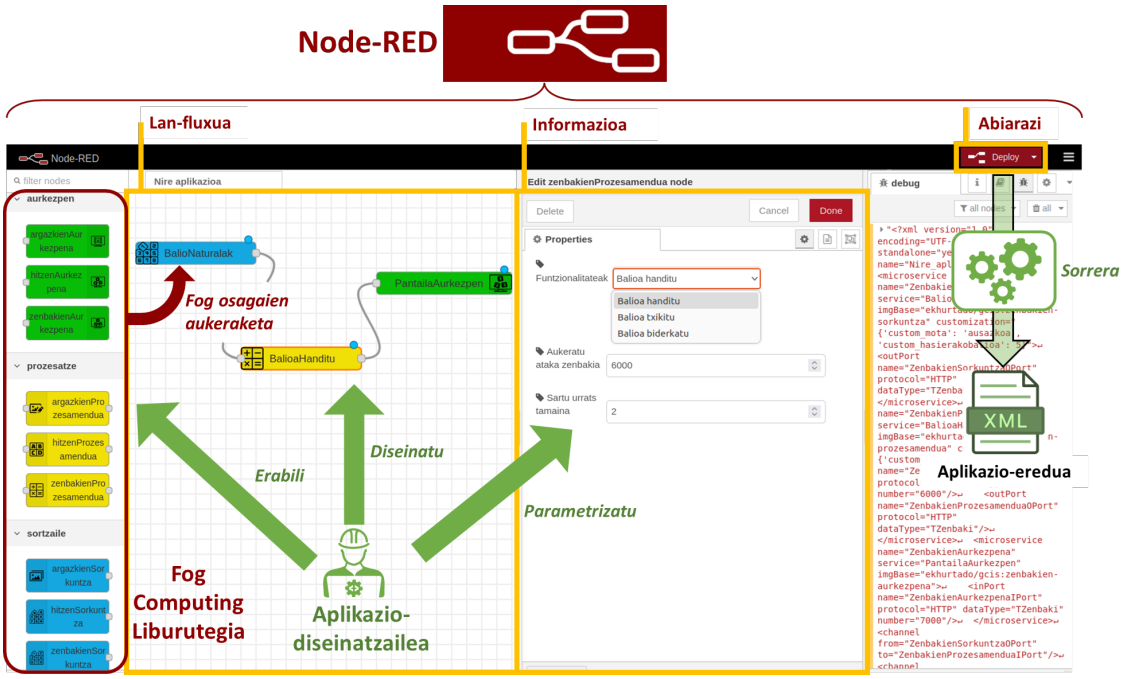
Bestetik, atal funtzionalerako *appModel.js* fitxategia sortzen da. Horrek, aurreko nodotik aplikazio-eredua lortzen du, mikrozerbitzuko informazioarekin aplikazio-ereduaren zatia eraikitzen du, eta eredu eguneratua hurrengo nodora bidaltzen du. Lan-fluxuko azkenengo nodoek, aplikazio-eredua meta-ereduarekin bat datorren konprobatzen dute, erabiltzaileari eredu finala erakutsi aurretik.

Node-RED plataforman nodo pertsonalizatu bat integratzeko beharrezko fitxategia guztiak automatikoki lortzeko programa bat garatu da, [40] errepositorioan *nodeGenerator.py* izenarekin eskuragarri dagoena. Osagai-programatzaileak programan osagai-eredua sartu beharko du eta, honek, XSLT teknologia eta M2T transformazioak erabiliz, Node-REDen zuzenean sar daitekeen nodo bat sortzen du.

Ondoriozta daiteke proposatutako Fog Computing Liburutegiak zeregin garrantzitsua betetzen duela ez bakarrik Fog aplikazioen diseinuan (atal bisuala), baita haien garapenean ere (atal funtzionala).

10.3 Fog aplikazioen diseinu grafikoa

Atal honetan, garatutako Fog Computing Liburutegiaren erabilera eta Fog aplikazioen diseinu grafikoaren prozesua azalduko dira. IDEak diseinatzaileei Fog aplikazioak mikrozerbitzuen lan-fluxu gisa diseinatzeko aukera ematen die (8.1. Irudiko 4. urratsaren inplementazio grafikoa).



Irudia 10.1: Node-REDen Fog aplikazioen diseinu grafikoaren prozesua.

10.1. Irudiak aplikazio-diseinatzaileak Fog aplikazioen diseinu eta garapenerako bete beharreko prozesua aurkezten da. Node-REDek nodoak ezkerreko atalean kokatzen

ditu, beraz, proposatutako Fog Computing Liburutegiaren nodoak bertan egongo dira eskuragarri. Diseinatzaileak instantziatu nahi diren nodoak arrastatu eta erdiko ataleko interfazean jaregin ditzake.

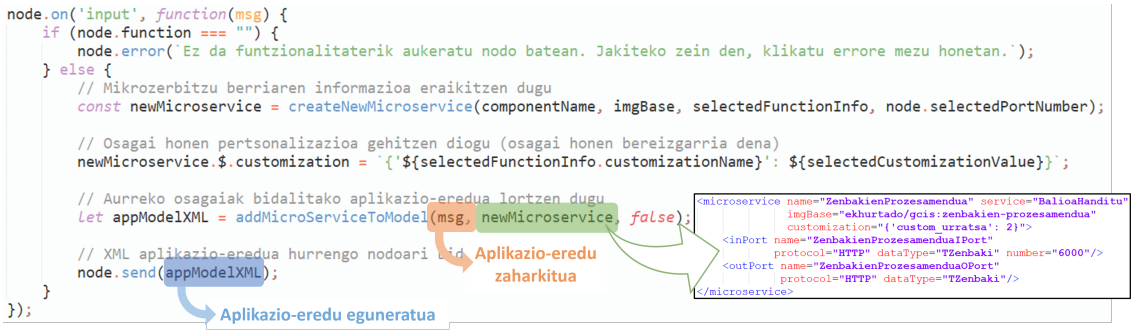
Ondoren, instantziatutako nodoetan klik eginez, beraien informazio atala ikus dezake, desiratutako funtzionalitatea hautatu, etab. Fog osagaiak pertsonalizatzeaz gainera, diseinatzaileak mikrozerbitzuen lan-fluxua zehaztu beharko du. Horretarako, erdiko interfazea erabilita, nodoak elkar ditzake.

10.4 Fog aplikazioen garapen automatikoa

Fog aplikazioa diseinatu ostean, Node-REDek aplikazio-eredua automatikoki gara dezake. *Deploy* botoian klik eginez, nahi den lan-fluxua exekuta daiteke, eta horrekin erlazionatutako aplikazio-eredua, zuzena bada, kontsolan agertuko da, *debug* atalean. Zuzena ez bada, kontsolatik errorearen arrazoia erakutsiko da.

9.2. atalean komentatu denez, aplikazio-ereduaren sorkuntza automatikorako, nodo bakoitzak aurreko nodotik eredua lortzen du, bere informazioa gehitu eta hurrengo nodoari eredu eguneratua bidaltzen dio. Prozesu honen arduraduna *appModel.js* fitxategia da, hau da, nodoaren zati funtzionala.

Lan-fluxuen azkenengo nodoek aplikazio-eredua meta-ereduarekin bat badatorren konprobatzen dute. XSLT teknologia eta M2T transformazioak erabilita, prozesu hauek aurrera eramateko kodea automatikoki sortzen da. 10.2. Irudiak komentatutako garapen automatikoa aurkezten du, *ZenbakienProzesamendua* osagaiaren *appModel.js* fitxategiaren zati bat erakutsiz. Exekutatzen diren metodoekin, horiek sortutako emaitzak ikus daitezke.



Irudia 10.2: Fog aplikazioen ereduaren garapen automatikorako, *ZenbakienProzesamendua* osagaiaren *appModel.js* fitxategiaren zati bat.

Sortutako aplikazio-eredua inplementazio plataformatik independentea izanez, generikoa izango da. Hala ere, kontuan hartu behar da, mikrozerbitzuak funtzionatzeko oinarri-irudiari osagaiaren pertsonalizazio guztia gehitu behar zaiola (funtzionalitatea, portuak...). Horretarako, Docker edukiontziak izanez, ingurune-aldagaiak erabiliko dira, eta horiek irudian sartzeko arduraduna inplementazio plataforma izango da.

11 Fog aplikazioen operazioa orkestrazio plataforma batean

Edukia

Aipatu bezala, aplikazio-eredurako XML lengoaia hautatu denez, soluzio generikoa da. Hala ere, martxan jarri nahi izanez gero, kasu bakoitzak inplementazio plataforma desberdin bat beharko du. Plataforma honen eskakizun bakarra Docker edukiontziak eraikitzeko eta abiarazteko gaitasuna izan behar duela da, proposatutako mikrozerbitzuen oinarria baitira.

Hala ere, kontuan eduki behar da mikrozerbitzuak eraikitzeko, Docker kontainerrari aplikazio-diseinatzaileak gauzatutako pertsonalizazioa gehitu behar zaiola. Aurreko kapituluetan komentatu den bezala, ingurune-aldagaiak dira edukiontziak pertsonalizatzeko baliabide egokienak. Beraz, mikrozerbitzuak eraikitzeko abiaraziko den edukiontziari, eta beraz, Fog osagaiaren oinarri-irudiari, ingurune-aldagai zehatz batzuk esleitu beharko dizkiogu. Alde batetik, diseinatzailearen hautespenak zehazteko aldagaiak gehitu behar dira, eta, bestetik, lan-fluxuarekin erlazionatutako informazioa (sarrera eta/edo irteera portuak, hurrengo mikrozerbitzuaren identifikatzailea...).

Docker edukiontziak eraiki, kudeatu eta abiarazteko orkestrazio plataforma deritzona erabili ohi da. 5. Kapituluan aztertu denez, Kubernetes [33] Docker edukiontziak orkestratzeko de facto plataforma da. Plataforma hedatzeko eskaintzen dituen tresnak erabilita, lan honetan Kubernetesen gehigarri bat proposatzen da, garatutako Fog osagai eta aplikazio kontzeptuak integratuta dituena.

11.1 Kubernetes orkestrazio plataformaren gehigarria

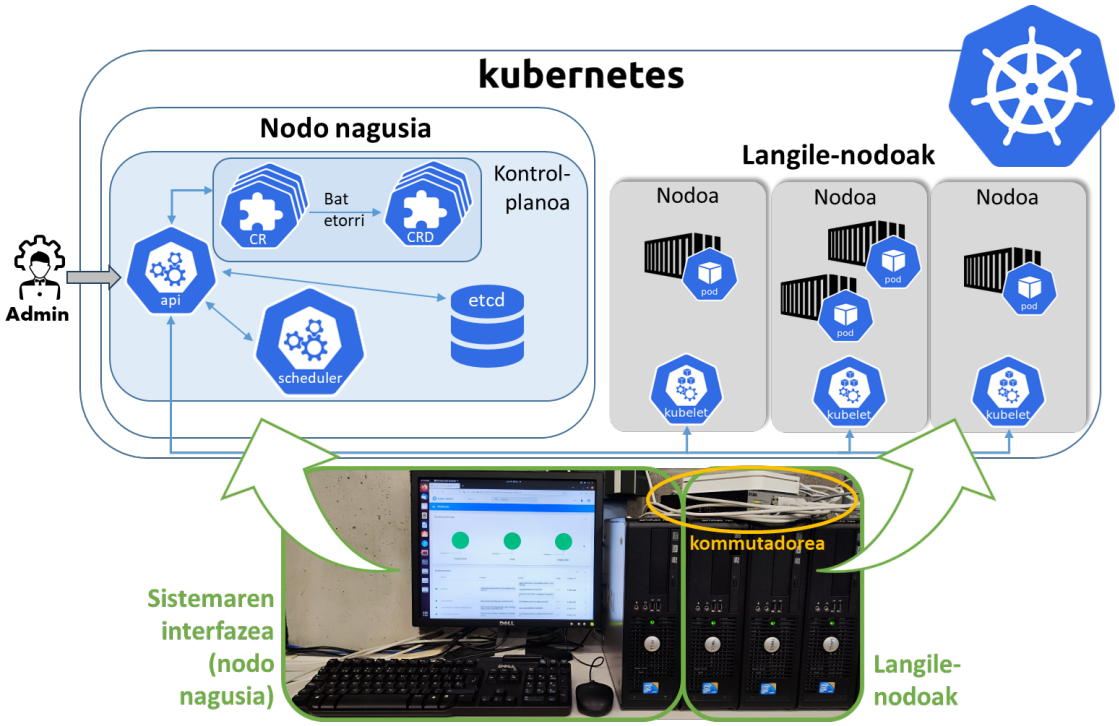
Kubernetes teknologiak hainbat abantaila eskaintzen ditu: kargen orekatzea, informazio-biltegiratzea, akatsak dituzten kontainerrak berrabiaraztea edo konfigurazioaren kudeaketa, besteak beste. Kubernetes kluster bat edukiontzidun aplikazioak exekutatzen dituzten nodo izeneko langile-makina multzo batek osatzen dute. Kluster bakoitzak gutxienez

langile-nodo (*worker node*) bat du. Horrela, nodo horiek aplikazioen osagai diren mikrozerbitzuak hartzen dituzte, *Pod* izeneko elementuetan kokatzen direnak.

Klusterraren kudeaketarako nodo nagusi bat dago (*master node*), kontrol-planoa barneratzen duena. Kontrol-planoak langile-nodoak eta *Pods* bezalako elementuak kudeatzen ditu eta hainbat elementuez osatuta dago. Alde batetik, erabiltzaileek elkarreragiten duten elementuak, hau da, Application Programming Interface edo APIak, *Kubernetes API*a eskaintzen duena plataformarekin lan egiteko. Beste alde batetik, *etcd* edo sistemaren datu-basea gako-balioetan oinarritutako biltegi koherentea eta eskuragarria da, klusterraren datu guztiak gordetzen dituena [33].

Planoko beste elementu bat *scheduler* edo planifikatzailea da, sortutako *Pod* elementuentzako langile-nodo aproposena zehazten duena. Erabakiak hartzeko kontuan hartu daitezkeen faktoreak era askotakoak dira: baliabideen eskakizunak, hardware edo software mugak, datuen kokapena, etab.

Kontrol-planotik kanpo, langile-nodoen kudeaketarako bakoitzak *kubelet* izendatutako elementu bat dauka. Hori nodo bakoitzean exekutatzen den agente bat da, edukiontziak *Pod*etan zuzen exekutatzen ari direla ziurtatzen duena. Kubernetes plataforma sortzeko GCIS ikerketa-taldeko baliabideak erabili dira: pantaila, arratoi, teklatu eta ordenagailu-multzo bat. Horietan Linux sistema eragileak izanda, klusterra eraiki ahal izan da. Komentatu dena grafikoki aztertzeko, 11.1. Irudiak Kubernetes plataformaren egitura aurkezten du.



Irudia 11.1: Kubernetes plataformaren egitura.

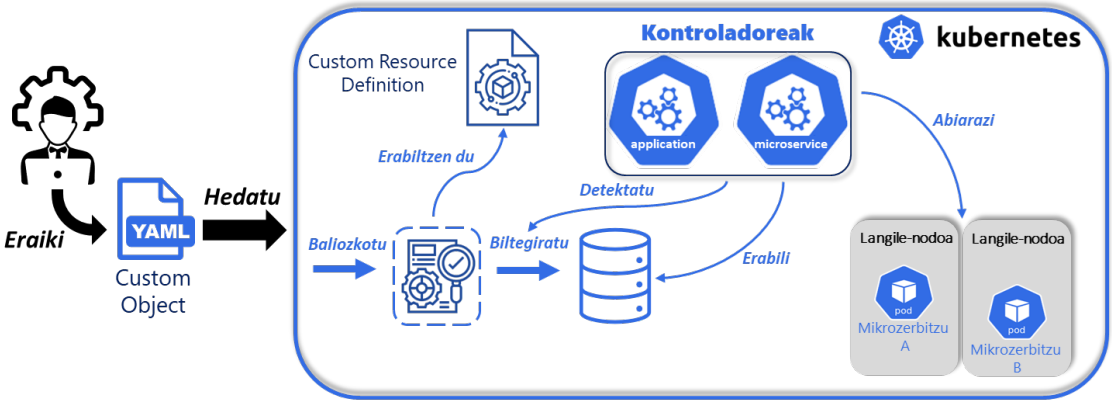
Kubernetesek plataformari gehigarri bat integratzeko *Custom Resource* deituriko baliabideak eskaintzen ditu [42]. *Custom Resource* Kubernetes APIaren gehigarria da, eta ez dago nahitaez Kubernetes instalazio lehenetsi batean eskuragarri. Kubernetes instalazio jakin baten pertsonalizazioa adierazten du, APIa osotuz elementu mota berriak integratzen ahalbidetzen baititu [15].

Custom resource edo *CR* baliabideak martxan dagoen plataforma edo kluster batean ager eta desager daitezke erregistro dinamikoaren bidez, eta klusterraren administratzaileek eguneratzeko aukera dute. Baliabide pertsonalizatu bat instalatuta dagoenean, erabiltzaileek sortu eta atzi ditzakete komando-lerroko interfaze bat edo beste tresna ezberdin bat erabiliz.

Esan bezala, datu egituratuak gordetzeko eta berreskuratzeko aukera ematen dute. Benetako API deklaratzailea eskaintzeko baliabide bakoitza kontroladore edo *Custom Controller* batekin konbinatu beharko da. Hau da, deklarazio APIak erantzukizunen banaketa ezartzen du, nahi duzun baliabidearen egoera adieraziz kontroladoreek Kubernetes objektuen uneko egoera sinkronizatuta mantentzen baitute.

Bestetik, *Custom Resource Definition* elementuak baliabide pertsonalizatuak nola izan behar diren definitzeko aukera ematen dute, hau da, baliabide horien meta-ereduak dira. Meta-eredu horiek jarraituz, objektu pertsonalizatuak edo *Custom Object* elementuak sor daitezke.

Horrela izanda, baliabide mota horiek erabilita, proiektu honek Kubernetes plataformaren gehigarri bat proposatzen du. *Custom Resource Definition* edo *CRD* meta-ereduekin objektu pertsonalizatu zuzenak defini daitezke, hala nola aplikazio ereduak. Erlazionatutako kontroladoreek, objektu horien sorkuntza detektatu eta aplikazioen osagaiak automatikoki heda ditzakete plataforman, hau da, mikrozerbitzuak. Aipatutako prozesua 11.2. Irudian aurkeztu da.



Irudia 11.2: Kubernetes plataforma hedagarriaren erabileraren prozesua, kasu honetan, Fog aplikazio baten hedatzea.

Kubernetesen sor eta kudea daitezkeen unitate hedagarri txikienak *Pods* deritzonak dira, eta bertan edukiontziak kokatuko da. Kubernetesek Docker baino teknologia gehiago onartzen dituen arren, Docker ezagunena da, eta Dockerreko terminologia batzuk erabiliz *Pod*ak deskribatzen laguntzen du.

Orkestrazio plataforma honen hedatze elementuak *Deployment* deiturikoak dira. Objektu horiek edukiontzidun osagai bat duten *pod* instantziak nola sortu edo aldatu behar diren definitzen dute. Horregatik, osagai bat Kubernetesen hedatu nahi izanez gero, *Deployment* objektu bat sortu behar da, eta horrekin erlazionatutako *Pod* bat abiaraziko da, edukiontzia barruan martxan jarriko dena.

11.1.1 Aplikazio eta mikrozerbitzu mailen definizioak

Betiere proposatutako meta-ereduak oinarri gisa hartuko dira, inplementazio plataformaren gehigarria garatzeko. Aipatu bezala, aplikazioak osagaiez osatuta egongo dira, azken horiek mikrozerbitzuak izango direnak. Horregatik, Kubernetes plataformaren gehigarrirako aplikazio eta mikrozerbitzu mailak edo objektu motak sortuko dira.

Beraz, maila bakoitzerako erlazionatutako meta-ereduak plataforman integratu dira. Horretarako, *CRD* objektuak erabiliko dira. Komentatu beharra dago Kuberneteseko objektu guztiak YAML lengoaian definituta daudela. *CRD* objektuetan bi zati garrantzitsu definituko ditugu: *spec* eta *status*. Objektuaren *spec* atalean meta-eredua bera sartuko da, *CR* objektuen espezifikazioa baita. Ostera, *status* atazan, *CR*en egoera jakiteko beharrezko informazioa gehituko da. 11.3. Irudian aplikazio-mailaren *CRD* fitxategia erakutsi da, mikrozerbitzu mailaren fitxategiarekin batera [40] errepositorioan eskuragarri daudenak.

Azter daitekeen bezala, Fog aplikazioen meta-eredua zuzenean integra daiteke. Elementu bakoitzerako zein atributu osatzen dituzten eta atributu horiek zein motakoak izan behar diren zehatz daiteke. *CRD* objektu hori Kubernetesen integratuta dagoenean, aplikazio bakoitzeko *CR* objektuak espezifikazio honekin bat datozen konprobatuko da, eta bakarrik baiezkoa denean biltegiratuko dira sistemaren datu-basean.

Mikrozerbitzu mailarako, Fog aplikazioak osatutako mikrozerbitzuen definizioa gehitu da, *CRD* objektu bat ere sortuz. Fitxategiaren *spec* zatia 11.3. Irudiko *microservices* atalaren eduki berdina izango du eta *status* zatia sinpleagoa izango da, mikrozerbitzu bakoitzaren egoerarako informazioa gehituz. CRD bakoitza berarekin erlazionatutako CR motarako balioko dute, beraz, egiaztapenak banaka egingo dira, objektu motaren arabera erlazionatutako CRD definizioa erabiliz.

Irudia 11.3: Aplikazio mailarako Kuberneteseko *CRD* fitxategia.

Aplikazio bat sortzerakoan, hau da, aplikazio motako *CR* bat, aplikazioa osatzen duten mikrozerbitzu bakoitzerako mikrozerbitzu motako *CR* bat sortuko da. Hori horrela bete egin da, objektu bakoitzak bere informazioa edukitzeko. Beraz, sistemaren administratzaileak aplikazioaren egoera osoa edo mikrozerbitzu bakar baten informazioa lor dezake.

11.1.2 Aplikazio eta mikrozerbitzu mailen kontroladoreak

Fog aplikazioen operazio automatikorako, pertsonalizatutako objektuekin lan egiteko kontroladoreak sortu dira, [40]-an eskuragarri daudenak. Hauek Python programazio-lengoaian garatu dira eta Kuberneteseko APIa erabiliko dute objektu horiek abiarazteko eta beraien bizi-ziklo osoa kudeatzeko.

Aurreko atalean aipatutako maila bakoitzerako kontroladore bat garatu da, bakoitza jarduera batzuez arduratuko dena. Kontroladoreek beraien *CR* objektuen gertaeren begirale bat inplementatuta edukiko dute, gertaera horien aurren beharrezko ekintzak burutzeko.

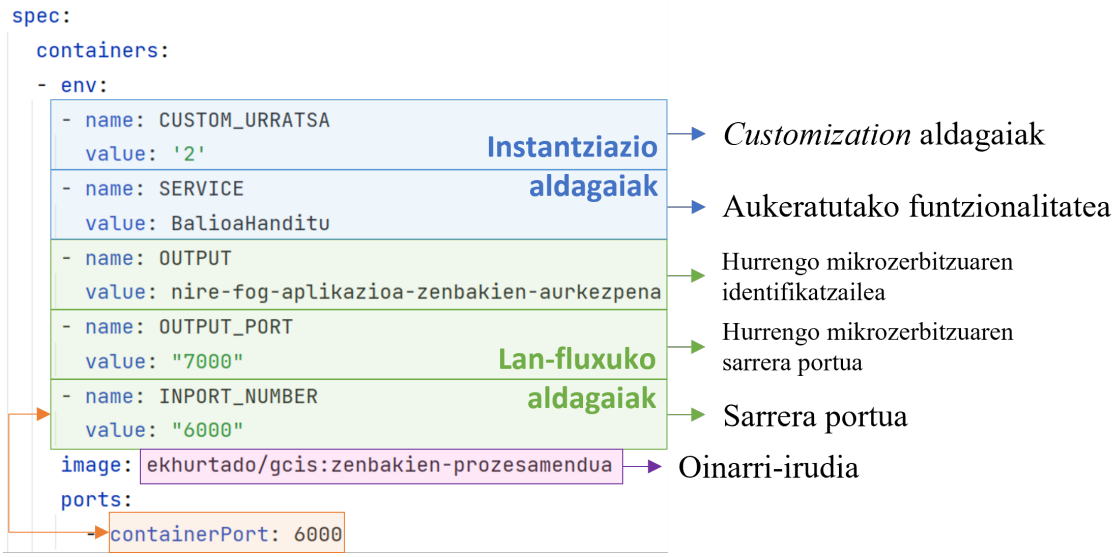
Aplikazio mailaren kontroladoreak aplikazio motako *CR*en hiru gertaera nagusi kudeatuko ditu: sorkuntza, aldaketa eta ezabapena. Aplikazio berri bat sortzen denean, kontroladoreak gertaera detektatu eta aplikazioa abiarazteko prozesua hasiko du. Lehenengo, aplikazioaren objektuaren *status* atala eraikiko du, berez Kubernetesek ez baitu egiten.

Ondoren, mikrozerbitzu bakoitzaren objektua sortuko du, aplikazioaren *CR*eko informazioarekin. Mikrozerbitzuaren identifikadorea sisteman bakarra izatea ziurtatzeko "*<aplikazio izena>-<mikrozerbitzu izena>*" formatua ematen zaio. Beste mikrozerbitzuekin komunikatzeko eskuragarri egoteko beharra badauka, Kubernetesek *Service* objektuak eskaintzen ditu. Horrek aplikazioak nola atzitzen diren deskribatzen duen objektua da, adibidez, *Pod*s multzo bat eta beraien atakak deskriba ditzake. Horregatik, komunikazioa ahalbidetu beharreko kasuetan, zerbitzu objektu bat sortuko da baita ere.

Mikrozerbitzu mailaren kontroladoreak, berriz, mikrozerbitzu motako *CR*en bi gertaera mota kudeatuko ditu: sorkuntza eta ezabapena. Aplikazioen kontroladoreak mikrozerbitzu objektuak sortzerakoan, mikrozerbitzuen kontroladoreak gertaera detektatu eta mikrozerbitzu hori abiarazteko prozesuari hasiera emango dio.

Lehenik, *status* atalean hasierako egoera gehituko da, hau da, mikrozerbitzua abiarazteko prozesuan dagoela. Ondoren, mikrozerbitzuarekin lotutako *Deployment* objektua sortuko da eta sisteman hedatuko da, inplementaziorako informazioa erabilita, hala nola aukeratutako funtzionalitatea, sarrera eta irteera portuak, etab. Hedatze prozesuan *Pod* bat eta barruan martxan jarriko den edukiontzia sortuko da.

Mikrozerbitzu zuzena lortzeko, edukiontziko oinarri-irudian aipatutako inplementazio informazioa zehaztu behar da. Horretarako, kapituluaren hasieran komentatu denez, ingurune-aldagaiak erabiliko dira. Horrela, mikrozerbitzu kontroladoreak *Deployment* objektuan ingurune-aldagai guztiak zehaztuko ditu, exekuzio denboran eskuragarri egoteko. Aipatu beharra dago, ingurune-aldagaiek maiuskulak eta minuskulak bereizten dituztela. Konbentzioz, baina ez arauz, izenak beti letra larriz idazten dira. Honakoa irudikatzeko 11.4. Irudiak *ZenbakienProzesamendua* osagaiaren mikrozerbitzu baten *Deployment* objektuaren zati bat erakusten du. Azter daitekeenez, 9.2. Kodean agertzen diren aldagaien izen berdina dute, exekuzio-denboran iturri-kodetik balioak lortuko direla ziurtatuz.



Irudia 11.4: *ZenbakienProzesamendua* osagaiaren mikrozerbitzu baten *Deployment* objektuaren ingurune-aldagaiak zehazten diren zatia.

Mikrozerbitzua martxan dagoenean (*Running* egoera), mikrozerbitzuen kontroladoreak mikrozerbitzu horrekin erlazionatutako aplikazio objektuaren *status* atalean bere zatia eguneratuko du, mikrozerbitzu hori abiarazita dagoela zehaztuz. Horrela, aplikazio zein mikrozerbitzuen egoerak eguneratuta egongo dira. Azaldutako prozesua grafikoki aurkezteko, 11.5. Irudiko sekuentzia-diagrama garatu da.

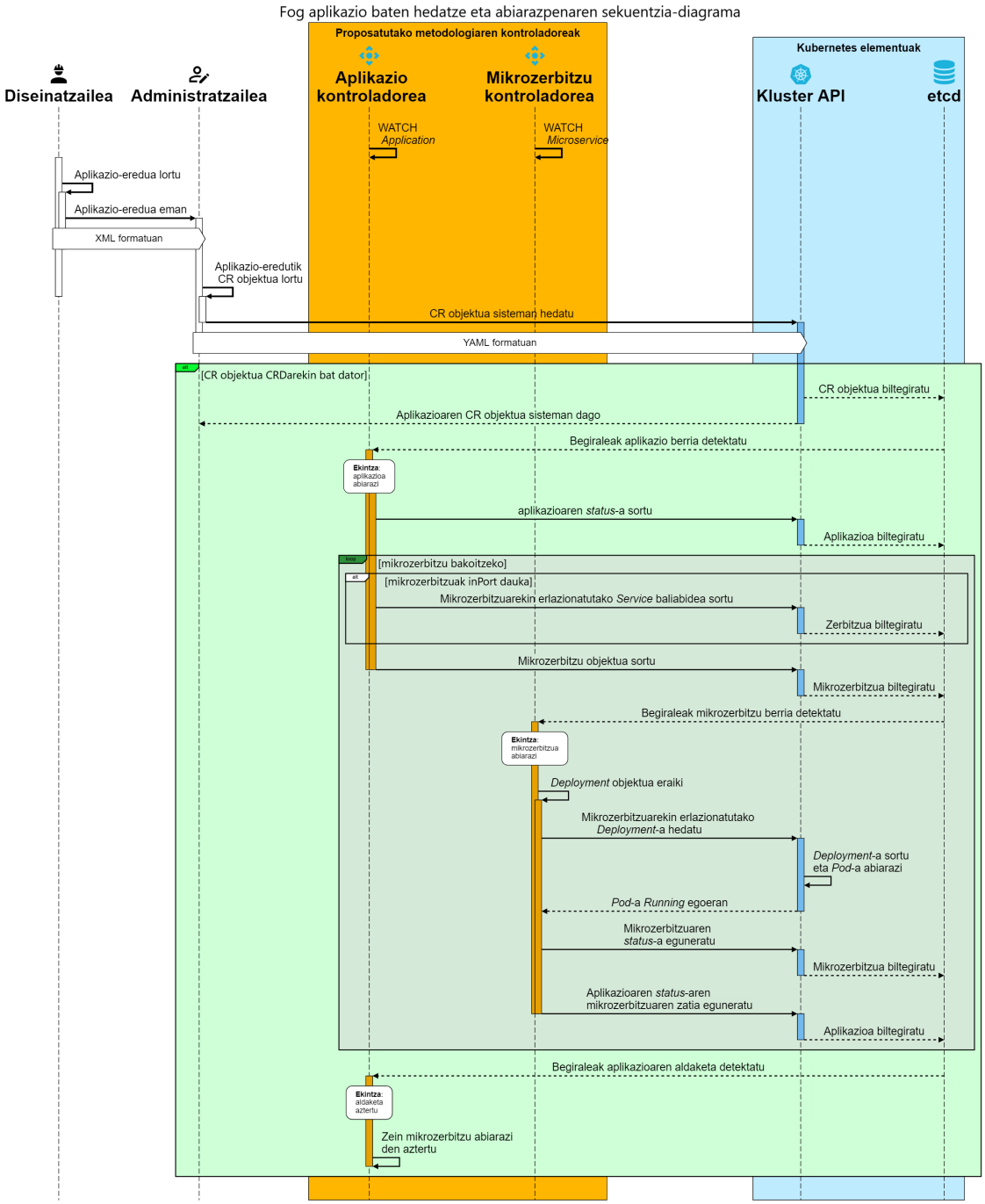
Ezabaketa prozesurako, sistemaren administratzaileak aplikazio objektua ezabatu beharko du, hedatze-fitxategi bera edo komando-lerroko interfazea erabiliz. Aplikazioen kontroladoreak gertaera detektatu eta aplikazio horren mikrozerbitzuen *CR* objektuak ezabatuko ditu. Mikrozerbitzuen kontroladoreak, bere aldetik, azkenengo gertaera dela-eta mikrozerbitzuekin lotutako *Deployment* objektuak ezabatuko ditu, horrek automatikoki *Pod* bakoitza ezabatuz.

Azter daitekeen bezala, sistemaren administratzaileak bakarrik aplikazio objektua sortu beharko du. Kontroladoreek mikrozerbitzuen objektuak automatikoki sortuko dituzte. Bestalde, Kubernetesek eskaintzen duen komando-lerroko interfazea erabilita, lan honetan proposatutako edozein objektuen informazioa lor daiteke.

Aplikazioa abiarazita dagoenean, mikrozerbitzu guztiak elkarren artean komunikatu ahalko dira, horretarako Kubernetes baliabideak, esaterako, *Services* objektuak, eta beharrezko ingurune-aldagaiak zehaztu direlako (lan-fluxuarekin erlazionatutako informazioa). Horrez gain, Fog aplikazio baten mikrozerbitzuak ez dira klusterraren langile-nodo berdinean egon behar. *Services* baliabideei esker, edozein nodotan egonda mikrozerbitzuak eskuragarri egongo dira.

Gainera, Kubernetes plataformak berak zenbait kudeaketa jarduera gauzatzen ditu. Langile-nodo batek akatsen bat jasanez gero, kontrol-planoak gertaera detektatzen du eta

langile-nodo horretan abiarazita zeuden *Pod* guztiak gainontzeko nodoetan banatzen ditu. Mikrozerbitzuaren Docker edukiontziak ere huts egiten badu, plataformak automatikoki berreraikitzen du.



Irudia 11.5: Fog aplikazio baten hedatze prozesuaren sekuentzia-diagrama.

11.2 Aplikazio-eredutik hedatze fitxategien sorkuntza automatikoa

Aztertu denez, proposatutako software plataformak aplikazioaren inplementazio prozesua guztiz automatiza dezake. Beharrezko baliabide bakarra *Custom Resource* fitxategia garatzea da eta hori sisteman hedatzerakoan, aplikazio eta bere osagaiak diren mikrozerbitzuak automatikoki abiaraziko dira.

Kubernetesek *CRD*en laguntzaz proposatutako Fog aplikazioen meta-eredua plataforman integratzeko aukera eskaintzen duenez, hedatze aurretik gertatzen den jarduera bakarra aplikazio-ereduko informazioa *CR* objektu batera transformatzea izango da, hala nola 8.1. Irudiko 5.1. urratsa.

Horretarako, transformazio hori automatizatzen duen programa bat garatu da, [40] errepositorioan eskuragarri dagoena *appModelTransformer* izenarekin. Horrek, XML lengoaian idatzitako aplikazio-eredua lortzen du, Fog aplikazioen meta-ereduarekin bat datorren konprobatzen du, eta XSLT teknologia eta M2T transformazioak erabilita, *Custom Resource* objektua eraikitzen du, ereduko informazio guztia gehituz.

Sortutako objektua informazio zuzena edukiko du, Fog aplikazioen meta-ereduarekin konprobatu delako. *CR* objektua YAML fitxategi batean biltegiratzen denez, sistemaren administratzaileak zuzenean fitxategi hori Kubernetes klusterrean heda dezake. Hedatzerakoan, Kubernetesek automatikoki ea objektua proposatutako aplikazioentzako *CRD* espezifikazioaren bat datorren konprobatuko du, eta, zuzena izanez gero, sistema biltegiratuko da eta aplikazioa abiaraziko da.

Atala III

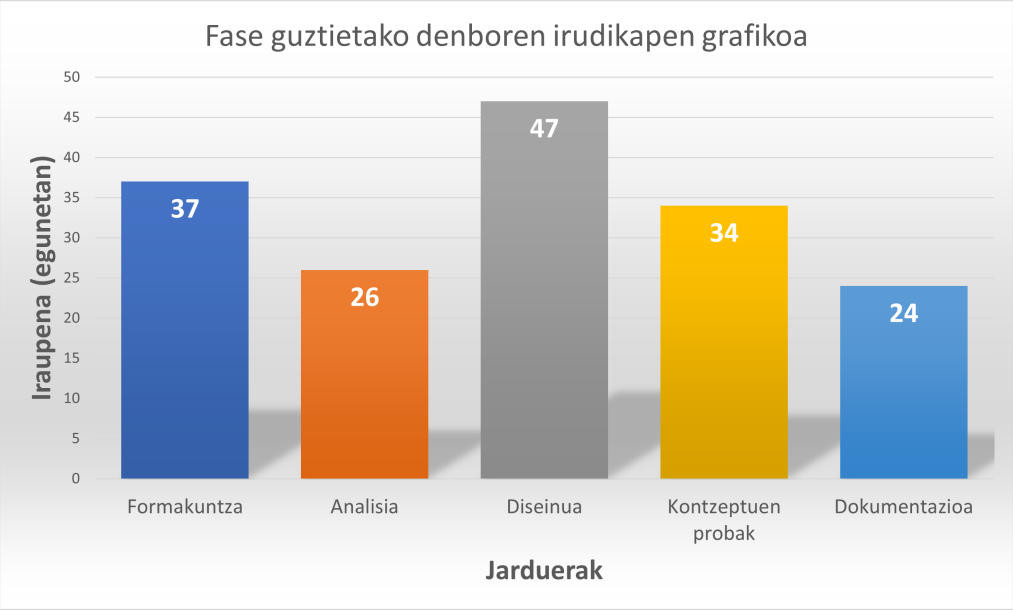
METODOLOGIA

Jardueren eta faseen lan plangintza 12

Edukia

MAL honetan zehaztutako helburuak lortzeko garatutako plangintza kapitulu honetan aurkezten da. 2. Kapituluan erakutsi denez, proiektuak hainbat helburu orokor eta partzialak ditu, horietako batzuek era independentean azter daitezkeenak eta beste batzuek erlazionatuta daudenak.

Betetako jarduera guztiak bost faseetan sailkatu dira, bakoitzak proiektuaren zati garrantzitsu bat barneratuko duena. Fase bakoitzean gauzatutako jarduerak aztertuko dira eta horiek aurrera eramateko behar izandako denbora. 12.1. Grafikoak faseetan zehar proiektu osoko denbora nola banandu den erakusten du.



Irudia 12.1: Faseetan zehar proiektu osoko denboraren banaketaren grafikoa.

Seigarren fase bat zehaztu daiteke, baina ez da aztertu *hammock* motako jarduerez osatuta baitago, hau da, proiektu osoan zehar irauten duten jarduerak. Proiektuaren kudeaketa aipatutakoaren adibidea da, MALa era zuzenean aurrera eramateko tutoreekin hainbat bilera eduki baitira, jarduerak aldatuz edo berriak ezarriz.

12.1 1. fasea: Formakuntza

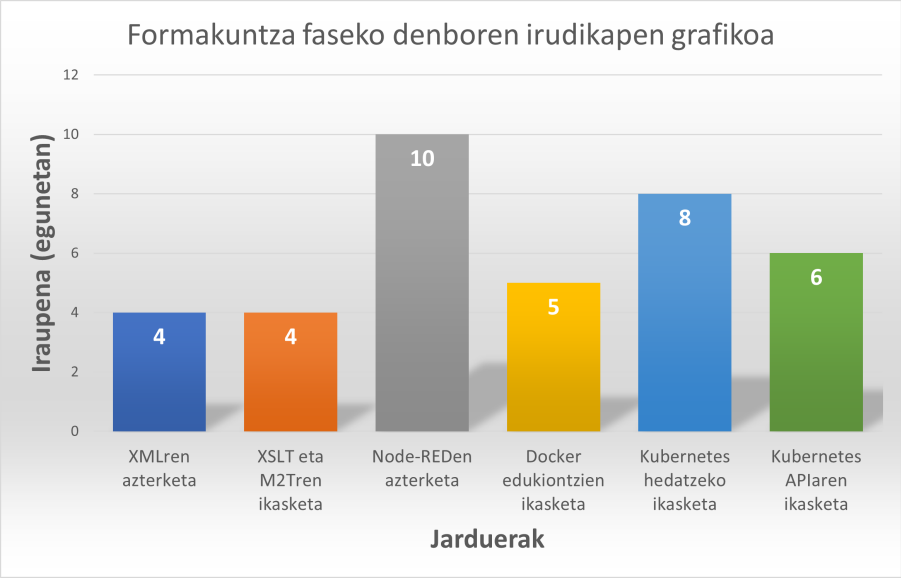
Proiektu honen lehenengo fasea erabiliko diren teknologien eta tresnen formakuntza da. Aipatu beharra dago Gradu Amaierako Lanean (GrAL) Docker eta Kubernetesen ezagutza lortu zela, beraz, teknologia horietan bertan lortutako oinarritik hasi da. Lan honetan pertsonalizazio eta aukera anitz eskaintzen dituzten teknologiak erabili direnez, haien errendimendu osoa ateratzeko beharrezko denbora esleitu da. 12.1. Taulan jarduera bakoitzaren eta fase osoaren data eta iraupena zehaztu da.

Taula 12.1: Formakuntza faseko jarduerek erabilitako denboren taula. Iraupena laneguneta adierazita dago.

Kodea	Jardueraren deskribapena	Hasiera-data	Amaiera-data	Iraupena
F1	**Formakuntza**	**2022/11/16**	**2023/01/06**	**37**
J1.1	XMLren azterketa	2022/11/16	2022/11/21	4
J1.2	XSLT eta M2Tren ikasketa	2022/11/22	2022/11/25	4
J1.3	Node-REDen azterketa	2022/11/28	2022/12/09	10
J1.4	Docker edukiontzien ikasketa	2022/12/12	2022/12/16	5
J1.5	Kubernetes hedatzeko ikasketa	2022/12/19	2022/12/28	8
J1.6	Kubernetes APIaren ikasketa	2022/12/29	2023/01/06	6

Lehenik eta behin, meta-ereduekin lan egiteko XML teknologiaren azterketa bete da. Gero, eredu horiekin gauzatu beharko diren transformazioetarako XSLT eta M2Tri buruzko ezagutza lortu da. Meta-ereduekin lotutako teknologiekin bukatuta, garapen-ingurune integratua eraikitzeko Node-RED plataformaren azterketa burutu da. Bertan, plataforma nola funtzionatzen duen, nodo pertsonalizatuak sortuz nola heda daitekeen eta meta-ereduak nola integra daitezkeen ikertu da.

Komentatu denez, Docker eta Kubernetesen ezagutza edukita, jarduera horien iraupena txikiagoa izan da. Batez ere, Kubernetes plataformaren hedagarria garatzeko eta Docker edukiontzi desberdinak bertan komunikatzeko baliabideak aztertu dira. Azkenik, Kuberneteseko kontroladoreek beraien funtzioa betetzeko APIa erabili beharko dutenez, honen irismena analizatu da. 12.2. Irudiak 12.1. Taulan zehaztutako jarduerek erabilitako denboren irudikapen grafikoa aurkezten du.



Irudia 12.2: Formakuntza fasean jardueren denboraren grafikoa.

12.2 2. fasea: Analisia

Bigarren fase honetan, teknologien analisia beteko da, aurreko fasean lortutako ezagutza erabiliz. Literaturan hainbat tresna desberdin erabiltzen dira eta egunero berriak agertzen dira. Horregatik, helburu bakoitzerako teknologia egokiena hautatu behar da, eskuragarri dauden aukeren analisia osatuz. 12.2. Taulak egindako analisien iraupenak eta datak aurkezten ditu.

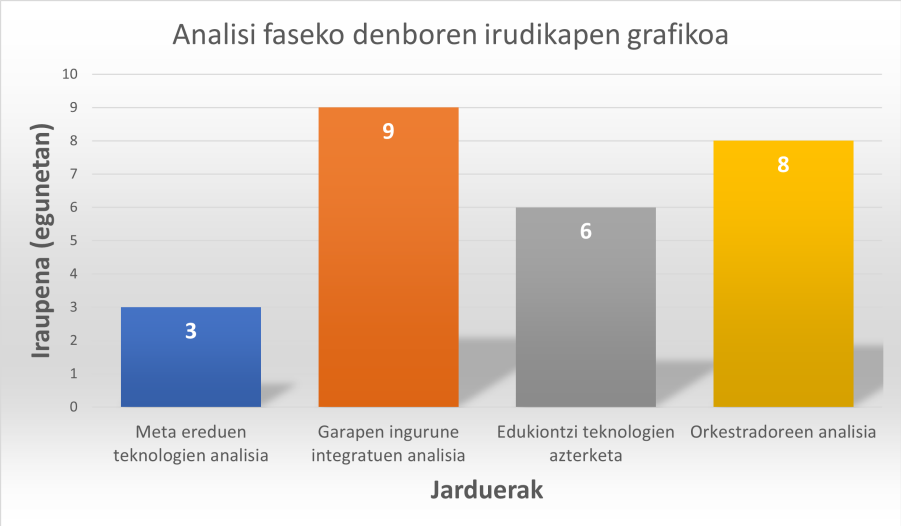
Taula 12.2: Analisi faseko jarduerek erabilitako denboren taula. Iraupena lanegunetan adierazita dago.

Kodea	Jardueraren deskribapena	Hasiera-data	Amaiera-data	Iraupena
F2	**Analisia**	**2023/01/09**	**2023/02/14**	**26**
J2.1	Meta-ereduen teknologien analisia	2023/01/09	2023/01/11	3
J2.2	Garapen-ingurune integratuen analisia	2023/01/12	2023/01/24	9
J2.3	Edukiontzi teknologien azterketa	2023/01/25	2023/02/01	6
J2.4	Orkestradoreen analisia	2023/02/02	2023/02/14	8

Hasteko, meta-ereduak sortzeko, aldatzeko eta egiaztapenak egiteko teknologien analisia gauzatu da. Bestalde, Fog aplikazioekin lan egiteko tresna grafiko baten beharra aztertu da, garapen-ingurune integratuaren ardura har dezakeena. Lan honek dituen ezaugarriak

kontuan hartuz (hedagarria, pertsonalizagarria, etab.) aukera egokiena lortzeko analisia burutu da.

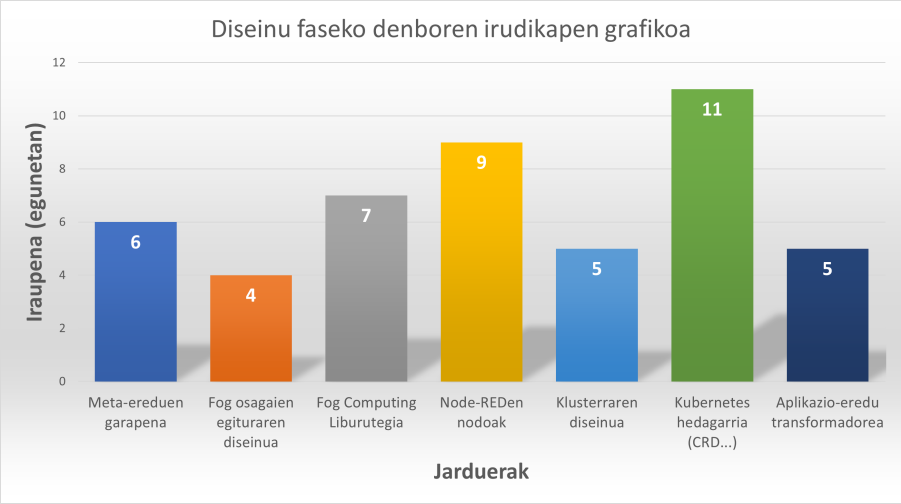
Gainera, GrALean lortutako ezagutzarekin edukiontzi eta horiek orkestratzeko teknologien aukerak analizatu dira, Fog aplikazioak osatuko dituzten mikrozerbitzuak sortzeko, kudeatzeko eta abiarazteko. Jarduera guzti horien denborak 12.3. Irudian grafikoki aurkeztu dira.



Irudia 12.3: Analisi fasean jardueren denboraren grafikoa.

12.3 3. fasea: Diseinua

Proiektua aurrera eramateko teknologien aukeraketa eginez, eta horiekin lan egiteko ezagutza edukiz, lanaren oinarria den proposatutako ikuspegiaren diseinua bete egin da. Fase hau, formakuntzarekin batera denbora gehien hartzen duten faseak dira, kontzeptu proban erabiliko diren tresna eta plataformen diseinua eta eraikuntza osatu egin baita.



Irudia 12.4: Diseinu fasean jardueren denboraren grafikoa.

12.4. Irudian jarduerek duten denbora-banaketa grafikoki aurkeztu da, 12.3. Taulako informazioarekin bat egiten duena. Hasteko, Fog aplikazio eta osagaien meta-ereduen diseinua garatu da. Ondoren, Fog aplikazioek osatuko dituzten osagaien egitura diseinatu da, kontuan edukiz mikrozerbitzuak sortzeko erabiliko diren teknologiak. Diseinu hori edukita, Fog Computing Liburutegia automatikoki sortzeko programen diseinua bete da.

Aurreko atalerako, garapen-ingurune integratuaren teknologia kontuan hartu da, Node-RED plataforma, hain zuzen ere. Plataformak desiratutako funtzionalitatea izateko, nodo pertsonalizatuak garatu dira, tresna hedatua eginez. Nodo horietan, beharrezko funtzioak garatzeaz gain, J3.1 jardueran lortutako meta-ereduak integratu dira.

Garapen-ingurune integratuarekin bukatuz, inplementazio-plataformaren garapena aurrera eraman da. Horretarako, klusterraren diseinua, eta bertan sortuko den Kubernetes plataformaren garapena bete da. Gainera, inplementazio-plataforma proposatutako ikuspegiarekin bat etortzeko, hobetu egin da, hedagarria sortuz. Adibidez, hedagarri horretan aipatutako meta-ereduak integratu dira. Azkenik, garapen-ingurune integratuaren emaitza den aplikazio-eredua inplementazio-plataforman exekutagarri izateko transformadorea garatu egin da.

Taula 12.3: Diseinu faseko jarduerek erabilitako denboren taula. Iraupena laneguneta adierazita dago.

Kodea	Jardueraren deskribapena	Hasiera-data	Amaiera-data	Iraupena
F3	**Diseinua**	**2023/02/15**	**2023/04/21**	**47**
J3.1	Meta-ereduen garapena	2023/02/15	2023/02/22	6
J3.2	Fog osagaien egituraren diseinua	2023/02/23	2023/02/28	4
J3.3	Fog Computing Liburutegirako programak	2023/03/01	2023/03/09	7
J3.4	Node-REDen nodo pertsonalizatuak	2023/03/10	2023/03/22	9
J3.5	Klusterraren diseinua	2023/03/23	2023/03/29	5
J3.6	Kubernetes hedagarria (CRD, kontroladoreak...)	2023/03/30	2023/04/13	11
J3.7	Aplikazio-eredurako transformadorea	2023/04/14	2023/04/21	5

12.4 4. fasea: Kontzeptuen probak

Proiektuaren proposamena den ikuspegirako plataforma guztien diseinua edukita, horren funtzionamendua zuzena den konprobatzeko kontzeptuen probak burutu egin dira. Lehenik eta behin, Fog aplikazio desberdinen ideiak batu egin dira, eta proposamenerako

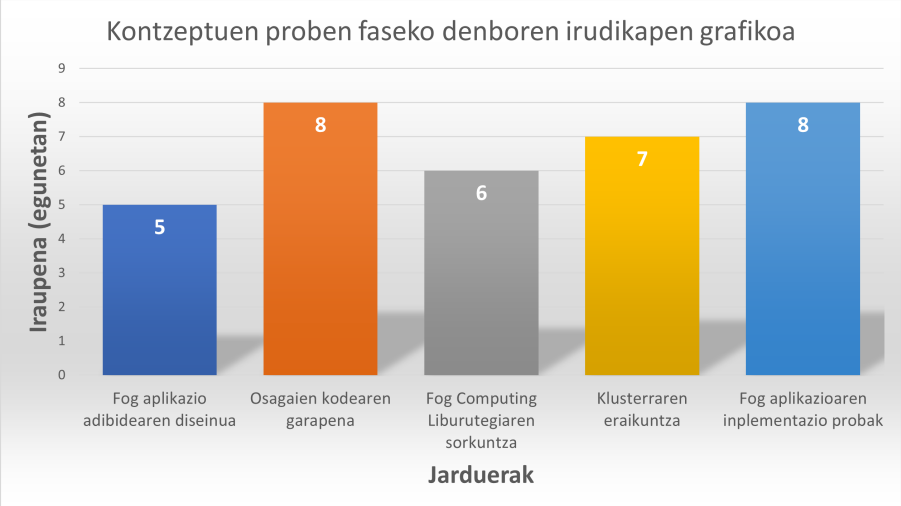
egokiena den diseinua bete da. Kontuan hartu izan dira, aplikazioa osatuko duten Fog osagaiak berrerabilgarriak, sinpleak edo bateragarriak izatea, besteak beste.

Fog aplikazioaren diseinua edukita, osagaien kodearen garapena aurrera eraman da. Programazio-lengoaia desberdinetan eraikita egonda, bakoitzak beharrezko baliabideak zehaztu dira baita ere, horien Docker oinarri-irudia sortuz. Ondoren, osagai horiekin Fog Computing Liburutegia osatu da, aplikazio-eredua lortzeko osagai guztiak eskuragarri edukitzeko.

Aurreko fasean diseinatutako klusterra eraiki egin da, inplementazio-plataforma lortuz. Tresna eta plataforma guztiak prest egonda, Node-REDekin aplikazio-eredua sortuko da, transformadorearekin hedatze-fitxategia lortuko da, eta Kubernetesen abiaraziko da, Fog aplikazioaren garapen eta operazio probak betez. Fase honek osotzen dituzten jardueren denborak 12.4. Taulan eta grafikoki 12.5. Irudian aurkeztu dira.

Taula 12.4: Kontzeptuen proben faseko jarduerek erabilitako denboren taula. Iraupena lanegunetan adierazita dago.

Kodea	Jardueraren deskribapena	Hasiera-data	Amaiera-data	Iraupena
F4	**Kontzeptuen probak**	**2023/04/24**	**2023/06/09**	**34**
J4.1	Fog aplikazio adibidearen diseinua	2023/04/24	2023/04/28	5
J4.2	Osagaien kodearen garapena	2023/05/01	2023/05/10	8
J4.3	Fog Computing Liburutegiaren sorkuntza	2023/05/11	2023/05/18	6
J4.4	Klusterraren eraikuntza	2023/05/19	2023/05/29	7
J4.5	Fog aplikazioaren garapen eta inplementazio probak	2023/05/30	2023/06/09	8



Irudia 12.5: Kontzeptuen proben fasean jardueren denboraren grafikoa.

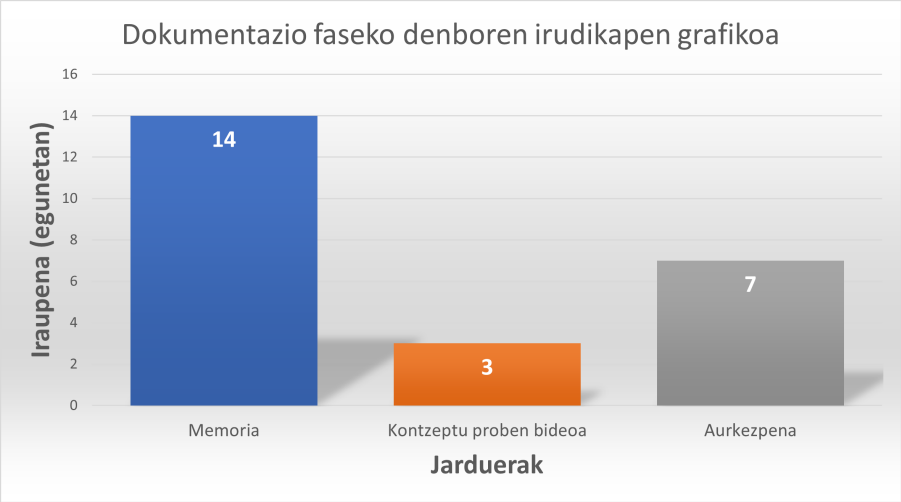
12.5 5. fasea: Dokumentazioa

Azkenengo fasea garatutako lan guztiaren dokumentazio zuzena egitea izan da, proposatutako metodologia erraz jarraitzeko eta aztertu beharreko atal garrantzitsurik geratu ez dadin bermatzeko. Aipatu beharra dago, aurreko fase guztietan zehar egindako lana dokumentatuz joan dela, eta amaitzerakoan informazio guztia MALeko memoriara pasa egin dela.

Hori dela eta, memoria garatzeko jarduera guztietako informazioa batu, analizatu eta garbira pasa da. Bestalde, memoriaren bestelako atazak idatzi egin dira. Proiektuaren aurkezpenerako, egindako lan guztia argi erakusteko bisualizazio euskarri bat garatu egin da. Aurreko faseetan bezala, 12.5. Taulan eta 12.6. Irudian jarduerek erabilitako denborak aurkeztu dira.

Taula 12.5: Dokumentazio faseko jarduerek erabilitako denboren taula. Iraupena laneguneta adierazita dago.

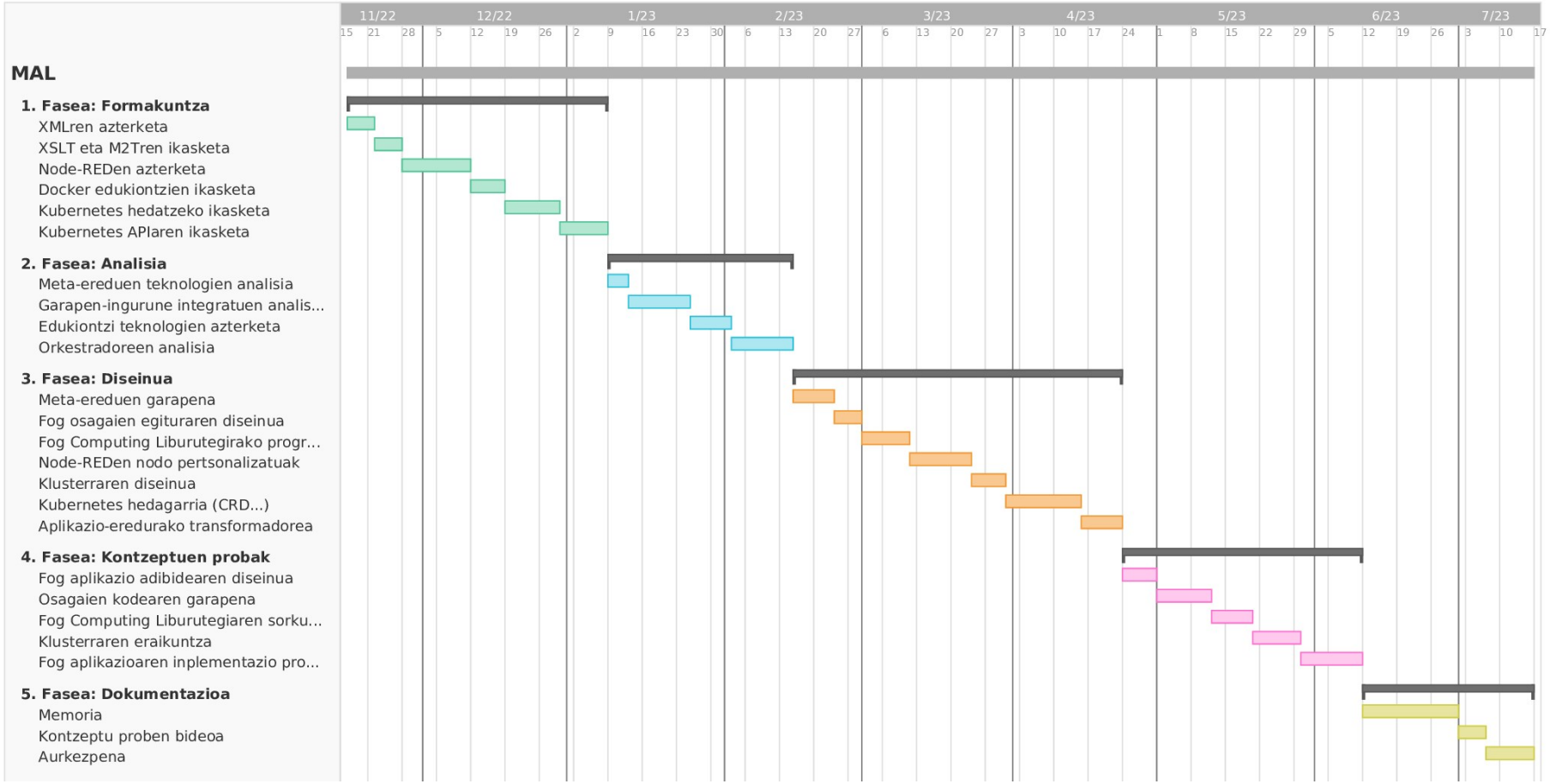
Kodea	Jardueraren deskribapena	Hasiera-data	Amaiera-data	Iraupena
F5	**Dokumentazioa**	**2023/06/12**	**2023/07/14**	**24**
J5.1	Memoria	2023/06/12	2023/06/29	14
J5.2	Kontzeptu proben bideoa	2023/06/30	2023/07/04	3
J5.3	Aurkezpena	2023/07/05	2023/07/14	7



Irudia 12.6: Dokumentazio fasean jardueren denboraren grafikoa.

13 Gantt diagrama

Ataza honetan proiektuaren planifikazioa aurkeztuko da. Horretarako, proiektu edo lan-andana bateko bilakaera erakusten duen Gantt diagrama erabiliko da. Diagramaren garapen eta diseinurako *teamGantt* tresna erabili da.



Irudia 13.1: Gantt diagrama

14 Emaitzen analisia

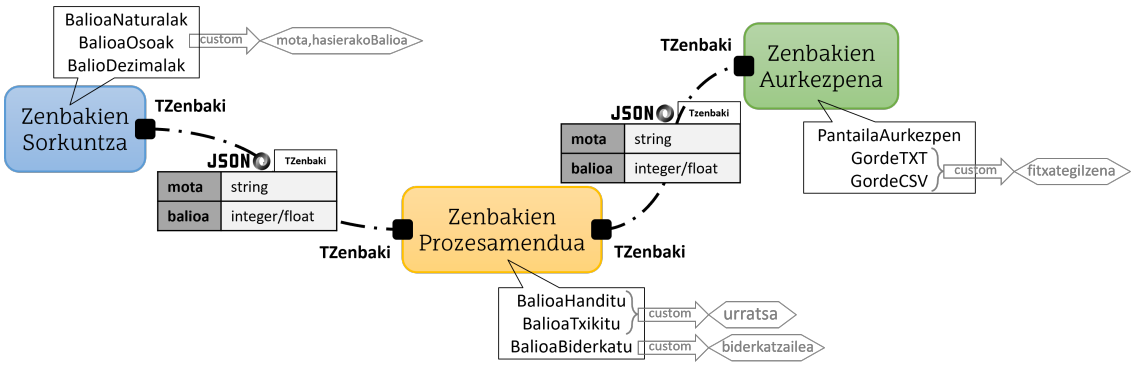
Fog aplikazioak diseinatzeko, garatzeko eta inplementatzeko ikuspegian proposatutako kontzeptuen probetarako, Fog aplikazio zehatz baten ideia aurrera eraman da. Aplikazio hori funtzionatzeko hiru Fog osagai garatu dira, bakoitzak hainbat funtzionalitate desberdin dituenak.

Kontzeptuen probarako funtzionamendu egokia dela konprobatu behar denez, eta metodologiak jarduera askoz osatuta dagoenez, zenbakiekin lan egiten duen aplikazio sinple bat diseinatu da.

Garatutako Fog osagaiak hiru izan dira. Bakarrik irteera duen osagaiari *ZenbakienSorkuntza* deitu diogu, C# programazio-lengoaian garatua izan dena. Horrek zenbakiak sortzeko ardura dauka, sorkuntza desberdineko zenbakietarako funtzionalitateak eskaintzen dituenak: balio naturalak, osoak edo dezimalak. Gainera, sorkuntza motarako ausazko zenbakiak, goranzko zenbakiak edo beheranzko zenbakiak sortzeko aukera eskaintzen du eta hasierako balioa zehaztea ahalbidetzen du.

Bestetik, sarrera eta irteera duen Pythonen garatutako osagaia *ZenbakienProzesamendua* izendatu da. Izenak dioen bezala, prozesamendu funtzionalitateak eskaintzen ditu: balioa handitu, balioa txikitu edo balioa biderkatu. Lehenengo bietan eragiketa bakoitzean beteko den urratsa zehaztu daiteke, eta azkenengoan, biderkatzailea.

Azkenik, bakarrik sarreradun osagaia *ZenbakienAurkezpena* deritzogu, Java lengoaian eraiki izan dena. Betiere lan fluxuaren azkenengo osagaia izango denez, zenbakien emaitzak kontsolatik erakusteko edo fitxategietan biltegiratzeko (bai .txt, bai .csv formatuan) funtzionalitateak eskaintzen ditu. Azkenengo kasurako, fitxategiaren izena zehaztu daiteke. Fog osagai guzti hauek, bidaltzen diren datu motekin batera, 14.1. Irudian grafikoki aurkeztu dira.



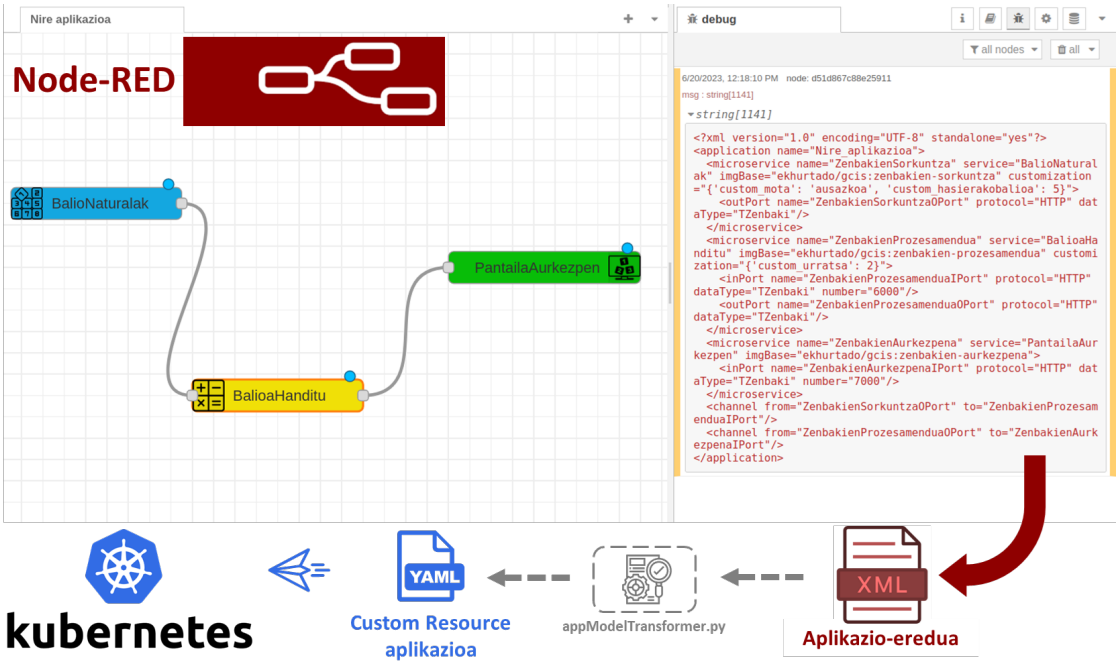
Irudia 14.1: Emaitzak aztertzeko garatutako Fog osagaien errepresentazio grafikoa, beraien arteko datu motaren egiturarekin batera.

Fog osagai horiek konbinatuz, hainbat Fog aplikazio desberdin sor daitezke. *ZenbakienProzesamendua* osagaia bat baino gehiagotan erabili daiteke aplikazio berdinean. Horrek, osagai bakoitzak eskaintzen duen pertsonalizazio handiarekin batera, Fog aplikazio desberdinak eraikitzeko aukera asko eskaintzen du, berrerabilpena eta malgutasuna sustatuz.

Fog osagaiak proposatutako metodologiarekin sortuta daudenez eta aipatutako tresnek prozesuaren zati handi bat automatizatzen dutenez, Fog aplikazioak zuzenak izatea ziurtatzen da. Gainera, elkarreragingarritasuna betetzen dela ziurtatzeko, Fog osagai bakoitzaren iturri-kodea sortzeko programazio-lengoaia desberdin bat erabili izan da.

Proposatutako ikuspegiaren azalpenean zehar, Fog osagai eta aplikazioen adibideak aurkeztu dira. Gainontzeko Fog osagaiak [40] errepositorioan daude eskuragarri. Aipatutako osagaiekin Fog Computing Liburutegia osatu da, prozesu automatiko hori zuzena dela konprobatuz.

Fog aplikazioen diseinu eta garapena konprobatzeko, Node-RED plataforman 9.4. Kodeko eredua lortzeko Fog osagaiak instantziatu eta pertsonalizatu dira. Aplikazio-eredu zuzena lortuta, inplementazio plataformarako *Custom Resource* fitxategia lortu da transformazio programa erabilita. Honakoa 14.2. Irudian erakutsi da, aplikazio-eredua sortzen denetik jarraitzen den prozesua adieraziz.



Irudia 14.2: Fog aplikazio adibidearen prozesua, IDE plataformatik software-plataformaraino.

Azter daitekeenez bezala, mikrozerbitzuen arteko mezuak JSON formatuan bidaliko dira, zenbakiaren mota eta balioa zehazteko. Hain zuzen ere, *TZenbaki* datu-motaren egitura da, osagai guztiek erabiltzen dutena. Aplikazioaren-diseinatzaileak diseinua amaitzerakoan,

Node-REDek automatikoki sortzen du eskuineko atalean agertzen den aplikazio-eredua. Testu formatuan egonez, diseinatzaileak edukia kopia dezake eta .xml fitxategi batean gorde.

Aplikazioa abiarazi nahi denean, diseinatzaileak aplikazio-eredu hori administratzaileari bidaltzen dio. Honek, transformadore-programa erabiliz, hedatze-fitxategi egokia lortzen du, zuzenean inplementazio-plataforman heda dezakeena.

Fog aplikazioaren definizio eta exekuzio egokia lortu delako, orokorrean, proiektuaren hasieran zehaztutako helburu guztiak lortu direla uste da. Metodologia funtzionala eta erraz erabiltzekoa izateko teknologia egokiak hautatu dira, eta kontzeptuen probak aipatutako ezaugarriak berresten dituzte.

Master Amaierako Lan honek proposatzen duen ikuspegia argitzeko, eta bete behar diren jarduera guztiak era grafiko batean aurkezteko euskarri bisual bat garatu da [43]. Bertan, erabiltzaile bakoitzaren (osagai-programatzaile, aplikazio-diseinatzaile eta sistemaren administratzailearen) lana ikus daiteke eta metodologia osoa zuzena dela eta interes-banaketa gauzatzen dela konproba daiteke. Gainera, Fog aplikazioen garapen, diseinu eta operazioa era egokian betetzen dela berresten da [1].

[1] Bideoa Youtube plataformako hurrengo estekan eskuragarri dago: `https://youtu.be/qo2UXgE1e1Y`

Atala IV

ALDERDI EKONOMIKOAK

15 Aurrekontuaren deskribapena

Memoriaren kapitulu honetan proiektuaren alderdi ekonomikoak aztertuko dira, aurrekontuaren deskribapena, hain zuzen ere. Ataza honetan bi zati nagusi bereiziko dira: lehenik, giza-baliabideek inbertitutako orduak (ikus 15.1. Taula); eta, bestetik, erabilitako materialen amortizazio ekonomikoa (ikus 15.2. Taula). Azkenik, aurreko bi ataletan lortutako emaitzekin proiektuaren kostu totala kalkulatuko da (ikus 15.3. Taula).

Esan bezala, hasteko giza-baliabideekin erlazionatutako kostuak aztertuko dira. Eskulanari dagokionez, proiektuan parte hartutako garapen taldeko kide bakoitzaren orduko kostuan kontuan hartu da. Hain zuzen ere, garapen taldeak hiru langilez osatuta dago, bakoitzak ardura bat edukiz, eta, beraz, orduko kostu desberdina. Master Amaierako Lan honek bi zuzendari izan dituenez, biak tutore bezala adierazi dira. Atal honetako kostuak 15.1. Taulan aurkeztu dira, $18750,00$€-ko kostu totalarekin.

Taula 15.1: Giza-baliabideek inbertitutako orduak.

Langilea	Kantitatea	Orduko kostua (€)	Inbertitutako orduak	Kostu totala (€)
Ingeniaria (ikaslea)	1	25	350	8750,00
Ingeniari irakaslea (tutorea)	2	40	100	8000,00
Proiektuaren zuzendaria	1	40	50	2000,00
	Totala			**18750,00**

Bigarren atalean, erabilitako materialen kostuak analizatuko dira. Proiektuaren garapena aurrera eramateko beharrezkoa den hardwareari dagokionez, garapen ia osoa software-motakoa izanik ordenagailu eramangarria ordu kopuru gehiengo gailua da. Fog aplikazioen exekuziorako inplementazio-plataforma osatzeko UPV/EHUko GCIS ikerketa-taldeko klusterra erabili da. Hori eraikitzeko, hainbat gailu behar izan dira. Azkenik, beste motako kalkuluak zehaztu dira, hala nola programen lizentziak, esaterako, JetBrains lizentzia IntelliJ edo Pycharm programak erabiltzeko.

Material bakoitzeko, urteko amortizazioa eta proiektuan daukan kostua kalkulatuko da. Kontuan hartu behar da proiektuaren iraupena 168 egunekoa izan dela, eta esfortzu totala 350 ordukoa. Kalkuluen emaitzak lortzeko, honako ekuazio matematikoak erabiliko dira:

$$
\begin{gathered}
Urteko\,amortizazioa = x/y \\
non\,x = unitate\,bakoitzeko\,kostua\,eta\,y = bizitza\,urteetan \\
Kostua\,proiektuan = \alpha \cdot \beta \cdot \gamma \cdot \delta \\
non\,\alpha = kantitatea,\ \beta = iraupena,\ \gamma = esfortzua\,eta\,\delta = amortizazioa
\end{gathered}
\tag{15.1}
$$

Adibide bezala, klusterra osatzen duten Dell ordenagailuek proiektuan daukaten kostuaren kalkuluak hurrengo ekuazioetan aurkeztu dira:

$$
\begin{aligned}
Urteko\,amortizazioa &= \frac{300}{5} = 60 \\
Kostua\,proiektuan &= 4 \cdot \frac{168egun}{365egun} \cdot \frac{200h}{350h} \cdot 60 = 63,12€
\end{aligned}
\tag{15.2}
$$

Material guztien informazioa, eta aipatutako kalkuluen emaitzak 15.2. Taulan adierazi dira.

Taula 15.2: Proiektuan erabilitako materialen amortizazioak.

	Materiala	Kantitatea	Unitate bakoitzeko kostua (€)	Bizitza (urteetan)	Erabilera (orduetan)	Kostua (€)
Kluster	Ordenagailua	1	1500	6	350	115,07
	Dell Optiplex 780	4	300	5	200	63,12
	Komunikazio kommutadorea	1	60	5	200	3,15
	Sare-kableak	5	15	6	200	3,29
	Elikatze-kableak	4	10	5	200	2,10
	Pantaila	1	70	4	200	4,60
	Besteak (JetBrains lizentzia...)	1	100	1	150	19,73
	Totala					**211,06**

Azkenengo atal bezala, proiektuaren kostu totala kalkulatuko da. Horretarako, aurreko ataletako kostuez gain, kostu ez-zuzenak ere hartuko dira kontuan. Kostu horiekin erlazionatutako baliabideek ez dute zuzenean lanaren garapenean eragiten. Hala ere, beharrezkoak dira bai hardware bai software baliabideak zuzen funtzionatu eta erabiltzeko.

Horren adibide argia elektrizitatea da. Kostu ez-zuzenak gainontzeko kostuen baturaren %2a zehaztu dira.

Kostu totalen laburpena 15.3. Taulan adierazi da. Giza-baliabideen kostuak ($18750,00$ €), materialen kostuak ($211,06$ €) eta kostu ez-zuzenak ($379,22$ €) batuz, MALaren kostu totala $19340,28$ €-koa izan da.

Taula 15.3: MALaren kostu globalen laburpenerako taula.

Kalkulu globalak	
Kontzeptua	**Kostua (€)**
Inbertitutako orduak	$18750,00$
Materialen amortizazioa	$211,06$
Subtotala	$18961,06$
Kostu ez zuzenak (%2)	$379,22$
Totala	**19340,28**

Atala V

ONDORIOAK

16 Ondorioak

Memorian zehar aztertu denez, garatutako lana hainbat ataletan bete da. Hori dela eta, atal bakoitzean ondorio desberdinak lortu dira. Kapitulu honetan, proiektuan lortutako ondorio horiek aztertuko dira, baita amaieran iritsi den ondorio globala ere.

Lan honek bi meta-eredu proposatu ditu: bat Fog osagaientzako eta beste bat Fog aplikazioentzako. Lehenengoak Fog osagaiak Fog aplikazioengandik era independentean definitzea ahalbidetzen du, eta bigarrenak, berriz, aplikazioa osatuko dituzten mikrozerbitzuak eta beraien arteko interkonexioak definitzea ahalbidetzen du, hau da, aplikazioaren lan-fluxua.

Era honetan, osagaien eta aplikazioaren logikaren definizioak banatzea lortzen da, bai logika, bai Fog osagaien berrerabilpena sustatzen den aldi berean. Meta-ereduen inplementaziorako XML teknologia erabilita, osagai eta aplikazio ereduen koherentzia baliozkotzea ahalbidetzen da, domeinuarekin independentzia eskaintzeaz gain.

Fog osagaien kasuan, Docker teknologiaren aukeraketa erabateko arrakasta izan dela aztertu da. Programatzaileei askatasun handia eskaintzen die iturri-kodea garatzeko, edozein programazio-lengoaia erabil baitezakete. Hala ere, lan gainkargak eta lan honen proposamenaren murrizketek osagaien programatzaileei eragiten diete batez ere, iturri-kodea meta-ereduak zehazten duen erara parametrizatzera behartuta baitaude.

Hala ere, murrizketa horrek abantailak ere ekar ditzake, kode berdinaren barruan hainbat funtzionalitate garatzeko aukera eskaintzen baita. Horrek baliagarria izan daiteke, esaterako, protokolo ezberdinen bidez komunikatu daitezkeen funtzionalitateak osagai bakarrean multzokatzeko. Gainera, hainbat funtzionalitate Fog osagai berean programatu daitezke, iturri-kodearen zati funtzionalak berrerabiltzea posible eginez.

Bestetik, programatzaileek Fog Computing Liburutegiko nodoak sortzeko ardura daukate, prozesua automatikoki betetzen dena. Horrek osagaien ereduen bidez kolaboratu dezaketen osagai-programatzaileen talde ezberdinen arteko elkarlana hobetzen du, iturri-kodean sartu beharrik gabe.

Fog aplikazioekin erlazionatutako teknologiak aztertuz, Node-RED plataformaren erabilera abantaila asko eskaini dizkio proposatutako ikuspegiari. Pertsonalizaziorako aukera anitz edukiz, osagai-programatzaileen eta aplikazio-diseinatzaileen arteko interes-banaketa lortu da, Fog Computing Liburutegiaren bitartez. Nodo pertsonalizagarrien integrazioaren bidez diseinatzaileak ez du zertan osagaiak nola sortu diren jakin behar, eta hori ezinbestekoa da diseinu mailan konplexutasuna murrizteko.

Bestalde, Node-REDek interfaze oso egoki bat aurkezten du, bai lan-fluxuak bezalako aplikazioak eraikitzeko, bai osagai bakoitza pertsonalizatzeko, erabiltzaileak ekintza sinple eta intuitiboak gauzatu behar baititu. Gainera, gehitutako automatizazioagatik, Fog aplikazioen diseinuak eta garapenak esfortzu txikia eskatzen dio aplikazio-diseinatzaileari.

Lan honetan inplementazio-plataforma bezala Kubernetes hautatu da. Docker kontainerretan oinarritutako mikrozerbitzuez osatutako aplikazioak exekutatu eta kudeatzeko plataforma egokiena dela berretsi da. Kubernetesek berez eskaintzen dituen abantailez gain (nodo bat erortzen bada *Pod* guztiak beste batera eramaten ditu, plataforma bere kabuz orekatzen du, etab.), kontrolagailuen gehitzeak funtzionaltasun maila handiagoa eman dio. Hala nola, plataformak aplikazio-eredutik osagaiak lortu eta abiarazteko gai da, fluxuak funtziona dezan beharrezko datuak gehitzeaz arduratzen da eta komando-lerroko interfazearekin sistema erraz kudeatzeko aukera ematen du.

Egileek dakitenez, edukiontzi-orkestratzaileek ez dute bere osagaiak multzokatzen aplikazio gisa, beraz, abantaila handia da Fog aplikazioa Kubernetesen entitate propio gisa gehitzea. Aplikazioa eta mikrozerbitzua lehen mailako entitate bezala sortzeari esker, Fog osagaiak ondo definituta daude eta aplikazio baten parte dira. Horrekin lotuta, meta-ereduen erabilerarekin aplikazioaren definizio zuzena bermatzen denez, eta *Custom Resources* elementuetarako eraldaketa automatikoki egiten denez, kontrolagailuek aplikazioa plataforman behar bezala zabaltzea bermatzen dute.

Orokorrean, MAL honetan proposatu diren helburu guztiak lortu direla ondoriozta daiteke. Metodologian parte hartzen duten hiru taldeen arteko interes-banaketa lortu da. Bakoitzak bakarrik hautatutako teknologien ezagutza eduki behar du. Hala nola, osagai-programatzaileak programazio (iturri-koderako), Docker (oinarri-irudia sortzeko) eta Node-REDeko nodoen egituraren (Fog Computing Liburutegirako) ezagutza beharko du; aplikazio-diseinatzaileak soilik Node-RED plataformaren ezagutzaren beharra du; eta, inplementazio-sistemaren administratzaileak Kubernetes kudeatzen jakin behar du.

Beste alde batetik, kontuan hartu behar da proposatu diren txantiloi eta baliabideetarako, batez ere Docker oinarri hartu dela. Hala ere, edukiontzi edo beraien orkestraziorako beste edozein tresna erabiliz gero, proposatutako ikuspegia baliagarria da, M2T transformazioak eta inplementazio-plataforma aldatzea besterik ez baita beharrezkoa.

17 Etorkizunerako garapen lerroak

Proiektu honetan garatu den lana hainbat aukera berri ireki ditu. Teknologien hautabideak aztertzerakoan, pertsonalizazio handikoak direla aurkitu da, ia edozein ideia aurrera eramateko baliabideak eskaintzen baitituzte. Beste alde batetik, garrantzitsua da lan honek dituen mugak ezagutzea, aipatutako teknologiek ahalbidetutako aukerekin muga horiek gainditzeko garapen-lerroak zehazteko.

MAL honen proposamenaren muga bat Fog osagai bera aplikazio desberdinetan erabiltzeko kasuan, mikrozerbitzu adina edukiontzi sortzen direla, hau da, aplikazio bakoitzeko edukiontzi bat sortzen dela, oinarri-irudi berdinarekin (pertsonalizazioa berdina edo desberdina izanda ere). Horrek baliabideak xahutzea dakar, edukiontzi bera, pertsonalizazioa berdina bada, aplikazio guztiek parteka baitezakete.

Hori dela eta, hurrengo urratsak izango diren garapen lerro bat aplikazioaren bizizikloa kudeatzeko gai den Fog arkitektura bat garatzera bideratuko da, hainbat aplikaziok edukiontzidun mikrozerbitzu batek eskaintzen dituen zerbitzuak partekatzea ahalbidetuz. Honakoa inplementazio-plataformaren ardura izanez, Kubernetes hedagarria aldatu beharko litzateke, garatutako kontroladoreek aukera berri hau ahalbidetu dezaten.

Proiektu honen beste mugetako bat aurrekoarekin erlazionatuta dago. Garatu den proposamenak aplikazio sinpleak soilik sortzeko aukera eskaintzen du, hau da, lan-fluxua gehienez sarrera eta/edo irteera bakarreko osagaiez osatuta egon daiteke. Fog osagai bat aplikazio bat baino gehiagotan egotea posible eginez, aplikazio konplexuagoak sor daitezke. Etorkizuneko lana meta-ereduen eta Node-REDen aldaketak aztertzea izango litzateke, osagai bat aplikazio bakar batetik edo aplikazio anitzetan parte hartzen duen osagai batetik bereizteko gai izan daitezen.

Aplikazio konplexuak sortzeko ideia berdinarekin, Fog osagaiak funtzionalitate anizkoitz eskaintzeko garapen lerro bat zehaztu daiteke. Horrela, diseinatzaileak ez luke funtzionalitate zehatz bat aukera beharko, baizik eta sisteman zein Fog osagai behar diren eta beraien arteko erlazioak ezarri. Kasu honetan, lan-fluxuko osagaiek hurrengokoari eskaintzen dituen zerbitzuen artean behar dutena eskatuko lukete. Garapen lerro honetarako metodologiaren zati gehienak berrantolatu beharko dira, Fog aplikazioaren kontzeptua bera aldatzen baita.

Bestalde, metodologia honekin aplikazio baten osagai bakarra eguneratzeko zailtasunak ager daitezke. Esaterako, diseinatzailearen erabaki baten ondorioz funtzionalitate-aukeraketa bat aldatzeko, Node-RED fluxua soilik aldatu behar du, eta aplikazio-eredu berria automatikoki sortuko da. Kubernetesen zabaltzerakoan, eredua zuzena izanda ere,

kontroladoreek ez dute aplikazioak exekuzio-denboran aldatzeko gaitasunik. Horregatik, garapen lerro bezala proposatutako kontroladoreak hobe daitezke, aplikazioen aldaketak exekuzioa eten barik bete daitezen. Mikrozerbitzuak sortzeko erabili diren teknologiengatik, Docker batez ere, honakoa egingarria da, soilik martxan dagoen edukiontziaren pertsonalizazioa aldatu beharko litzatekeelako.

Bertsio berri baten ondorioz Fog osagaia bera aldatu behar bada, programatzaileak irudia birsortu eta errepositorioan biltegiratu beharko du. Hala ere, Fog Computing Liburutegian integratzeko prozesu guztia errepikatu beharko du. Aurreko mugarekin erlazionatuta, kontroladoreak hobetzeaz gain Node-RED plataforma hobe daiteke, aldaketaren bat egonez gero prozesu guztia errepikatzea ekiditeko.

BIBLIOGRAFIA

[1] Grupo Carman. *Historia de la ingenieria*. 2019. URL: https://grupocarman.com/blog/2019/03/11/historia-de-la-ingenieria/.

[2] Euskomedia Fundazioa. *Industria-iraultza*. 2008. URL: http://industrializazioa.eusko-ikaskuntza.eus/industriairaultza.html.

[3] Josef Taalbi. "Origins and pathways of innovation in the third industrial revolution1". Non: *Industrial and Corporate Change* (2018). DOI: 10.1093/icc/dty053.

[4] Redowan Mahmud, Adel N. Toosi, Kotagiri Ramamohanarao eta Rajkumar Buyya. "Context-Aware Placement of Industry 4.0 Applications in Fog Computing Environments". Non: *IEEE Transactions on Industrial Informatics* 16.11 (2020), orrk. 7004-7013. DOI: 10.1109/TII.2019.2952412.

[5] Frank Siqueira eta Joseph G. Davis. "Service Computing for Industry 4.0: State of the Art, Challenges, and Research Opportunities". Non: *ACM Computing Surveys (CSUR)* 54 (2021), orrk. 1-38. DOI: 10.1145/3478680.

[6] Karsten Schweichhart. *Reference Architectural Model Industrie 4.0 (RAMI 4.0)*. 2016. URL: https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf.

[7] C. E. Belman-Lopez, J. A. Jiménez-García eta S. Hernández-González. "Análisis exhaustivo de los principios de diseño en el contexto de Industria 4.0". Non: *Revista Iberoamericana de Automática e Informática industrial* 17.4 (2020), orr. 432. DOI: 10.4995/riai.2020.12579.

[8] Shiyong Wang, Jiafu Wan, Di Li eta Chunhua Zhang. "Implementing Smart Factory of Industrie 4.0: An Outlook". Non: *International Journal of Distributed Sensor Networks* 12.1 (2016), orr. 3159805. DOI: 10.1155/2016/3159805.

[9] E. Hurtado, A. López, A. Armentia et al. "On the Development of Fog-Edge Feedback Applications". Non: *IEEE 17th International Conference on Automation Science and Engineering* (2021).

[10] Andreas Seitz, Dominic Henze, Daniel Miehle et al. "Fog Computing as Enabler for Blockchain-Based IIoT App Marketplaces - A Case Study". Non: (2018), orrk. 182-188. DOI: 10.1109/IoTSMS.2018.8554484.

[11] Paolo Bellavista eta Alessandro Zanni. "Feasibility of Fog Computing Deployment based on Docker Containerization over RaspberryPi". Non: *Proceedings of the 18th International Conference on Distributed Computing and Networking*. Hyderabad India: ACM, 2017, orrk. 1-10. DOI: 10.1145/3007748.3007777.

[12] E. Hurtado, A. López, et al. “Diseño basado en modelos de aplicaciones Fog como workflow de microservicios”. Non: *XLII Jornadas de Automática* (2021). DOI: https://doi.org/10.17979/spudc.9788497498043.701.

[13] Ivan Stojmenovic eta Sheng Wen. “The Fog computing paradigm: Scenarios and security issues”. Non: *Federated Conference on Computer Science and Information Systems* (2014), orrk. 1-8. DOI: 10.15439/2014F503.

[14] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito eta Omer Rana. “Fog computing for the Internet of Things: A survey”. Non: *ACM Transactions on Internet Technology* 19 (2019), orrk. 1-41. DOI: 10.1145/3301443.

[15] Julen Cuadra, Ekaitz Hurtado, Federico Pérez, Oskar Casquero eta Aintzane Armentia. “OpenFog compliant application aware platform: a Kubernetes extension”. Non: *Applied Sciences* 13 (2023). Accepted for Publication.

[16] IEEE Standards Association. *IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*. 2018. URL: https://ieeexplore.ieee.org/document/8423800.

[17] OASIS Open standards. *TOSCA Simple Profile in YAML Version 1.3*. 2020. URL: https://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.3/os/TOSCA-Simple-Profile-YAML-v1.3-os.html.

[18] Muhammad Waseem, Peng Liang, Mojtaba Shahin, Amleto Di Salle eta Gastón Márquez. “Design, Monitoring, and Testing of Microservices Systems: The Practitioners’ Perspective”. Non: *Journal of Systems and Software* 182 (2021). DOI: 10.1016/j.jss.2021.111061. arXiv: 2108.03384[cs].

[19] Maria Fazio, Antonio Celesti, Rajiv Ranjan et al. “Open Issues in Scheduling Microservices in the Cloud”. Non: *IEEE Cloud Computing* 3.5 (2016), orrk. 81-88. DOI: 10.1109/MCC.2016.112.

[20] Nam Ky Giang, Rodger Lea eta Victor C.M. Leung. “Developing applications in large scale, dynamic fog computing: A case study”. Non: *Software - Practice and Experience* 50 (2020), orrk. 519-532. DOI: 10.1002/SPE.2695.

[21] R. Dintén, P. López Martínez eta M. Zorrilla. “Arquitectura de referencia para el diseño y desarrollo de aplicaciones para la Industria 4.0”. Non: *Revista Iberoamericana de Automática e Informática industrial* 18.3 (2021), orr. 300. DOI: 10.4995/riai.2021.14532.

[22] Xiang He, Zhiying Tu, Xiaofei Xu eta Zhongjie Wang. “Programming framework and infrastructure for self-adaptation and optimized evolution method for microservice systems in cloud–edge environments”. Non: *Future Generation Computer Systems* 118 (2021), orrk. 263-281. DOI: 10.1016/J.FUTURE.2021.01.008.

[23] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos et al. “Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions”. Non: *IEEE Access* 6 (2018), orrk. 47980-48009. DOI: 10.1109/ACCESS.2018.2866491.

[24] Rabeea Basir, Saad Qaisar, Mudassar Ali et al. “Fog Computing Enabling Industrial Internet of Things: State-of-the-Art and Research Challenges”. Non: *Sensors* 19.21 (2019), orr. 4807. DOI: 10.3390/s19214807.

[25] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning eta Tie Qiu. “Survey on fog computing: architecture, key technologies, applications and open issues”. Non: *Journal of Network and Computer Applications* 98 (2017), orrk. 27-42. DOI: 10.1016/j.jnca.2017.09.002.

[26] Nam Ky Giang, Michael Blackstock, Rodger Lea eta Victor C.M. Leung. “Developing IoT applications in the Fog: A Distributed Dataflow approach”. Non: *2015 5th International Conference on the Internet of Things (IOT)*. Seoul, South Korea: IEEE, 2015, orrk. 155-162. DOI: 10.1109/IOT.2015.7356560.

[27] Harshit Gupta, Amir Vahid Dastjerdi, Soumya K. Ghosh eta Rajkumar Buyya. “iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments: iFogSim: A toolkit for modeling and simulation of internet of things”. Non: *Software: Practice and Experience* 47.9 (2017), orrk. 1275-1296. DOI: 10.1002/spe.2509.

[28] Ivan Alfonso, Kelly Garces, Harold Castro eta Jordi Cabot. “Modeling self-adaptative IoT architectures”. Non: *2021 ACM/IEEE International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C)*. Fukuoka, Japan: IEEE, 2021, orrk. 761-766. DOI: 10.1109/MODELS-C53483.2021.00122.

[29] Vrani Ibarra-Junquera, Apolinar Gonzalez-Potes, Carlos Mario Paredes, Diego Martinez-Castro eta Rubi A. Nunez-Vizcaino. “Component-Based Microservices for Flexible and Scalable Automation of Industrial Bioprocesses”. Non: *IEEE Access* 9 (2021), orrk. 58192-58207. DOI: 10.1109/ACCESS.2021.3072040.

[30] Matteo Bogo, Jacopo Soldani, Davide Neri eta Antonio Brogi. “Component-aware Orchestration of Cloud-based Enterprise Applications, from TOSCA to Docker and Kubernetes”. Non: *Software: Practice and Experience* 50.9 (2020), orrk. 1793-1821. DOI: 10.1002/spe.2848. arXiv: 2002.01699[cs].

[31] Docker Inc. *Docker*. 2022. URL: https://www.docker.com/.

[32] Adrián Orive, Aitor Agirre, Hong Linh Truong, Isabel Sarachaga eta Marga Marcos. “Quality of Service Aware Orchestration for Cloud-Edge Continuum Applications”. Non: *Sensors* 22 (2022), orrk. 1755-1775. DOI: 10.3390/S22051755.

[33] Kubernetes. *Kubernetes*. 2022. URL: https://kubernetes.io/.

[34] Leila Abdollahi Vayghan, Mohamed Aymen Saied, Maria Toeroe eta Ferhat Khendek. “A Kubernetes controller for managing the availability of elastic microservice based stateful applications”. Non: *Journal of Systems and Software* 175 (2021), orrk. 1-13. DOI: 10.1016/J.JSS.2021.110924.

[35] Yue Wang, Choonhwa Lee, Shuyang Ren, Eunsam Kim eta Sungwook Chung. “Enabling Role-Based Orchestration for Cloud Applications”. Non: *Applied Sciences* 11 (2021), orrk. 6656-6678. DOI: 10.3390/APP11146656.

[36] Merlijn Sebrechts, Sander Borny, Tim Wauters, Bruno Volckaert eta Filip De Turck. “Service relationship orchestration: Lessons learned from running large scale smart city platforms on kubernetes”. Non: *IEEE Access* 9 (2021), orrk. 133387-133401. DOI: 10.1109/ACCESS.2021.3115438.

[37] Eddy Truyen, Dimitri Van Landuyt, Davy Preuveneers, Bert Lagaisse eta Wouter Joosen. “A Comprehensive Feature Comparison Study of Open-Source Container Orchestration Frameworks”. Non: *Applied Sciences* 9.5 (2019), orr. 931. DOI: 10.3390/app9050931.

[38] W3C (World Wide Web Consortium). *W3C XML Schema Definition Language (XSD)*. 2012. URL: https://www.w3.org/TR/xmlschema11-1/.

[39] N Md Jubair Basha, Dr Gopinath Ganapathy eta Dr Mohammed Moulana. “A Preliminary Exploration on Component Based Software Engineering”. Non: *International Journal of Computer Science and Network Security* 22.9 (2022), orrk. 143-148. DOI: 10.22937/IJCSNS.2022.22.9.22.

[40] GCIS UPV/EHU. *Proiektuan garatutako kodearen errepositorioa*. URL: https://github.com/ekhurtado/EkaitzHurtado-MAL.

[41] OpenJS Foundation. *Node-RED*. 2022. URL: https://nodered.org/.

[42] Kubernetes. *Kubernetes*. 2022. URL: https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/.

[43] Ekaitz Hurtado. “Ekaitz Hurtadoren MALerako euskarri bisuala [Bideo online]”. Non: *https://youtu.be/qo2UXgE1e1Y* (2023).