UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS E INFORMÁTICA

PLAN DE TRABAJO DE GRADO

FECHA DE PRESENTACIÓN: Bucaramanga, 8 de enero de 2023

TÍTULO: Mecanismos de adaptación autonómica de arquitectura software para la plataforma Smart Campus UIS

MODALIDAD: Trabajo de investigación

AUTOR: Daniel David Delgado Cervantes, 2182066

DIRECTOR: PhD. Gabriel Rodrigo Pedraza Ferreira, Escuela De Ingeniería De Sistemas e Informática

CODIRECTOR: MSc. Henry Andrés Jiménez Herrera, Escuela De Ingeniería De Sistemas e Informática

ENTIDAD INTERESADA: Universidad Industrial de Santander

TABLA DE CONTENIDO

MECANISMOS DE ADAPTACIÓN AUTONÓMICA DE ARQUITECTURA SOFTWARE PARA LA PLATAFORMA SMART CAMPUS UIS

1 INTRODUCCIÓN

Dentro de la computación distribuida, una de las tendencias recientes dentro de la industria, es la búsqueda de maneras de reducir la complejidad de administrar los sistemas computacionales. A medida que estos crecen, en términos de tamaño y extensión, el costo humano de la administración de los sistemas se vuelve insostenible. En respuesta a esto, diferentes enfoques han surgido. Uno de estos es el *autonomic computing* (computación autonómica). Planteada originalmente por IBM en el año 2001, se presentaba como una posible solución a la problemática a partir del diseño y la implementación de componentes auto-gestionados (Kephart, 2011).

Este acercamiento, aunque inicialmente concebido únicamente para sistemas distribuidos, puede también ser particularmente útil en otras ramas como lo son las

2 PLANTEAMIENTO Y JUSTIFICACIÓN DEL PROBLEMA

El crecimiento de las capacidades de los sistemas de software en términos de la escala, tienen como consecuencia el aumento en la complejidad, heterogeneidad e incertidumbre (Salehie y Tahvildari, 2005). El manejo de una gran cantidad de estos componentes es un reto en términos de recursos técnicos y humanos (Horn, 2001, pp. 4-5).

Una de las posibles maneras de dar solución a esta problemática, en cuanto al manejo de sistemas, está en el área de la computación autonómica. Este acercamiento basado en conceptos biológicos busca solventar los problemas de complejidad, heterogeneidad e incertidumbre (Salehie y Tahvildari, 2005) a partir de la abstracción de las metas de los administradores y delegación del manejo del sistema a sí mismo (Lalanda, Diaconescu, y McCann, 2014).

La propuesta de IBM, responsables de los primeros acercamientos a la computación autonómica, es especialmente interesante en los casos de las arquitecturas de software orientadas a microservicios, uno de los patrones de diseño más usados (Forrester Research, 2019); al igual los sistemas embebidos se hacen más presentes en la industria (Deichmann, Doll, Klein, Mühlreiter, y Stein, 2022); los cuales afrontan una gran cantidad de incertidumbre.

Considerando lo anterior, una de las aplicaciones de los sistemas está en la IoT y los Smart Campus, una variación de las Smart Cities en las cuales se busca la recolección de información y monitoreo en tiempo real con el fin de apoyar la toma de decisiones, mejora de servicios, entre otros (Min-Allah y Alrashed, 2020). Es en este tipo de aplicaciones, debido incertidumbre generada por la cantidad de componentes distribuidos en puntos geográficamente apartados, que la computación autonómica se presenta un gran potencial.

Dentro del marco del presente proyecto, se tiene Smart Campus SC3, una plataforma de IoT de la Universidad Industrial de Santander, en la cual se han realizado implementaciones parciales de una arquitectura autonómica con capacidad de auto-describirse al igual que un avance a la auto-sanación (Jiménez, Cárcamo, y Pedraza, 2020), características principales de un sistema autonómico (Horn, 2001).

Dicho esto, y en búsqueda dar continuidad con los esfuerzos de desarrollo realizados en la plataforma, se plantea como caso de estudio la implementación de mecanismos de adaptación con los cuales se pueda alterar el estado de la plataforma a partir de un objetivo establecido.

3 OBJETIVOS

3.1 OBJETIVO GENERAL

- Implementar un conjunto de mecanismos autonómicos para permitir la adaptación de la Arquitectura Software IoT respecto a un modelo objetivo en la plataforma Smart Campus UIS

3.2 OBJETIVOS ESPECÍFICOS

- Proponer una notación (lenguaje) para describir una arquitectura objetivo de un sistema software IoT.
- Diseñar un mecanismo para determinar las diferencias existentes entre una arquitectura actual en ejecución y una arquitectura objetivo especificada.
- Diseñar un conjunto de mecanismos de adaptación que permitan disminuir las diferencias entre la arquitectura actual y la arquitectura objetivo.
- Evaluar la implementación realizada a partir de un conjunto de pruebas con el fin de establecer la efectividad de los mecanismos usados.

4 MARCO DE REFERENCIA

Como base para el desarrollo del proyecto es necesario establecer los fundamentos para la selección de los mecanismos de adaptación, por lo cual es indispensable conocer los principios de la computación autonómica, las aplicaciones de la misma en la industria y las partes requeridas para la integración, las cuales se describen a continuación.

4.1 COMPUTACIÓN AUTONÓMICA

El concepto de computación autonómica, definido inicialmente por IBM (2001), se refiere a un conjunto de características que presenta un sistema computacional el cual le permite actuar de manera autónoma, o auto-gobernarse, con el fin de alcanzar algún objetivo establecido por los administradores del sistema (Lalanda y cols., 2014).

Los 8 elementos clave, definidos por IBM, que deberían presentar este tipo de sistemas son:

1. Auto-conocimiento: habilidad de conocer su estado actual, las interacciones del sistema.
2. Auto-configuración: capacidad de reconfigurarse frente a los constantes cambios en el entorno.
3. Auto-optimización: búsqueda constante de optimizar el funcionamiento de sí mismo.
4. Auto-sanación: aptitud de restaurar el sistema en el caso de que se presenten fallas.
5. Auto-protección: facultad de protegerse a sí mismo de ataques externos.
6. Auto-conciencia: posibilidad de conocer el ambiente en el que el sistema se encuentra.
7. Heterogeneidad: capacidad de interactuar con otros sistemas de manera cooperativa.
8. Abstracción: ocultar la complejidad a los administradores del sistema con objetivos de alto nivel de abstracción.

En el caso de que un sistema tenga una implementación parcial de estas características, este podría considerarse autonómico. En este sentido debería tener la capacidad de lidiar con los problemas como la complejidad, heterogeneidad e incertidumbre (Salehie y Tahvildari, 2005) al igual que reducir la cantidad de recursos tanto técnicos como humanos requeridos para mantener los sistemas en funcionamiento.

MAPE-K

IBM, en cuanto a la implementación de las características, propone una modelo de ciclo auto-adaptativo, denominado MAPE-K (Krikava, 2013). Este acercamiento, compuesto de cinco fases, es uno de ciclos de control más usado en implementaciones de sistemas auto-adaptativos y computación autonómica (Arcaini, Riccobene, y Scandurra, 2015). En la figura 1, se presentan las fases que *manejador* debe desarrollar para así administrar cada uno de los elementos del sistema computacional basado en una base de conocimiento común (Gorla, Pezzè, Wuttke, Mariani, y Pastore, 2010).

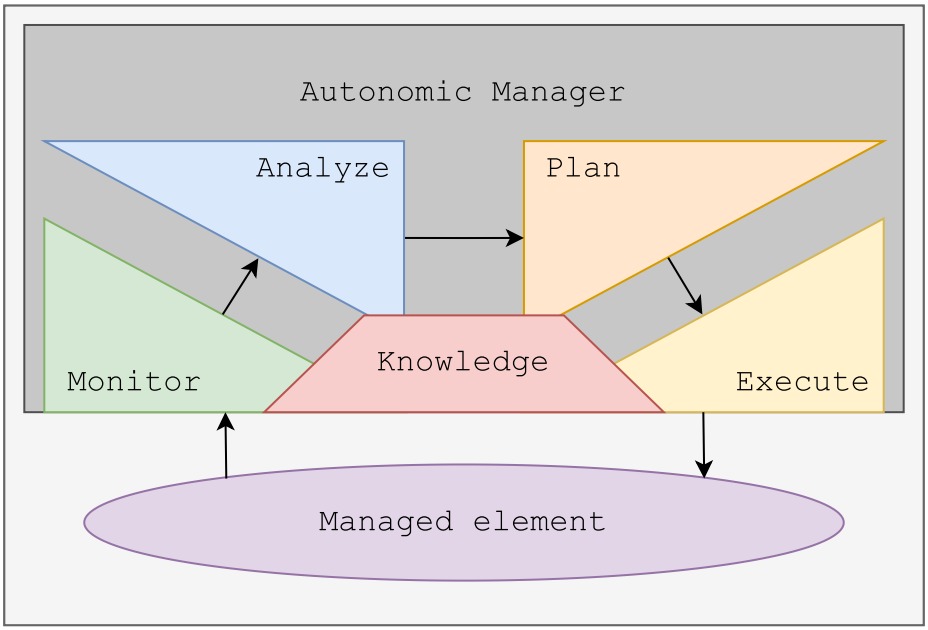

Figura 1: El ciclo auto-adaptativo MAPE-K.
(Gorla y cols., 2010)

Cada una de estas fases son:

- Monitorear (M): Esta fase se compone de la recolección, filtración y reportar la información adquirida sobre el estado del elemento a manejar.
- Analizar (A): La fase de análisis se encarga del interpretar el entorno en el cual se encuentra, el predecir posibles situaciones comunes y diagnosticar el estado del sistema.
- Planear (P): Durante la planificación se determina las acciones a tomar con el fin de llegar a un objetivo establecido a partir de una serie de reglas o estrategias.
- Ejecutar (E): Finalmente, se ejecuta lo planeado usando los mecanismos disponibles para el manejo del sistema.

Es de resaltar que este modelo, aunque útil para el desarrollo de este tipo de sistemas, es bastante general en cuanto a la estructura y no usan modelos de diseño establecidos (Ouareth, Boulehouache, y Mazouzi, 2018).

MECANISMOS DE DESCRIPCIÓN

La fase de monitoreo dentro del ciclo MAPE-K es vital para el funcionamiento del manejador autonómico pues es a partir de la información que se construirá la base de conocimiento requerida por las demás partes del ciclo. Parte de esta, está compuesta por el *estado del sistema* el cual incluye la descripción del sistema en un momento dado (Weiss, Zeller, y Eilers, 2011).

Existen varias maneras de realizar implementaciones de mecanismos de auto-descripción y la utilidad de cada uno de estos varía dependiendo en el tipo de sistema de software que se esté usando. Para el marco del proyecto, nos interesan aquellos que estén orientados a los sistemas embebidos e IoT, algunos de estos son:

- **JSON Messaging**: Iancu y Gatea (2022) plantean un protocolo que emplea mensajería entre *gateways* con el fin de recibir información sobre estas. En términos simples, estas funcionan como un *ping* hacia el nodo que luego retorna sus datos, al igual que los dispositivos conectados a ella, al encargado de recolectar toda esta información con el fin de construir una descripción del sistema.-
- **IoT Service Description Model**: O IoT-LMsDM, es un servicio de descripción desarrollado por Zhen y Aiello (2021) el cual está orientado al contexto, servicios e interfaz de un sistema IoT. De este se espera poder contar no solo con descripciones del estado del sistema en términos del ambiente, pero la funcionalidad (es decir, los *endpoints* a usar) al igual que las estructuras de datos que estos consumen.
- Henry!!!

De esto podemos ver no solo las diferentes maneras en las que las implementaciones realizan las descripciones de los sistemas asociados, sino que también el alcance de estos en cuanto a lo que pueden describir.

MECANISMOS DE ADAPTACIÓN

La adaptación, en el contexto de la computación autonómica, es la parte más importante en cuanto a la auto-gestión de un sistema de software se refiere. Así mismo, presenta el mayor reto debido a la necesidad de modificar código de bajo nivel, tener que afrontarse a incertidumbre de los efectos que pueden tener dichas alteraciones al sistema al igual que lidiar con esto en *runtime* debido a los problemas que el *downtime* tendría en los negocios (Lalanda y cols., 2014).

Esta adaptabilidad puede exponerse en múltiples puntos dentro de un sistema de software. Pueden realizarse adaptaciones en sistema operativo, lenguaje de programación, arquitectura e incluso datos (Lalanda y cols., 2014).

Manteniéndose en el marco del proyecto, son las implementaciones relacionadas con la modificación de la arquitectura los cuales nos interesan. Siendo así, nos centraremos en los mecanismos de adaptación de componentes, o de reconfiguración:

- **Binding Modification**: Este mecanismo hace referencia a la alteración de los vínculos entre los diferentes componentes de la arquitectura. Estos tienen el objetivo de modificar la interacción entre componentes, lo que es especialmente común en implementaciones con *proxies*. Este tipo de mecanismo de adaptación fue usado por Kabashkin (2017) para añadir fiabilidad a la red de comunicación aérea.
- **Component Replacement, Addition and Subtraction**: En términos simples, este mecanismo se encarga de alterar los componentes que componen la arquitectura; de esta manera, modificando su comportamiento. Ejemplos de esto puede verse en el trabajo de Huynh (2019) en el cual se evalúan varios acercamientos a la reconfiguración de arquitecturas a partir del remplazo de componentes a nivel individual al igual que grupal.

  Este acercamiento a la mutación de la arquitectura también puede verse en el despliegue de componentes como respuesta a cambios en los objetivos de negocio de las aplicaciones al igual que como respuesta a cambios inesperados dentro de la aplicación. Esto puede verse en trabajos como el de Patouni (2006) donde se realizan este tipo de implementaciones.
- **Interface Modification**:

4.2 SISTEMAS EMBEBIDOS

Los sistemas de cómputo embebidos, o simplemente sistemas embebidos, hacen referencia a un sistema compuesto de microcontroladores los cuales están orientados a llevar a cabo una función o un rango de funciones específicas (Heath, 2002). Este tipo

de sistemas, debido a la posibilidad de combinar hardware y software en una manera compacta, se ha visto en multiples campos de la industria como lo son el sector automotor, de maquinaria industrial o electrónica de consumo (Deichmann y cols., 2022).

INTERNET OF THINGS

SMART CAMPUS

4.3 NOTACIÓN

El concepto de *notación* está definido como la representación gráfica del habla (Crystal, 2011). En el contexto de las ciencias de la computación, esta idea se ha extrapolado con el fin de representar diferentes conceptos específicos del software y algoritmia de manera visual (Rutanen, 2018). Esto puede verse con la existencia de lenguajes de notación como lo es UML con el cual se realizan representaciones que van desde arquitecturas de software, estructuras de base de datos, entre otros (Booch, Rumbaugh, y Jacobson, 2005).

GRAMÁTICA

La gramática, más específicamente gramáticas libres de contexto, son un conjunto de reglas descriptivas. Este conjunto de reglas, en conjunto de una notación, cumplen la función de dictar si una frase es válida para un lenguaje dado (Sipser, 2012, p. 101).

SERIALIZACIÓN DE DATOS

La serialización de datos se refiere a la traducción de una estructura de datos hacia una manera en la que pueda ser almacenada. En el contexto del proyecto, esta serialización nos permitirá describir las arquitecturas objetivo a partir de la notación y gramáticas establecidas. Así mismo, debería darnos la flexibilidad de describir cualquier tipo de meta para el sistema de software.

4.4 ALGORITMIA DE COMPARACIÓN DE GRAFOS

5 METODOLOGÍA

Para el desarrollo del trabajo de grado, se propone un modelo de prototipado iterativo compuesto de 5 fases (Ver fig. 2). De esta manera, se avanzará a medida que se va completando la fase anterior y permitirá a futuro el poder iterar sobre lo que se ha desarrollado anteriormente.

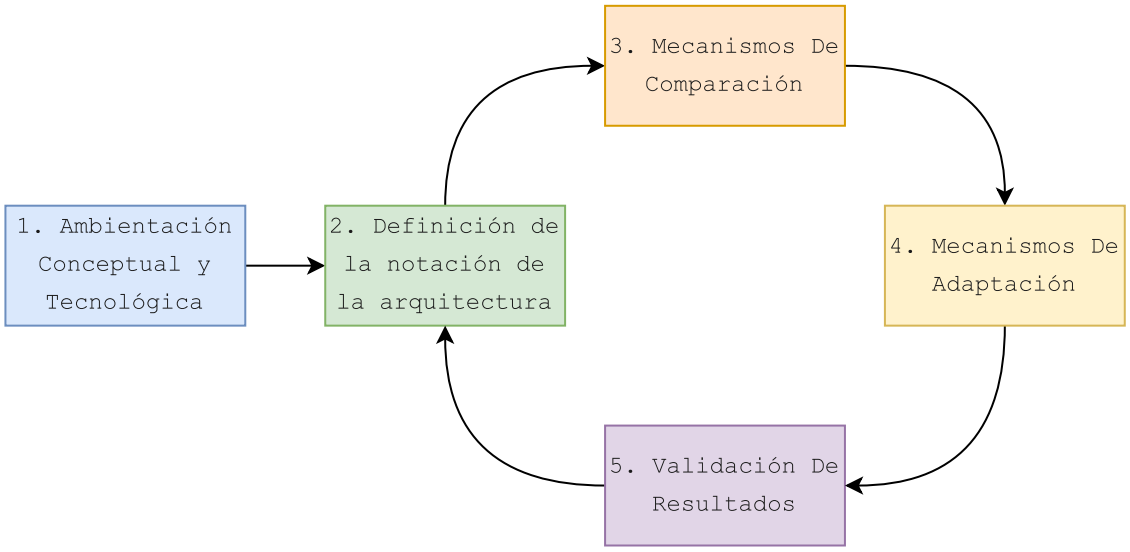

Figura 2: Metodología del proyecto

5.1 AMBIENTACIÓN CONCEPTUAL Y TECNOLÓGICA

La primera fase de la metodología se basa en la investigación de la literatura, al igual que de la industria, necesaria para cubrir las bases tanto conceptuales como técnicas necesarias para el desarrollo del proyecto.

ACTIVIDADES

5.1.1. Identificación de las características principales de un sistema auto-adaptable.

5.1.2. Análisis de los mecanismos de adaptación de la arquitectura.

5.1.3. Análisis los algoritmos empleados para la comparación de la comparación de las arquitecturas.

5.1.4. Determinación de los criterios de selección para el lenguaje de notación.

5.1.5. Evaluación de los posibles lenguajes de programación para la implementación a realizar.

5.1.6. Imprevistos.

5.1.7. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5.2 DEFINICIÓN DE LA NOTACIÓN DE LA ARQUITECTURA

La segunda fase está en la definición del cómo se realiza la declaración de la arquitectura. Partiendo de los criterios de selección establecidos en la fase 1, se espera determinar un lenguaje de notación el cual nos permita definir la arquitectura objetivo a alcanzar, al igual que la gramática correspondiente para poder realizar dicha declaración.

ACTIVIDADES

5.2.1. Selección del lenguaje de marcado a usar a partir de los criterios establecidos.

5.2.2. Definición la gramática a usar para la definición de la arquitectura.

5.2.3. Implementación la traducción de la notación al modelo de grafos.

5.2.4. Determinación como se realizará la representación de los componentes y partes de la arquitectura.

5.2.5. Imprevistos.

5.2.6. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5.3 MECANISMOS DE COMPARACIÓN

Durante la tercera fase del proyecto, se buscará poder determinar e implementar cómo se realizará la comparación entre el estado de la arquitectura obtenido durante la auto-descripción de la misma y el objetivo establecido. Así mismo, y con el fin de reportar a los administradores de los sistemas, también será necesario definir *niveles* de similitud entre las 2 arquitecturas.

ACTIVIDADES

5.3.1. Selección del mecanismo de comparación a usar para evaluación de estado de la arquitectura.

5.3.2. Implementación del mecanismo de comparación seleccionado.

5.3.3. Determinación de los diferentes niveles de similitud entre arquitecturas.

5.3.4. Imprevistos.

5.3.5. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5.4 MECANISMOS DE ADAPTACIÓN

La cuarta fase del proyecto está orientada a la selección, al igual que la implementación en Smart Campus UIS, del conjunto de mecanismos de adaptación de la arquitectura.

5.4.1 ACTIVIDADES

5.4.1. Definición el conjunto de mecanismos de adaptación.

5.4.2. Implementación el conjunto de mecanismos de adaptación seleccionados.

5.4.3. Imprevistos.

5.4.4. Análisis, retroalimentación y conclusiones del desarrollo de la fase.

5.5 VALIDACIÓN DE RESULTADOS

La fase final del proyecto se encargará principalmente de la realización de pruebas de los mecanismos implementados, los resultados obtenidos al igual que la documentación de todo lo que se desarrolló durante el proyecto.

ACTIVIDADES

5.5.1. Realización de las pruebas del funcionamiento de la implementación realizada con diversas arquitecturas objetivo.

5.5.2. Recopilación la documentación generada durante el desarrollo de cada una de las fases del proyecto.

5.5.3. Compilación de la documentación para generar el documento de final.

5.5.4. Correcciones y adiciones para la presentación final del proyecto de grado.

6 CRONOGRAMA

Se debe realizar un cronograma que relacione las actividades prioritarias del proyecto y el tiempo que destinará a cada una de ellas. Tenga en cuenta que el semestre tiene 16 semanas y debe desarrollar todo el trabajo de grado en este tiempo.

7 PRESUPUESTO

Descripción	Responsable	Valor	Cantidad	Precio
DIRECTOR DE PROYECTO PhD. Gabriel Rodrigo Pedraza Ferreira	UIS	COP 305.000/Hora	4 horas mensuales por 4 meses	COP 4’880.000

8 BIBLIOGRAFÍA

Arcaini, P., Riccobene, E., y Scandurra, P. (2015). Modeling and analyzing mape-k feedback loops for self-adaptation. En *2015 ieee/acm 10th international symposium on software engineering for adaptive and self-managing systems* (p. 13-23). doi: 10.1109/SEAMS.2015.10

Booch, G., Rumbaugh, J., y Jacobson, I. (2005). *The unified modeling language user guide* (2.ª ed.). Boston, MA: Addison-Wesley Educational.

Crystal, D. (2011). *A dictionary of linguistics and phonetics*. John Wiley & Sons.

Deichmann, J., Doll, G., Klein, B., Mühlreiter, B., y Stein, J. P. (2022, Mar). *Cracking the complexity code in embedded systems development.* McKinsey's Advanced Electronics Practice.

Forrester Research. (2019, Jun). *Mainframe in the age of cloud, ai, and blockchain.* Forrester Consulting. Descargado de `https://www.ensono.com/resources/white-papers/old-workhorse-new-tech-mainframe-age-cloud-ai-and-blockchain-commissioned-study-conducted/`

Gorla, A., Pezzè, M., Wuttke, J., Mariani, L., y Pastore, F. (2010, 01). Achieving cost-effective software reliability through self-healing. *Computing and Informatics*, *29*, 93-115.

Heath, S. (2002). *Embedded systems design*. Elsevier.

Horn, P. (2001, Oct). *autonomic computing: Ibm's perspective on the state of information technology.* IBM. Descargado de `https://homeostasis.scs.carleton.ca/~soma/biosec/readings/autonomic_computing.pdf`

Huynh, N.-T. (2019). An analysis view of component-based software architecture reconfiguration. En *2019 ieee-rivf international conference on computing and communication technologies (rivf)* (p. 1-6). doi: 10.1109/RIVF.2019.8713678

Iancu, B., y Gatea, A. (2022). Towards a self-describing gateway-based iot solution. En *2022 ieee international conference on automation, quality and testing, robotics (aqtr)* (p. 1-5). doi: 10.1109/AQTR55203.2022.9801938

Jiménez, H., Cárcamo, E., y Pedraza, G. (2020). Extensible software platform for smart campus based on microservices. *RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao*, *2020*(E38), 270-282. Descargado de `www.scopus.com`

Kabashkin, I. (2017). Dynamic reconfiguration of architecture in the communication network of air traffic management system. En *2017 ieee international conference on computer and information technology (cit)* (p. 345-350). doi: 10.1109/CIT.2017.13

Kephart, J. (2011, 01). Autonomic computing: the first decade. En (p. 1-2). doi: 10.1145/1998582.1998584

Krikava, F. (2013, 11). *Domain-specific modeling language for self-adaptive software system architectures.*

Lalanda, P., Diaconescu, A., y McCann, J. A. (2014). *Autonomic computing: Principles, design and implementation*. Springer.

Min-Allah, N., y Alrashed, S. (2020, agosto). Smart campus—a sketch. *Sustainable Cities and Society*, *59*, 102231. Descargado de

https://doi.org/10.1016/j.scs.2020.102231 doi: 10.1016/j.scs.2020.102231
Ouareth, S., Boulehouache, S., y Mazouzi, S. (2018). A component-based mape-k control loop model for self-adaptation. En *2018 3rd international conference on pattern analysis and intelligent systems (pais)* (p. 1-7). doi: 10.1109/PAIS.2018.8598529
Patouni, E., y Alonistioti, N. (2006). A framework for the deployment of self-managing and self-configuring components in autonomic environments. En *2006 international symposium on a world of wireless, mobile and multimedia networks(wowmom'06)* (p. 5 pp.-484). doi: 10.1109/WOWMOM.2006.11
Rutanen, K. (2018). Minimal characterization of o-notation in algorithm analysis. *Theoretical computer science*, *713*, 31–41.
Salehie, M., y Tahvildari, L. (2005, 01). Autonomic computing: emerging trends and open problems. *ACM SIGSOFT Software Engineering Notes*, *30*, 1-7.
Sipser, M. (2012). *Introduction to the theory of computation* (3.ª ed.). Belmont, CA: Wadsworth Publishing.
Wang, Z., Sun, C.-a., y Aiello, M. (2021). Lightweight and context-aware modeling of microservice-based internet of things. En *2021 ieee international conference on web services (icws)* (p. 282-292). doi: 10.1109/ICWS53863.2021.00046
Weiss, G., Zeller, M., y Eilers, D. (2011). *Towards automotive embedded systems with self-x properties.* Fraunhofer-Gesellschaft. Descargado de https://publica.fraunhofer.de/handle/publica/224379 doi: 10.24406/PUBLICA-FHG-224379