**Universidad de Pamplona**

**Facultad de Ingenierías y Arquitectura**

**Programa de Ingeniería de Sistemas**

**Tema:**

# MODELO GENERAL DE CONFIANZA EN SISTEMAS AUTÓNOMOS MEDIANTE UNA ONTOLOGÍA

**Autor:**

**Jorge Erick Vasquez Martínez**

**Pamplona, Norte De Santander**

**Junio 2021**

**Universidad de Pamplona**

**Facultad de Ingenierías y Arquitectura**

**Programa de Ingeniería de Sistemas**

**Trabajo de grado presentado para optar al título de Ingeniero de Sistemas.**

**Tema:**

# MODELO GENERAL DE CONFIANZA EN SISTEMAS AUTÓNOMOS MEDIANTE UNA ONTOLOGÍA

**Autor:**

**Jorge Erick Vasquez Martínez**

**Director:**

**Luz Marina Santos Jaimes**

**Ph.D. Ciencias de la computación y matemáticas computacionales**

**Pamplona, Norte de Santander.**

**Junio 2021.**

**Resumen**

Los sistemas autónomos se refieren a sistemas capaces de operar en un entorno del mundo real interactuando entre sí mismos, estos sistemas se han introducido en gran mayoría de los sectores industriales y ahora se empiezan a incorporar en las vidas cotidianas de las personas, todo esto gracias a la revolución tecnológica en aumento. El éxito de la aceptación rápida de los sistemas autónomos en las diferentes actividades del ser humano depende de qué tanto las personas confían en su correcto funcionamiento. Por lo anterior, este proyecto busca sentar las bases iniciales de lo que podría ser una de las formas de medir la confianza de los humanos hacia un sistema autónomo.

Utilizando herramientas de código abierto como Protege, el lenguaje de programación Python y la librería Owlready2 se diseñó y validó un modelo general de confianza mediante una ontología implementada en los vehículos autónomos. Se realizaron una serie de simulaciones para medir la reputación de un vehículo autónomo donde de manera indirecta se cuantifica la confianza del usuario hacía el vehículo.

**Palabras claves:** Confianza, sistema de reputación, ontología, vehículo autónomo, Methontology.

## Abstract

Autonomous systems refer to systems capable of operating in a real-world environment interacting with each other, these systems have been introduced in great part of the industrial sectors, and now they start to incorporate to the daily lives of the people, all this thanks to the increasing of the technological revolution. The success of the rapid acceptation of autonomous systems in the different activities of the human being, it depends on how much the people trust on their proper functioning. Therefore, this project seeks to lay the initial foundations of what it could be one of the ways to measure the trust of humans towards an autonomous system.

Using the open-source tools such as Protege, the Python programming language and the Owlready2 library, a general trust model was designed and validated through an ontology implemented in autonomous vehicles. A series of simulations were carried out to measure the reputation of an autonomous vehicle where, in an indirect way, the user's trust in the vehicle quantified.



**Keywords:** Trust, reputation system, ontology, autonomous vehicle, Methontology.

**Tabla de contenidos**

## Tabla de Figuras

## Lista de tablas

## Lista de Anexos

# 1. Introducción

El presente proyecto desarrolla una ontología para modelar la confianza de las personas en el uso de sistemas autónomos, se emplean la librería Owlready2, el editor de ontologías Protégé, el lenguaje de programación Python y la plataforma Devsnets. Se selecciona los vehículos autónomos como estudio de caso para validar la ontología y se realizan una serie de simulaciones las cuales tienen el objetivo de evaluar la fórmula para la obtención de la reputación del vehículo. En el repositorio (Vasquez, 2021) se encuentra los archivos correspondientes a las simulaciones del proyecto.

Las contribuciones conseguidas durante el transcurso del presente proyecto se dan principalmente en los siguientes ítems:

- Modelo general de confianza para sistemas autónomos mediante una ontología en Protégé.
- El poder registrar, borrar y actualizar los datos presentes en la ontología a través de Python por medio del paquete Owlready2.
- La obtención de la confianza de un vehículo autónomo por medio de un sistema de reputación.
- El aporte de una primera fórmula para la cuantificación de la reputación, donde se consideró la experiencia del usuario y la parte sensorial del sistema.

## 1.1 Planteamiento del problema

Los sistemas autónomos son sistemas que reaccionan en tiempo real ante situaciones inesperadas, estos sistemas actúan sin la intervención humana en condiciones normales. El uso de los sistemas autónomos en la vida cotidiana que proporcionan algún nivel de autonomía hace surgir la pregunta, ¿Qué tanto se puede confiar en los sistemas autónomos? (Sifakis, 2019). La confianza en estos sistemas es de vital importancia debido a que ayudaría en su promoción y mayor uso, trayendo como efecto una mejora en la calidad de vida de las personas. Actualmente se están desarrollando y poniendo en marcha sistemas autónomos en todos los escenarios posibles, en condiciones específicas y controladas, pero con poco o ningún enfoque en la confiabilidad.

Cuando los sistemas autónomos se utilizan en un entorno no controlado, donde hay un nivel alto de interacción con la gente y un número mucho mayor de variables, el potencial resultante de eventos inesperados y / o indeseables es nada despreciable, estos eventos imprevistos podrían tener un impacto negativo en la aceptabilidad, y por lo tanto comprometer el despliegue generalizado de sistemas autónomos. Para que la sociedad utilice y se beneficie de los sistemas autónomos, las personas deben confiar en ellos. Esto significa que los sistemas autónomos deben tener un funcionamiento adecuado, y deben diseñarse y probarse para garantizar que funcionan de una manera consistente y segura, evidenciando la gran ventaja de poder usar sistemas autónomos en ambientes peligrosos en lugar de arriesgar vidas humanas (Stormont, 2008).

## 1.2 Objetivos

### Objetivo General:

- Desarrollar un modelo general de confianza en sistemas autónomos mediante una ontología.

### Objetivos Específicos:

- Determinar qué tipo de relación de confianza se va a modelar en los sistemas autónomos.
- Identificar los tipos de sistemas autónomos que se van a considerar en el modelo general de confianza.
- Implementar la metodología Methontology para obtener la ontología general de confianza para sistemas autónomos.
- Garantizar la asequibilidad a los sistemas autónomos para llevar a cabo la evaluación.
- Validar la ontología de confianza en sistemas autónomos.

## 1.3 Justificación

Hay preocupación porque la sociedad confíe y adopte los sistemas autónomos, y de este modo se asegure que los dispositivos autónomos posean una mayor aceptabilidad en las personas teniendo mayor cavidad en la actualidad. Por lo anterior, se requiere encontrar la forma de recopilar, tratar, y gestionar datos de las experiencias de los sistemas autónomos para conseguir la inferencia del grado de confianza, a través de avances plenamente integrados de las ciencias técnicas y sociales.

Este proyecto plantea implementar un modelo general de confianza mediante una ontología obteniendo las experiencias que ha tenido el usuario usando el sistema autónomo y teniendo en cuenta las características que poseen estos sistemas para su buen funcionamiento, la ontología identifica los conceptos y relaciones sobre la confianza, llevados al plano de los sistemas autónomos donde se involucran los vehículos, aeroplanos, robots, dispositivos de Internet de las cosas inteligentes, entre otros.

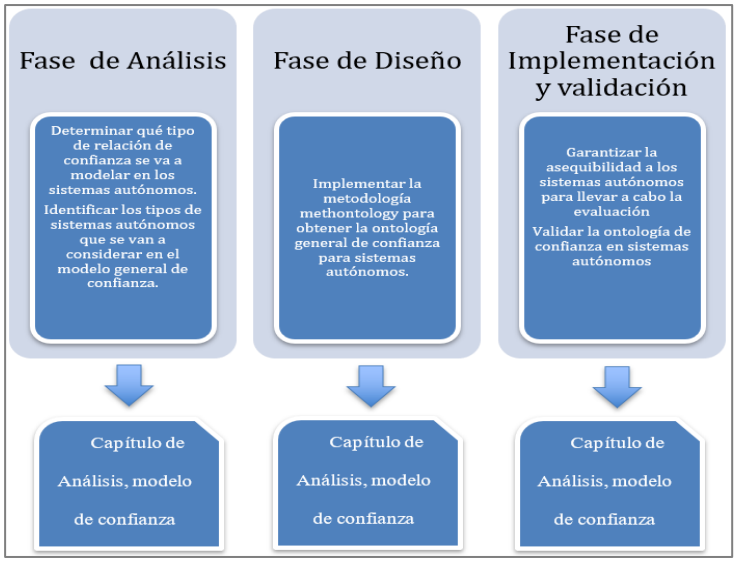
## 1.4 Metodología

La metodología que se seleccionó para el desarrollo del proyecto fue con un enfoque de investigación aplicada, dado que el proyecto se centra en el análisis y resolución de problemas dando el uso más óptimo a los recursos con los que se cuentan para alcanzar a cabalidad el cumplimento de los objetivos de este proyecto.

El proyecto se fragmento en (3) fases: La primera fase se basó en explorar, investigar y dar una apertura teórica al tema, se exploraron antecedentes, los diferentes tipos de editores de ontologías, las diferentes metodologías para la creación de ontologías y las diferentes librerías de programación orientadas a ontologías. En el desarrollo de esta fase se seleccionó los vehículos autonomos como el sistema a implementar y el sistema de reputación como forma de cuantificar la confianza. Seguidamente, en la segunda fase con toda la información recopilada se realizó la selección de la librería, el editor de ontologías más acorde al proyecto y la metodología apropiada para la creación de ontologías para así poder continuar con la tercera y última fase que básicamente consistió en el diseño y desarrollo de una ontología para finalmente poder validar esta ontología con base a los elementos teóricos y herramientas seleccionadas en las fases anteriores.

La recopilación de información para este proyecto es basada en fuentes de información como artículos, investigaciones, libros e internet. En la Figura 1, se representa las fases de desarrollo del proyecto para dar cumplimiento a cabalidad con los objetivos planteados y el capítulo del documento donde se desarrolla cada uno.

Figura 1. Fases de desarrollo




Fuente: Autor del proyecto.

# 2. Marco teórico y estado del arte

## 2.1 Marco Conceptual

### 2.1.1 Ontologías

Las ontologías definen los términos a utilizar para describir y representar un área de conocimiento, y ayudan a especificar un vocabulario relativo a un cierto dominio. Se asume la ontología como una descripción de los conceptos y relaciones en un dominio de aplicación. Este vocabulario define entidades, clases, predicados, funciones y las relaciones entre otros componentes.

- **Funciones:** Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- **Relaciones:** Representan la interacción entre los conceptos del dominio.
- **Reglas:** Describen inferencias lógicas que pueden ser derivables de una aserción en una forma particular.

Un indicador de la complejidad de una ontología es el conjunto de relaciones conceptuales. (Fox, 1998) Sostiene que una ontología se define como un vocabulario más una especificación del significado de dicho vocabulario. Para el desarrollo de ontologías existen unos principios a seguir, algunos de ellos son (Gómez-Pérez, 1999):

- **Claridad y objetividad:** La ontología debe proporcionar el significado de los términos.

- **Completitud:** Los términos se deben definir a partir de condiciones suficientes y necesarias.
- **Principio de distinción ontológica:** Las clases de la ontología son disjuntas.
- **Estandarización de nombres:** Definir y seguir un estándar para nomenclatura de los conceptos

**Tipos de ontologías**

- **Ontologías de dominio:** Las ontologías de dominio representan una parte del mundo, en la que se representa el conocimiento especializado perteneciente a un dominio o un subdominio. Por ejemplo, una ontología acerca del dominio de hardware de computadoras podría modelar los conceptos a tarjeta de red.
- **Ontologías de tareas:** Describe términos involucrados en los procesos de resolución de problemas los cuales pueden estar relacionados con tareas similares en el mismo dominio o en dominios distintos.
- **Ontologías de información:** Especifican la estructura de registro de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.
- **Ontologías generales:** Representan conceptos generales que no son específicos de un dominio, las ontologías generales pueden llegar a reutilizarse a través de diferentes dominios.

De acuerdo a la W3C (World Wide Web Consortium) las ontologías son utilizadas en las siguientes áreas: portales Web, colecciones multimedia, administración de sitios Web corporativos, documentación de diseño, agentes inteligentes, entre otros.

### 2.1.2 OWL (Web Ontology Language)

Es un lenguaje de marcado para publicar y compartir datos usando ontologías en la WWW (World Wide Web). OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF (Resource Description Framework) y codificado en XML (Extensible Markup Language).

### 2.1.3 Selección del editor de ontologías

Los editores son herramientas para la construcción de ontologías que permiten utilizar entornos gráficos para la visualización y creación de ontologías, así como de otros servicios de acuerdo a la herramienta. Para la selección del editor de ontologías, inicialmente se realizó una exploración e investigación de las herramientas más comunes que se adaptaran a la propuesta de la ontología, la cual es que sea fácil de implementar y posea la practicidad adecuada. En la Tabla 1, se presentan las diferentes características que existen entre los distintos editores de ontologías.

Tabla 1. Editores de ontologías

| Características | Apollo | OntoEdit | Protégé | Swoop |
|---|---|---|---|---|
| **Desarrolladores** | KMI Open University | Ontoprise | SMI Stanford University | MND University of Maryland |
| **Disponibilidad** | Código abierto | Licencia de Software | Código abierto | Código abierto |
| **Arquitectura y web semantica** | Standalone | Eclipse Cliente-Servidor | Standalone y Cliente-Servidor | Web basado en Cliente-Servidor |
| **Almacenamiento de ontología** | Archivos | DBMS | Archivos y DBMS (JDBC) | Modelos HTML |
| **Importación de ontología** | Apollo Meta language | XML(S), OWL, RDF(S), Diagrama UML, Esquemas de base de datos (Oracle, MSSQL, DB2, MySQL), Outlook, archivo Sistema Metadata y Remote OntoBroker | XML(S), RDF(S), OWL, (RDF, UML, XML) backend, Excel, BioPortal y DataMaster | OWL, XML, RDF and formatos de texto |
| **Exportación de ontología** | OCML y CLOS | XML(S), OWL, RDF(S), UML y OXML | XML(S), RDF(S), OWL, Clips, SWRL-IQ, Selección de instancias, MetaAnalysis, OWLDoc, Queries y (RDF, UML, XML) backend | RDF (S), OIL y DAML |
| **Modelo de paradigma de conocimiento** | Frames (OKBC) | Frames y First Order logic | Frames, First Order logic, SWRL y Metaclasses | Owl |
| **Motor de inferencia** | No | No | RACER, FACT, FACT++, F-logic y Pellet | Pellet y RDF-like |
| **Grafica de taxonomía** | No | Si | Si | Si |

Fuente: (Alatrish, 2012)

A través de la tabla se pueden identificar las diferentes características que diferencias a algunos editores de ontologías, el editor Protégé es una herramienta de código libre para desarrollar sistemas basados en el conocimiento. Además de ser código libre este editor posee una documentación bastante amplia donde se pueden encontrar infinidad de ejemplos de ontologías hechas en esta herramienta, cuenta con una interfaz amigable con el usuario

haciéndola una herramienta muy intuitiva para el uso, finalmente la variedad de archivos de exportación, motores de inferencia y documentación hizo que esta herramienta fuera la más indicada para el cumplimiento de los objetivos propuestos en la investigación.

### 2.1.4 Metodologías para el desarrollo de ontologías

Por el momento no existe una metodología estándar para el diseño de Ontologías, con el paso del tiempo se va extendiendo el uso de las Ontologías por lo cual desencadena nuevas metodologías las cuales identifican diferentes pasos a seguir para el desarrollo de Ontologías (Bautista-Zambrana, 2015). A continuación, para realizar la comparativa se partió de escritores que ya hubieran realizado estudios a estas metodologías. A través de la Tabla 2 se representa las características más importantes entre las metodologías para el desarrollo de ontologías.

Tabla 2. Características de las metodologías para el desarrollo de ontologías.

| Metodologías | Ontologías basadas en corpus | Público Objetivo | Nivel de detalle | Aplicación de software asociado | Fase de conceptualización |
|---|---|---|---|---|---|
| **Uschold and King** | No | Desarrolladores de ontologías | 3 | No | No |
| **Methontology** | No | Ingenieros de ontología e investigadores | 5 | WebODE y Protégé | Si |
| **On-To-Knowledge** | No | Desarrolladores de ontologías | 4 | OntoEdit (Ahora llamado OntoStudio) | Si |
| **Noy and McGuinness** | No | Desarrolladores de ontologías | 5 | Protégé | No lo especifica |
| **Terminae** | Si | Ingenieros del conocimiento y terminólogos | 4 | Terminae | Si |
| **Termontography** | Si | Constructores de ontologías, terminografos y lexicógrafos | 3 | Termontography Workbench | No lo especifica |

Fuente: (Bautista-Zambrana, 2015)

Basado en las características de la Tabla 2 se puede observar que la metodología Methontology cumple con la mayoría de los criterios, exceptuando las ontologías basadas en corpus. Methontology posee un nivel de detalle alto, además contiene una gran cantidad de repositorios donde se encuentren ejemplos del uso de esta metodología.

### 2.1.5 Código abierto

El software open source es un código diseñado de manera que sea accesible al público: todos pueden ver, modificar y distribuir el código de la forma que consideren conveniente. El open source se convirtió en un movimiento y una forma de trabajo que trasciende la producción del software. Este movimiento utiliza los valores y el modelo de producción descentralizada del software open source para hallar nuevas maneras de solucionar problemas en las comunidades y los sectores (Hat, 2020).

### 2.1.6 Arquitectura de software

En palabras simples la arquitectura de software son patrones o lineamientos que ayudan a la construcción de un programa (aplicación) (Jucaripo, 2019). Estos patrones permiten tener una guía para los desarrolladores, analistas y todos los cargos relacionados para lograr cumplir con los requerimientos de la aplicación. A continuación, se describen las etapas para el desarrollo de la arquitectura de software.

- **Requerimientos**: En esta etapa se recolecta la información y se documentan los requerimientos que influyen en la arquitectura de la aplicación.

- **Análisis:** En esta etapa se profundiza en los procesos de negocio que están involucrados en el sistema.
- **Diseño**: Es la etapa más crucial, aquí se define el uso de tecnologías adecuadas para resolver el problema y no solo porque una tecnología está de moda.
- **Documentación**: Una vez se ha definido el diseño es necesario comunicarlo de manera eficiente y eficaz a todos los involucrados, es importante crear documentación que sirva como referencia a todos y sea el marco de trabajo para todos.
- **Desarrollo:** En esta etapa los programadores tienen asignadas tareas específicas y el arquitecto de software estará probando los diferentes módulos para poder hacer mejoras, o correcciones al sistema final.
- **Implementación:** En esta etapa se realiza cuando se monta el sistema en producción y evaluar su comportamiento, verificando qué funciona correctamente.

## 2.2 Estado del arte

### 2.2.1 Internacional

**An ontology of trust: Formal semantics and transitivity.** En la Web, las personas y los agentes de software tienen que interactuar con "extraños". Esto hace que la confianza sea un factor crucial en la Web (Huang, 2006). Básicamente, la confianza se establece en la interacción entre dos entidades y cualquier entidad solo tiene un número finito de relaciones de confianza directas. Sin embargo, las actividades en la Web requieren que las entidades

interactúen con otras entidades desconocidas o desconocidas. Este documento formaliza la semántica de la confianza y estudia la transitividad de la confianza.

**A New Model for Trust and Reputation Management with an Ontology Based Approach for Similarity Between Tasks.** Este trabajo muestra la integración del modelo de confianza, reputación y WSMO (Web Service Modeling Ontology) de dos maneras (Caballero, 2006):

- Como los agentes usan WSMO como ontología para definir sus requisitos, respuestas, características y métricas dependientes del dominio;
- Como se puede mejorar el proceso de descubrimiento de servicios web en WSMO utilizando criterios de confianza y reputación proporcionados por el modelo a partir de los datos almacenados por los agentes consumidores en interacciones anteriores.

Este proyecto propone un nuevo modelo para gestionar los valores de confianza y reputación que un agente tiene sobre el resto de agentes asociados a la realización de una determinada tarea.

**Engineering an Ontology for Autonomous Systems.** La ontología modular contiene conceptos tanto genéricos como específicos de dominio para la descripción e ingeniería de sistemas autónomos. La ontología sirve como base en una metodología para obtener los modelos conceptuales del sistema autónomo. El objetivo es obtener y utilizar estos modelos como entrada (Bermejo-Alonso, 2011). Este artículo describe el desarrollo de una ontología

para sistemas autónomos, como la etapa inicial de un programa de investigación sobre ingeniería de sistemas autónomos dentro de un enfoque de control basado en modelos.

**Applied ontologies and standards for service robots.** El auge de los robots domésticos y de asistencia personal pronostica una sociedad colaborativa entre humanos y robots (Haidegger, 2013). Una de las principales tareas de la comunidad de la robótica es agilizar las tendencias de desarrollo, trabajar en la armonización de taxonomías y ontologías, junto con la estandarización de términos, interfaces y tecnologías. Es importante mantener sincronizados el progreso científico y la comprensión del público, mediante una difusión y una educación eficientes. Este artículo describe la necesidad de desarrollar ontologías y estándares robóticos centrándose en los esfuerzos de investigación pasados y actuales.

**An evaluation of reputation with regard to the opportunistic forwarding of messages in VANETs.** El artículo plantea una ontología para la evaluación de reputación en el intercambio de mensajes en redes Ad Hoc Vehiculares (Moreira, 2019). En la propuesta basada en un sistema de reputación centralizado, ser un vehículo con buen comportamiento es el resultado de no enviar mensajes falsos y participar cooperativamente en el encaminamiento de mensajes a otros vehículos a su alrededor, los vehículos evalúan a sus vecinos y se envía la puntuación al sistema. A diferencia del presente proyecto se plantea la ontología como parte del sistema del vehículo y con diversos factores que califican la experiencia de utilizar un vehículo autónomo.

**Trustonomy.** El presente proyecto ha recibido financiación del programa de investigación e innovación Horizonte 2020 de la Unión Europea, este proyecto consiste en

construir un marco de trabajo el cual busca la aceptación y la confianza en los vehículos autonomos. (Trustonomy, 2020).

**"Trust but Verify": The Difficulty of Trusting Autonomous Weapons Systems.** Los sistemas de armas autónomas (AWSs) plantean muchos desafíos en entornos complejos del campo de batalla. Los debates realizados se han centrado en gran medida en cuestiones tecnológicas o políticas. Los procesos actuales de adquisición, capacitación e implementación impiden el desarrollo de dicha confianza, por lo que actualmente no hay rutas para que un combatiente desarrolle confianza en un AWS (Danks & Roff, 2018). El trabajo estudia tres posibles cambios en las practicas actuales con el fin de facilitar el tipo de confianza profunda que se requiere para el uso apropiado y ético de la AWS.

### 2.2.2 Nacional

**Un acercamiento a la construcción de ontologías con la herramienta libre Protege.** Esta ontología denominada Sawa, permite soportar la búsqueda inteligente de información sobre trabajos de grado. La metodología contempló cuatro fases (Cabrera, 2014).

- Se apropió el conocimiento sobre web semántica, Ontologías y la herramienta Protege.
- Se diseñó y se definieron los componentes de la ontología Sawa.
- Logro implementarse la ontología Sawa con la herramienta libre Protege.
- Se realizaron pruebas y se evaluaron los resultados.

En este artículo se presenta los resultados obtenidos en el proyecto de investigación que tiene como objetivo la construcción de una ontología sobre los trabajos de grado de los estudiantes de pregrado de la Universidad de Nariño (Colombia), utilizando la herramienta de software libre Protege.

**Ontología para el proceso evaluativo en la educación superior.** La evaluación en los entornos virtuales comparte algunas estrategias realizadas en la modalidad presencial, pero se puede asignar el mismo valor. Aunque el propósito es similar, el proceso tiene variantes importantes. Para lograr la sistematización se requiere de técnicas e instrumentos que permitan obtener información antes, durante y después del proceso de enseñanza aprendizaje en la conducta y habilidades del estudiante (Tabares García, 2014).

La ontología propuesta permite la especificación del lenguaje para aspectos de evaluaciones virtuales. En este artículo se muestra el uso de ontologías para apoyar las evaluaciones en los procesos educativos.

### 2.2.3 Regional

**Diseño de una ontología para agentes que monitorean mediciones de sensores.** En el artículo se propone el uso de sistemas multi-agente y ontologías para solucionar el problema de monitoreo de procesos heterogéneos cuyos datos son adquiridos mediante sensores (Hernandez Cruz, Parra Ortega, & Portilla Granados, 2017). El resultado de la simulación demuestra que la ontología diseñada facilita las actividades asociadas al monitoreo de diversas variables, y al control de procesos.

# 3. Análisis preliminar

Este capítulo inicia con una breve introducción acerca de los Sistemas Autónomos, luego se especifica las cualidades que poseen los vehículos autónomos, sus niveles de accidentalidad, sus niveles de autonomía, sensores y ventajas y desventajas que estos poseen. Los vehículos autonomos es el tipo de sistema autónomo que se va a considerar para el modelo general de confianza, con esto se lleva a cabo el cumplimiento del objetivo propuesto de identificar el tipo de sistema autónomo que se va a modelar. Además, se da un concepto sobre la confianza centrándose en la reputación, donde la reputación es el método seleccionado por el cual se cuantifico la confianza, dando cumplimiento al objetivo de determinar el tipo de relación de confianza que se va a modelar en los sistemas autonomos.

## 3.1 Sistema Autónomo (AS)

La utilización de las técnicas adecuadas permite el diseño de sistemas que soportan la interacción e integración de máquinas de aprendizaje con sistemas de computadoras, esto da como resultado el desarrollo de sistemas los cuales tienen un alto grado de autonomía donde estos pueden realizar diferentes tareas.

Los Sistemas Autónomos deben sortear distintos escenarios, debido a esto, requieren de mecanismos que les permitan aprender y usar ese conocimiento para inferir decisiones. Algunos de los mecanismos que se pueden usar para brindarle esa capacidad adaptativa a los sistemas autónomos, están vinculados a las diferentes áreas de la Inteligencia Artificial; como, por ejemplo, las técnicas de aprendizaje de máquina, la teoría de agentes, y en general, las estrategias de computación inteligente. Existen tres eventos destacados en el manejo de

la autonomía de los sistemas, el tipo de tarea que realiza la máquina; la relación entre el humano y una máquina en la realización de la tarea; y la sofisticación de la decisión de la máquina al realizar la tarea. A continuación (Chang, 2018) proporciona una serie de componentes que posee el sistema de un controlador cognitivo para robots autónomos los cuales están compuestos por cuatro agentes.

- Modelo del robot que describe su comportamiento.
- Dispositivo planificador que decide, ordena, selecciona o crea una acción determinada de acuerdo a la información sensorial proveniente del entorno y de los modificadores de comportamiento que genera el modelo del robot.
- Función de coste cuyo objetivo es el ajuste tanto del modelo del robot como del planificador.
- Agenda con las acciones programadas a realizar.

## 3.2 Vehículo Autónomo

Los vehículos autónomos ya son comunes verlos en las carreteras de países europeos, estos vehículos poseen diferentes características que hacen el viaje aún más seguro que en un vehículo tradicional sin esta tecnología. Se espera un gran impacto en la sociedad aun así teniendo la circulación de la mayoría de vehículos autónomos entre los niveles 2 y 3 (Ver Sección 3.2.2), para el año 2030 se estipula que 1 de cada 10 vehículos sea autónomo (Rg, 2020), esto quiere decir que para el año 2030 los autos que posean un nivel 5 de autonomía circulen por todas las carreteras europeas.

### 3.2.1 Índices de accidentalidad

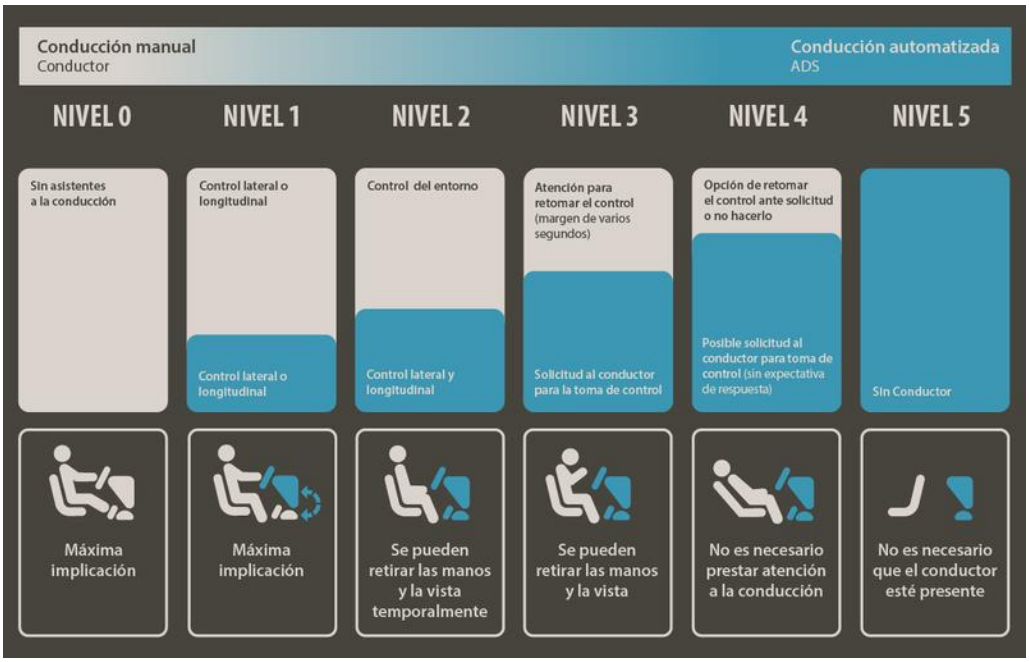
Una de las grandes ventajas que poseen los vehículos autónomos es la seguridad que estos ofrecen, la asistencia automatizada que estos vehículos poseen hace tener una percepción de seguridad a sus ocupantes, esto hace que exista una esperanza sobre la capacidad que puede tener esta tecnología de evitar colisiones.

En el cuarto trimestre de 2020 en las vías de Estados Unidos, se registró un accidente por cada 3,45 millones de millas conducidas en las que los conductores tenían el piloto automático activado. Para aquellos que conducen sin piloto automático, pero con las características de seguridad activas, se registró un accidente por cada 2.05 millones de millas conducidas. Para aquellos que conducen sin piloto automático y sin las características de seguridad activas, se registró un accidente por cada 1,27 millones de millas conducidas. En comparación, los datos más recientes de la NHTSA muestran que en los Estados Unidos hay un accidente automovilístico cada 484,000 millas (Tesla, 2020).

### 3.2.2 Niveles de autonomía

La conducción autónoma ha sido clasificada en seis niveles por la Sociedad de Ingenieros de Automoción (SAE, por sus siglas en inglés). La actualización más reciente es el estándar J3016. Los tres primeros niveles de la SAE para los vehículos autónomos (0 al 2) incluyen características de asistencia al conductor. Mientras tanto, los tres últimos (3 al 5) incluyen características reales de automatización (Terol, 2021). En la Figura 2 se aprecia los diferentes niveles de autonomía de los vehículos autónomos.

Figura 2. Niveles de conducción




Fuente: (Km77, 2020).

- **Nivel 0: Sin automatización en el vehículo:** En este nivel los vehículos son los más comunes que existen actualmente, se caracterizan porque el conductor realiza todas las tareas, de igual forma los coches no poseen ningún tipo de control autónomo.
- **Nivel 1: Asistencia en la conducción:** Los modelos disponibles en este nivel suelen incluir el sistema automático de frenado de emergencia, control de asistencia de colisión, sistema de control de carril, entre otros.
- **Nivel 2: Automatización parcial:** En este nivel aún se necesaria la presencia de un conductor, el vehículo tendrá la capacidad de realizar algunas tareas que antes eran realizadas por el conductor.

- **Nivel 3: Automatización condicionada:** Aunque la autonomía es elevada en este nivel aún se precisa de un conductor, los vehículos en este nivel comienzan a utilizar sensores LIDAR para registrar lo que ocurre alrededor. Estos sensores combinan visión computarizada, cámaras, radar y localización.
- **Nivel 4: Automatización elevada:** En este nivel, el sistema de conducción automatizada puede guiar el vehículo de forma sostenida en el tiempo sin la expectativa de que el conductor responda ante una demanda de intervención, salvo cuando se encuentre fuera de su ámbito de funcionamiento (Km77, 2020).
- **Nivel 5: Automatización completa:** Aprovechan de compartir y utilizar los datos generados por las telecomunicaciones de las ciudades inteligentes y el IOT. El Audi Aicon es un ejemplo de coche conceptual nivel 5 que solo es posible con tecnología 5G (Terol, 2021).

El nivel 2 es el que más comúnmente se encuentra en los concesionarios de diferentes marcas, mientras que el nivel 3 de momento solo se encuentra en algunas de ellas (Álvarez, 2016).

### 3.2.3 Ventajas y desventajas

La tecnología de los vehículos autónomos, así como cualquier otra tecnología proporciona unas ventajas y desventajas del uso de esta tecnología, cada día hay más probabilidades de encontrarse con un vehículo autónomo por las carreteras debido a la gran aceptación que está teniendo este tipo de tecnología, a continuación, se nombran algunas ventajas y desventajas que poseen estos vehículos.

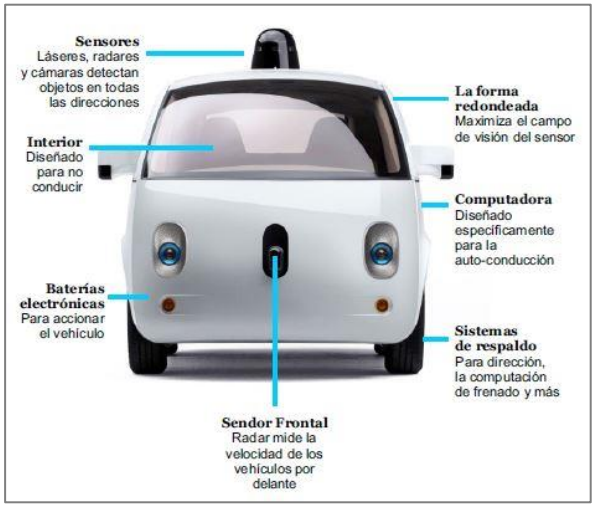
Tabla 3. Ventajas y desventajas de los vehículos autónomos

| Ventajas | Desventajas |
|---|---|
| **Menos atascos,** debido a que estos vehículos poseen tecnología de punta conlleva a que su conducción sea la más óptima y eficiente por lo cual ayudara a reducir los atascos | **El precio**, la salida de estos vehículos al mercado suele tener unos costos elevados. |
| **Reducirán los accidentes,** actualmente existe una gran cantidad de accidentes que tienen como causa los errores humanos por lo cual con esta tecnología estos errores humanos desaparecerán | **Desempleo,** el uso de esta tecnología en empresas de vehículos públicos traerán mayor desempleo ya que no necesitarán choferes. |
| **Mayor productividad,** debido a que mientras el vehículo se anda conduciendo autónomamente el individuo podría estar realizando otra tarea. | **Problemas en el código, los** vehículos autónomos están desarrollados por los humanos por lo que existe la posibilidad que puedan existir problemas en el código por lo que se podrían generar problemas. |

Fuente: Autor del proyecto

### 3.2.4 Sensores y sistemas del vehículo autónomo

Figura 3. Sensores de vehículos autónomos




Fuente: (Garciglia, 2020).

- **Iluminaciones adaptables:** Consiste en una iluminación automática en la carretera donde no existe intervención del conductor. Los faros adaptables automáticos son capaces de: Auto nivelarse, adaptarse en curvas, detectar el nivel lumínico y encenderse o apagarse automáticamente.
- **Asistencia de atasco:** Se muestra una información que va más allá del "horizonte de tráfico" y contiene previsiones en mapas con información de tráfico. Para

recibir la información actualizada sobre la zona en obras, los sistemas del vehículo analizan la ruta.

- **Freno de emergencia autónomo:** Este revolucionario sistema frena de forma autónoma el automóvil cuando detecta que se va a producir una colisión o un atropello.
- **GPS (Global Positioning System):** Es un sistema que permite posicionar cualquier objeto (una persona, un vehículo) sobre la Tierra con una precisión de hasta centímetros.
- **Control crucero:** Su cometido es sencillo, mantiene una velocidad prefijada por el conductor de forma automática, sin necesidad de que el conductor tenga que seguir acelerando. Al hacerse cargo de la velocidad que lleva el vehículo, facilita la labor del conductor liberándole de tener que controlar el velocímetro y de tener que modular con el pedal del acelerador.
- **FSD (Full self-driving):** Es capaz de proporcionar un rendimiento y un control inteligente para alcanzar nuevas cotas de seguridad y conducción autónoma sin afectar a los costos ni a la autonomía.
- **Detección de fatiga:** Lo que hace es interpretar los movimientos del volante y recomendar al conductor que se tome un pequeño descanso si detecta anomalías en su conducción, ya que la analiza desde el momento en que éste inicia la marcha. Este patrón de conducta es analizado junto a otras maniobras y factores, como la duración del trayecto o la hora del día, y se calcula un índice de fatiga.

- **Lidar (Light Detection and Ranging):** Funciona emitiendo pulsos de luz láser millones de veces por segundo, determinando la distancia a los objetos circundantes y construyendo un mapa 3D en directo del entorno basado en la luz reflejada para que el coche pueda tomar decisiones.
- **Reconocimiento de señales de tráfico:** Funciona como un avisador automático sobre límites de velocidad y las señales que el coche se va encontrando por el camino (canalMOTOR, 2019). Las cámaras con las que va equipado el vehículo captan las señales y envían la información a la pantalla interior del vehículo para que el conductor pueda reducir la velocidad. Otros, incluso, permiten que el coche pueda adaptarse de forma automática a la vía por la que circula.
- **Sistema de estacionamiento:** El sistema avisa al conductor de que hay un espacio válido, y le indica hasta dónde tiene que avanzar y dejar el coche (Ibañez, 2012). Y así le seguirá dando instrucciones en la pantalla en el momento oportuno: para insertar la marcha atrás, para acelerar, para frenar, para insertar la marcha hacia adelante, etc. El sistema se encargará de controlar la dirección, y hará girar el volante lo que sea preciso, y cuando sea preciso, sin error, gracias a un electromotor.

## 3.3 Sistema de confianza

La palabra confianza tiene diferentes tipos de significados dependiendo del contexto que se use, ya sea en la psicología, economía e incluso la informática. Algunos estudios más sobre la confianza bajo el enfoque de la “teoría de la información”. La base fundamental de este

enfoque consiste en el supuesto de que la confianza puede concebirse y reducirse a una probabilidad subjetiva, una simple medida numérica basada en la frecuencia y la experiencia, y por tanto pueden realizar modelos para establecerla y calcularla de forma automática por una máquina.

La confianza es un nivel particular de probabilidad subjetiva con la que un agente evalúa que otro agente o grupo de agentes realizará una acción en particular. Cuando se dice que se confía en alguien o que alguien es digno de confianza, se quiere decir implícitamente que la probabilidad de que realice una acción que sea beneficiosa o al menos no perjudicial es lo suficientemente alta.

La confianza es un concepto esquivo sobre el que los investigadores han intentado repetidamente modelar en sus implementaciones de tecnologías, la confianza como tal es un valor subjetivo, algo que no se puede llegar a cuantificar, aunque existen formas de llevar a cabo esta cuantificación y por medio de ella obtener la confianza de un sistema. La reputación es una forma intuitiva de establecer y cuantificar la confianza (Folkerts, 2005). A través de las investigaciones realizadas en el proyecto se tomó la decisión de por medio de la reputación obtener la confianza.

### 3.3.1 Reputación

La reputación se cierne a generar un sentido de confianza entre los usuarios. Al igual que con las tiendas físicas, la confianza y la reputación se pueden construir a través de los comentarios, experiencias y valoraciones de los clientes.

A continuación, se describirá algunas propiedades que son necesarias para que los sistemas de reputación funcionen con eficacia. La reputación entra en escena como fuente alternativa de información para la evaluación de la confianza (Sabater-Mir, 2012).

- Las entidades deben tener una vida útil prolongada y crear expectativas precisas de interacciones futuras.
- Deben capturar y distribuir comentarios, experiencias y valoraciones sobre interacciones anteriores.
- Deben utilizar la retroalimentación para guiar la confianza.

Los sistemas de reputación sirven habitualmente para cinco objetivos (rey, 2014):

- **Construir confianza:** e-Bay, con sus evaluaciones de vendedores y compradores
- **Filtrar:** Amazon, que pondera y señaliza las mejores contribuciones
- **Conectar:** Generar "Matching" a partir de una síntesis de atributos relevantes
- **Estimular comportamientos:** Penalizar acciones negativas, señalizar resultados deseados, recompensar a usuarios y retroalimentar comportamientos
- **Fidelizar:** Producir un efecto de "retención", de elevar los "costes de salida" para los que han acumulado una reputación positiva

### 3.3.2 Fórmula de reputación

En el proyecto se planteó una fórmula para poder obtener la reputación del vehículo autónomo, consistiendo en forma general de tres componentes: proveniente de los sensores del vehículo, de la reputación histórica del vehículo, y de las experiencias del usuario.

***Componente proveniente de los sensores del vehículo***

- **Percepción inicial:** Este valor proviene de la cantidad de sensores y/o sistemas que posee el vehículo autónomo. La presencia de los sensores es fundamental para lograr una conducción que no requiera de la intervención humana.
- **Probabilidad de detección de objetos:** Es el factor de probabilidad con que el vehículo autónomo realiza el reconocimiento de distintos objetos durante un viaje, mediante la interpretación de imágenes que toma la cámara.

***Componente procedente de la reputación histórica del vehículo***

- **CarroLleva:** Este factor es la cantidad de personas que han realizado un viaje en el vehículo autónomo.
- **Histórico de reputación:** Este factor representa el consolidado histórico de las reputaciones recibidas por el vehículo de las experiencias anteriores. Una forma de plantearlo es llevar el acumulado de las reputaciones anteriores y dividirlo en el número de usuarios que han utilizado el sistema.

***Componente proveniente de las experiencias del usuario***

Los dos factores siguientes fueron tomados del Modelo de aceptación de tecnología (TAM) (Martín, 2018), los factores son las experiencias y valoraciones asignadas por el usuario.

- **Disfrute percibido:** Se refiere al grado en el cual una persona encuentra una actividad placentera al utilizar la tecnología.

- **Facilidad percibida de uso:** Es el grado en el cual una persona cree que utilizando un sistema particular se liberará del esfuerzo.

Cada uno de los componentes recibe un peso para calcular la reputación del vehículo autónomo. El porcentaje correspondiente al componente proveniente de los sensores del vehículo es del 20%, este porcentaje se origina debido a que los sensores del vehículo desde su fábrica tienden a ser constantes, estos valores son difíciles que cambien con el tiempo; así como también el algoritmo de detección de objetos, si no hay ningún cambio en el código fuente la probabilidad de detección de objetos tiende a ser la misma. El porcentaje proporcionado al componente procedente de la reputación histórica del vehículo es del 50%, se justifica en la importancia que tienen las acciones anteriores para el cálculo de la reputación del vehículo, siendo en la misma proporción que la última valoración de confianza obtenida. Finalmente, el porcentaje de 30% asignado al componente proveniente a las experiencias del usuario tomadas del Modelo de Aceptación de Tecnología (TAM), se sustenta en que la perspectiva del usuario a la hora de calcular la reputación llega ser importante debido a que este en quien usa la tecnología y se da realmente cuenta de que tan significativo puede llegar a ser la tecnología usada en su vida. A continuación, la fórmula planteada para evaluar la confianza en un vehículo autónomo.

$$Reputación = ((Percepción\ Inicial + Probabilidad\ de\ detección)/2) * 0.20 + (Historico\ de\ reputación/CarroLleva) * 0.50 + ((Facilidad\ percibida\ de\ uso + Disfrute\ percibido)/2) * 0.30$$

En la sección 5.3 se realiza una variación de la fórmula para así poder comparar los resultados expuestos en la sección 5.2 los cuales fueron obtenidos por medio de la formula presente en esta sección.

### 3.4 Owlready2

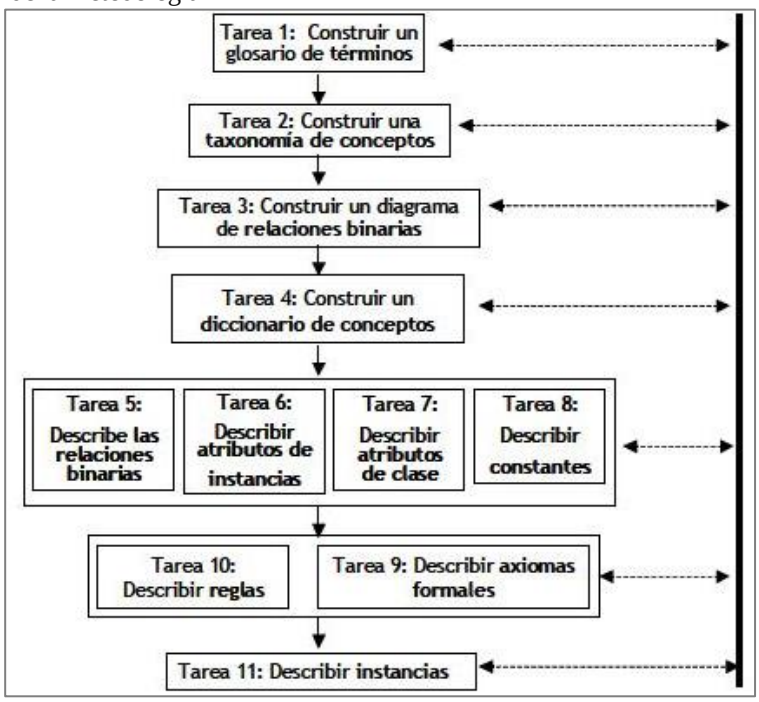
Owlready2 es un paquete para manipular las ontologías OWL 2.0 en Python. Este paquete se seleccionó debido a la facilidad de poder cargar, modificar y guardar ontologías, además es compatible con el razonamiento a través de HermiT (JB., 2017). El paquete Owlready2 posee las siguientes ventajas:

- Importar ontologías OWL 2.0 en formato NTriples, RDF/XML u OWL/XML.
- Exportar ontologias OWL 2.0 a NTriples o RDF/XML.
- Manipula las clases de ontología, las instancias y las propiedades de forma transparente, como si fueran objetos Python normales.
- Añadir métodos Python a las clases de ontología.

## 4. Diseño de la ontología

La construcción y diseño de la ontología se realizó por medio de la metodología Methontology, la cual posee un nivel de detalle muy superior, donde por medio de esta permite satisfacer los requerimientos propuestos en el proyecto, con esto se da solución al objetivo de implementar la metodología Methontology para obtener la ontología general de confianza para sistemas autónomos. En la Figura 4, se visualiza la estructura de la metodología Methontology.

Figura 4. Estructura de la metodología




Fuente: (Corcho, Fernández-López, & Gómez-Pérez, 2005).

A continuación, se describen tareas para el desarrollo de la ontología en el dominio de la confianza en los vehículos autónomos. En la Tarea 8 se debe describir las constantes, pero en la ontología Carro Autónomo no se usó ninguna constante por lo tanto este paso

quedó omitido. En la Tarea 9, se describen expresiones lógicas que son siempre verdaderas, para la ontología Carro Autónomo las restricciones no siempre van a ser verdaderas, debido a esto, este paso queda omitido.

### 4.1 Tarea 1. Construir el glosario de términos.

En esta tarea se construye un glosario de términos que incluye todos los aspectos relevantes del dominio (conceptos, instancias, atributos, relaciones entre conceptos, etcétera). En la Tabla 4, se representa el glosario de términos de la ontología Carro Autónomo.

Tabla 4. Criterios taxonómicos

| Nombre | Sinónimos | Acrónimos | Descripción | Tipo |
|---|---|---|---|---|
| **Carretera** | _____ | _____ | Carretera por la que transita el vehículo | Concepto |
| **Confianza** | _____ | _____ | Es la confianza que se siente por el uso de una tecnología | Concepto |
| **Carro** | _____ | _____ | Carro con el cual se van a realizar las simulaciones de un vehículo autónomo | Concepto |
| **Usuario** | _____ | _____ | Es el individuo que realiza las calificaciones del vehículo autónomo | Concepto |
| **Viaje** | _____ | _____ | Trayecto que recorrerá el usuario en el vehículo | Concepto |
| **Conductor** | _____ | _____ | Individuo que conduce el vehículo en ciertos momentos de la simulación | Relación |
| **Lleva** | _____ | _____ | El vehículo que lleva a la persona | Relación |
| **Inicia** | _____ | _____ | Desde que dirección inicia | Relación |

| | | | | |
|---|---|---|---|---|
| | | | la simulación con el vehículo | |
| **Termina** | _____ | _____ | Desde que dirección termina la simulación con el vehículo | Relación |
| **Adquiere** | _____ | _____ | El carro adquiere una reputación | Relación |
| **Reputacion_Carro** | _____ | _____ | Es la reputación que posee el vehículo | Atributo de instancia |
| **Disfrute_percibido** | _____ | _____ | Se refiere al grado en el cual una persona encuentra una actividad placentera al utilizar la tecnología | Atributo de instancia |
| **Facilidad_percibida_de_uso** | _____ | _____ | Es el grado en el cual una persona cree que utilizando un sistema particular se libera de un esfuerzo | Atributo de instancia |
| **Historico_reputacion** | _____ | _____ | Este atributo viene de la suma de todas las reputaciones asignadas por los usuarios que realizaron la simulación en el vehículo | Atributo de instancia |
| **Marca_carro** | _____ | _____ | Es la marca del vehículo | Atributo de instancia |

Fuente: Autor del proyecto.

## 4.2 Tarea 2. Construir una taxonomía de conceptos.

Se define una jerarquía de conceptos, esta es elaborada a partir del glosario de términos definidos en la Tarea 1. En la Figura 5, se representa la taxonomía de conceptos de la ontología Carro Autónomo descritos en la Tarea 1.

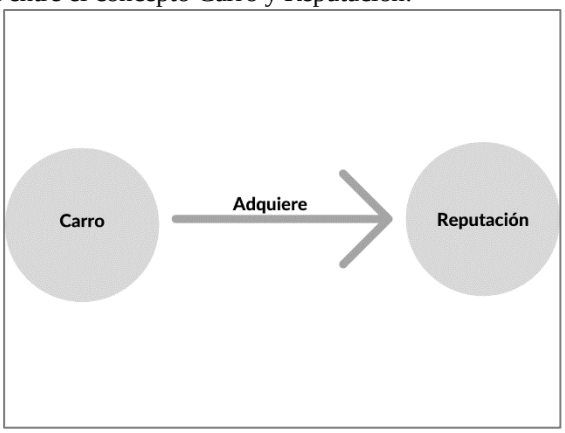
Figura 5. Consolidación de conceptos


Fuente: Autor del proyecto.

## 4.3 Tarea 3. Construir un diagrama de relaciones binarias.

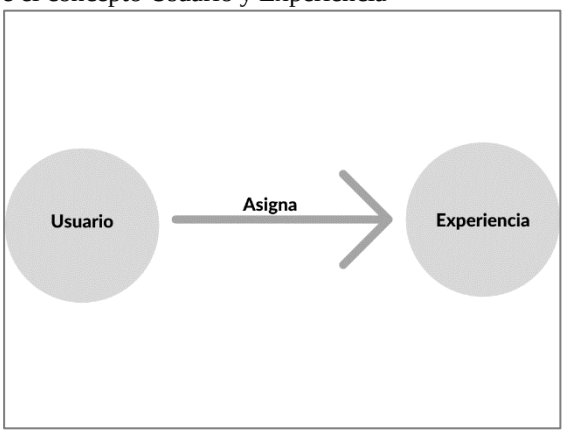
En esta tarea se establecen los tipos de relaciones entre los conceptos (clases) especificados en la taxonomía. De la Figura 6 a la Figura 17 se representan las relaciones binarias existentes en la ontología Carro Autónomo.

Figura 6. Relación binaria entre el concepto Carro y Reputación.



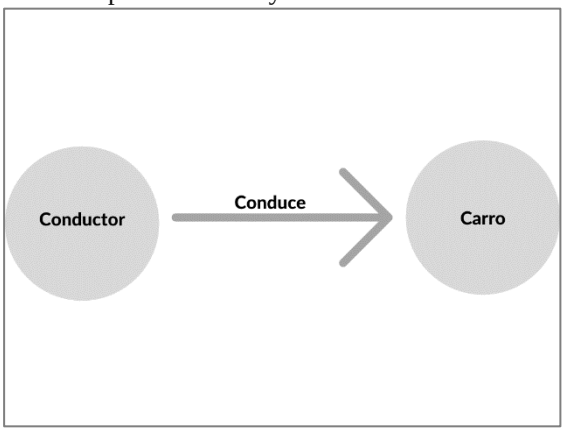

Fuente: Autor del proyecto.

Figura 7. Relación binaria entre el concepto Usuario y Experiencia



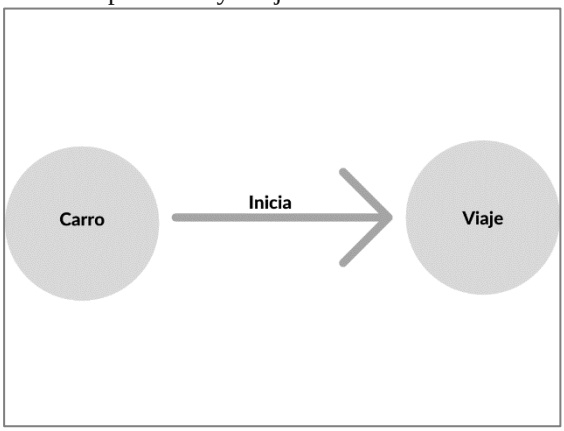

Fuente: Autor del proyecto.

Figura 8. Relación binaria entre el concepto Conductor y Carro



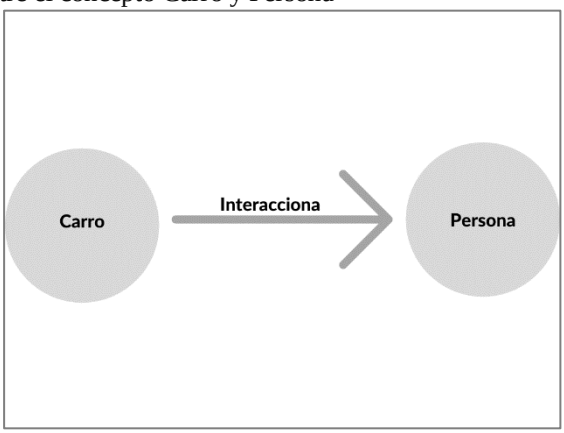

Fuente: Autor del proyecto.

Figura 9. Relación binaria entre el concepto Carro y Viaje



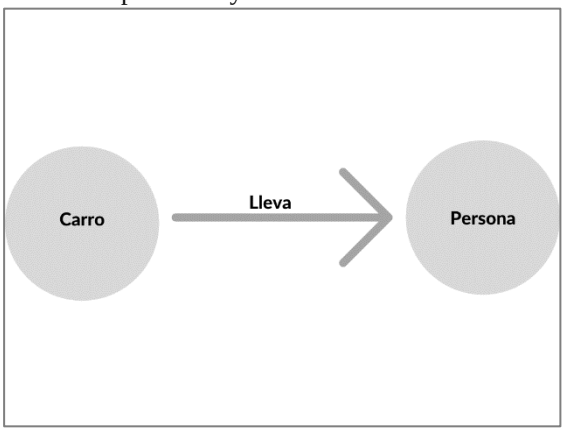

Fuente: Autor del proyecto.

Figura 10. Relación binaria entre el concepto Carro y Persona



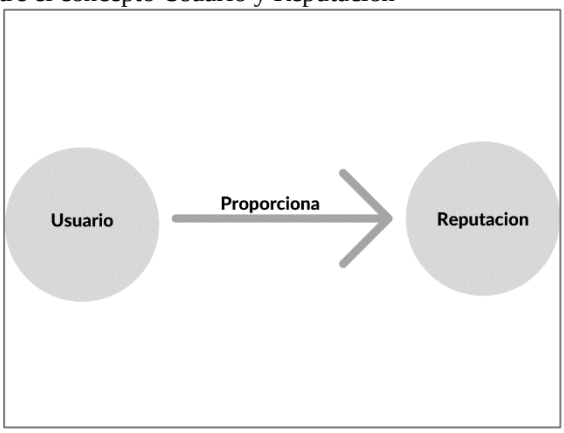

Fuente: Autor del proyecto.

Figura 11. Relación binaria entre el concepto Carro y Persona



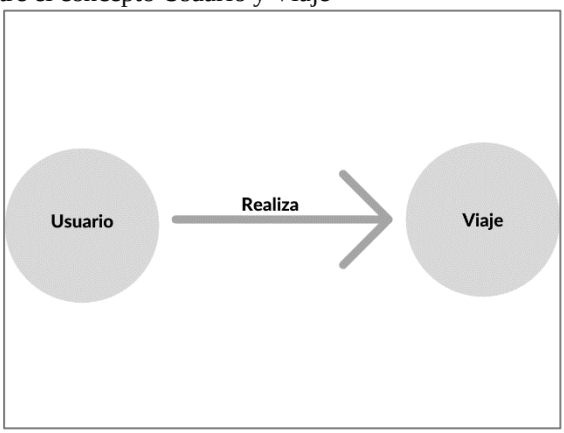

Fuente: Autor del proyecto.

Figura 12. Relación binaria entre el concepto Usuario y Reputación



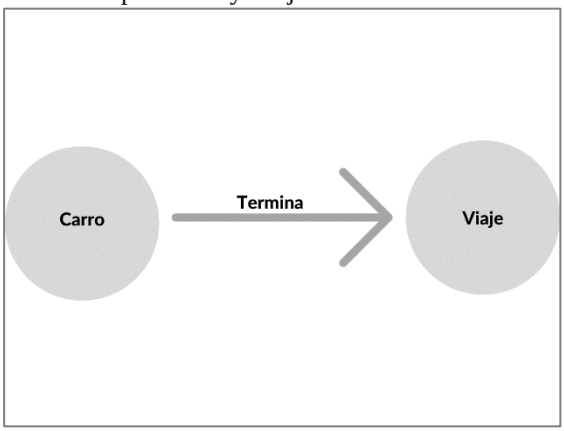

Fuente: Autor del proyecto.

Figura 13. Relación binaria entre el concepto Usuario y Viaje



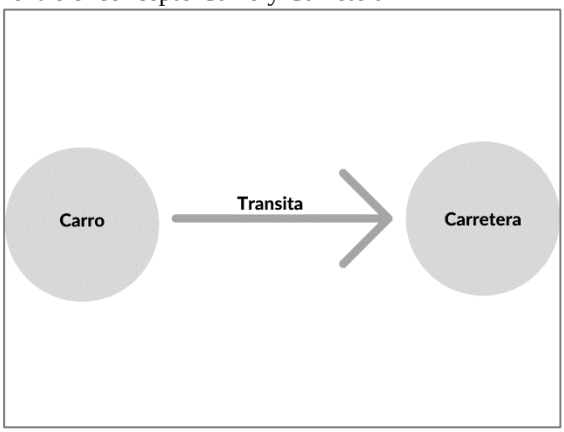

Fuente: Autor del proyecto.

Figura 14. Relación binaria entre el concepto Carro y Viaje



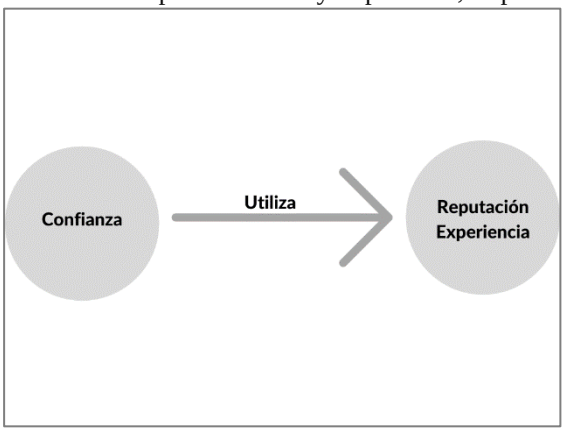

Fuente: Autor del proyecto.

Figura 15. Relación binaria entre el concepto Carro y Carretera



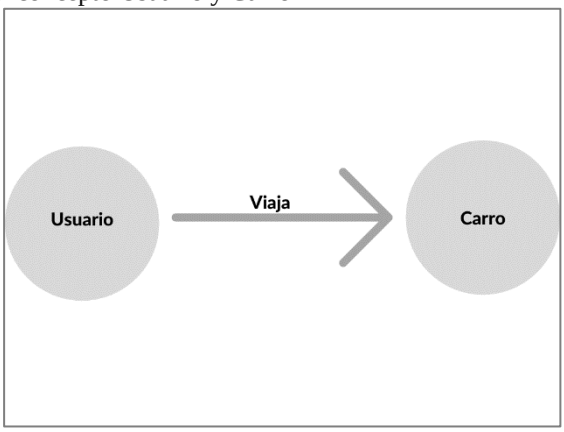

Fuente: Autor del proyecto.

Figura 16. Relación binaria entre el concepto Confianza y Reputación, Experiencia




Fuente: Autor del proyecto.

Figura 17. Relación entre el concepto Usuario y Carro




Fuente: Autor del proyecto.

## 4.4 Tarea 4. Construir el diccionario de conceptos.

El diccionario de conceptos contiene los conceptos más importantes del dominio, sus relaciones, instancias, atributos de clases y atributos de instancias del dominio. En la Tabla 5, se representa la construcción del diccionario de conceptos de la ontología Carro Autónomo.

Tabla 5. Construcción del diccionario de conceptos

| Nombre del concepto | Instancias | Atributos de clases | Atributos de instancias | Relaciones |
|---|---|---|---|---|
| **Carretera** | carrera_27_con_calle_20, carrera_27_con_calle_34, carrera_27_con_calle_56, carrera_27_con_calle_9 | _____ | _____ | _____ |
| **Camara_Carretera** | _____ | _____ | _____ | _____ |
| **Limite_de_velocidad** | _____ | _____ | _____ | _____ |
| **Linea_carretera** | _____ | _____ | _____ | _____ |
| **Linea_Continua** | _____ | _____ | _____ | _____ |
| **Linea_Trazos** | _____ | _____ | _____ | _____ |
| **Sensores_Carretera** | _____ | _____ | _____ | _____ |
| **Señal_de_trafico** | _____ | _____ | _____ | _____ |
| **Semáforo** | _____ | _____ | _____ | _____ |
| **Carro** | RTY456, WE234, ZX1922 | Marca Carro, Asistencia_de_atasco, Control_Crucero, Deteccion_de_fatiga, Freno_de_emergencia_autonomo, FSD, GPS, Iluminaciones_adaptables, LIDAR, Reconocimiento_de_señales_de_trafico, Sistema_de_estacionamiento, Historico_reputacion, Kilometraje, Percepción_inicial | Asistencia_de_atasco, Control_Crucero, Deteccion_de_fatiga, Freno_de_emergencia_autonomo, FSD, GPS, Iluminaciones_adaptables, LIDAR, Reconocimiento_de_señales_de_trafico, Sistema_de_estacionamiento | Adquiere, Inicia, Interacciona, Lleva, Termina, Tiene, Transita |
| **Reporte** | _____ | _____ | _____ | _____ |
| **Sensores** | _____ | Asistencia_de_atasco, Control_Crucero, Deteccion_de_fatiga, Freno_de_emergencia_autonomo, FSD, GPS, | _____ | _____ |

| | | | | |
|---|---|---|---|---|
| | | Iluminaciones_adaptables, LIDAR, Reconocimiento_de_señales_de_trafico, Sistema_de_estacionamiento | | |
| **Confianza** | _____ | Disfrute_percibido, Facilidad_percibida_de_uso, Reputacion_Carro | _____ | Utiliza |
| **Experiencia** | _____ | Disfrute_percibido, Facilidad_percibida_de_uso | _____ | _____ |
| **Reputación** | _____ | _____ | _____ | _____ |
| **Usuario** | Carlos, David, Enrique, Esperanza, Estefanny, Francy, Jorge, Judith, Lucrecio, Nancy, Natalia, Nury | Edad, Genero, Ciudad_Origen, Nivel_de_estudio | Edad, Genero, Ciudad_Origen, Nivel_de_estudio, Disfrute_percibido, Facilidad_percibida_de_uso | Asigna, Realiza, Viaja, Proporciona |
| **Viaje** | _____ | _____ | _____ | _____ |
| **Conductor** | Raul | _____ | _____ | Conduce |

Fuente: Autor del proyecto.

### 4.5 Tarea 5. Definir las relaciones binarias en detalle.

Las relaciones a detallar, mediante una tabla de relaciones binarias, son aquellas especificadas en el diccionario de conceptos. Para cada relación se especifica el nombre, conceptos origen y destino, cardinalidad máxima y relación inversa del dominio. En la Tabla 6, se representa las relaciones binarias en detalle de la ontología Carro Autónomo.

Tabla 6. Relaciones binarias

| Nombre de la relación | Concepto origen | Cardinalidad máxima | Concepto destino | Relación inversa |
|---|---|---|---|---|
| **Adquiere** | Carro | N | Reputación | _____ |
| **Asigna** | Usuario | N | Experiencia | _____ |
| **Inicia** | Carro | N | Viaje | _____ |
| **Interacciona** | Carro | N | Persona | _____ |
| **Lleva** | Carro | N | Persona | _____ |
| **Proporciona** | Usuario | N | Reputación | _____ |
| **Realiza** | Usuario | N | Viaje | _____ |
| **Termina** | Carro | N | Viaje | _____ |
| **Tiene** | Carro | N | Confianza | _____ |
| **Transita** | Carro | N | Carretera | _____ |
| **Utiliza** | Confianza | N | Experiencia, Reputación | _____ |
| **Viaja** | Usuario | N | Carro | _____ |
| **Conduce** | Conductor | N | Carro | _____ |

Fuente: Autor del proyecto.

## 4.6 Tarea 6. Definir los atributos de instancias en detalle.

Se describe en detalle todos los atributos de instancia incluidos en el diccionario de conceptos. Para cada atributo de instancia se especifica el nombre, concepto al que pertenece, tipo de valor, rango de valores (para valores numéricos) y cardinalidad. En la Tabla 7, se visualiza los atributos de las instancias en detalle de la ontología Carro Autónomo.

Tabla 7. Descripción atributo de instancia

| Nombre del atributo de instancia | Concepto al que pertenece | Tipo de valor | Cardinalidad | Rango de valores |
|---|---|---|---|---|
| **Nivel_de_estudio** | Usuario | Texto | 1:1 | _____ |
| **Disfrute_percibido** | Usuario | Entero | _____ | [>=0, <=100] |
| **Facilidad_percibida** | Usuario | Entero | _____ | [>=0, <=100] |
| **Edad** | Usuario | Entero | _____ | [>=18] |
| **Ciudad_Origen** | Usuario | Texto | 1:1 | _____ |
| **Genero** | Usuario | Texto | 1:1 | _____ |
| **Iluminaciones_adaptables** | Carro | Booleano | _____ | _____ |
| **Asistencia_de_atasco** | Carro | Booleano | _____ | _____ |
| **Historico_reputacion** | Carro | Booleano | _____ | _____ |
| **Freno_de_emergencia** | Carro | Booleano | _____ | _____ |
| **GPS** | Carro | Booleano | _____ | _____ |
| **Reputacion_Carro** | Carro | Booleano | _____ | _____ |
| **Control_Crucero** | Carro | Booleano | _____ | _____ |
| **FSD** | Carro | Booleano | _____ | _____ |
| **Deteccion_de_fatiga** | Carro | Booleano | _____ | _____ |
| **LIDAR** | Carro | Booleano | _____ | _____ |
| **Reconocimiento_de_señales_de_trafico** | Carro | Booleano | _____ | _____ |
| **Percepción_inicial** | Carro | Entero | _____ | [>=10, <=100] |
| **Sistema_de_estacionamiento** | Carro | Booleano | _____ | _____ |
| **Marca_Carro** | Carro | Texto | 1:1 | _____ |

Fuente: Autor del proyecto.

## 4.7 Tarea 7. Definir los atributos de clases.

Los atributos de las clases son las características que van a tener cada una de las clases de la ontología. Para cada atributo de clase se describen el nombre, concepto, y tipo de dato. En la Tabla 8, se visualiza los atributos de cada clase de la ontología Carro Autónomo.

Tabla 8. Atributos de clases

| Nombre del atributo de clase | Concepto | Tipo de valor |
|---|---|---|
| **Marca_Carro** | Carro | Texto |
| **Asistencia_de_atasco** | Sensores, Carro | Booleano |
| **Ciudad_Origen** | Usuario | Texto |
| **Control_Crucero** | Carro, Sensores | Booleano |
| **Deteccion_de_fatiga** | Carro, Sensores | Booleano |
| **Disfrute_percibido** | Experiencia, Confianza | Entero |
| **Edad** | Persona | Entero |
| **Facilidad_percibida_de_uso** | Experiencia, Confianza | Entero |
| **Freno_de_emergencia_autonomo** | Carro, Sensores | Booleano |
| **FSD** | Carro, Sensores | Booleano |
| **Genero** | Usuario | Texto |
| **GPS** | Carro, Sensores | Booleano |
| **Historico_reputacion** | Carro | Entero |
| **Iluminaciones_adaptables** | Carro, Sensores | Booleano |
| **Kilometraje** | Carro | Entero |
| **LIDAR** | Carro, Sensores | Booleano |
| **Nivel_de_estudio** | Usuario | Texto |
| **Percepción_inicial** | Carro | Entero |
| **Reconocimiento_de_señales_de_trafico** | Carro, Sensores | Booleano |
| **Reputacion_Carro** | Confianza y Reputación | Entero |
| **Sistema_de_estacionamiento** | Carro, Sensores | Booleano |

Fuente: Autor del proyecto.

## 4.8 Tarea 10. Definir las reglas.

Las reglas necesarias en la ontología Carro Autónomo, se definen en la Tabla 10. Para cada regla se especifica el nombre, descripción, conceptos a los que hace referencia, la expresión, los atributos, las relaciones que intervienen y las variables de las que se hizo uso. Para su especificación se sugiere la forma: condiciones o acciones de algunas reglas que fueron definidas para la ontología Carro Autónomo. En la Tabla 9, se representan las

variables que incidieron en la Tabla 10 que realiza una descripción de las reglas existentes en la ontología.

Tabla 9. Variables intervinientes

| Variable | Descripción |
|---|---|
| **C** | Los individuos que pertenecen al concepto Carro |
| **RC** | Es el valor que poseen los individuos del atributo Reputacion_Carro |
| **P** | Los individuos que pertenecen al concepto Persona |
| **G** | Son los carros que son conducidos por un individuo que pertenecen al concepto Carro |

Fuente: Autor del proyecto.

Tabla 10. Descripción de la regla

| Nombre de la regla | Descripción | Expresión | Conceptos | Atributos | Relaciones | Variables |
|---|---|---|---|---|---|---|
| **Reputacion_Alta** | Arroja como resultado el carro que tenga una reputación mayor a 40 | Carro(?c) ^ Reputacion_Carro(?c, ?rc) ^ swrlb:greaterThan(?rc, 40) -> sqwrl:select(?c, ?rc) | Carro | Reputacion_Carro | ____ | C,RC |
| **Conduce** | Que persona es la que conduce los carros | Persona(?p) ^ Conduce(?p, ?g) -> sqwrl:select(?p, ?g) | Persona | ____ | Conduce | P,G |

Fuente: Autor del proyecto.

## 4.9 Tarea 11. Definir las instancias.

En esta tarea se definen las instancias que han sido identificadas para cada clase en la ontología. Se muestra por medio de una tabla, su nombre, la clase a la cual pertenece y valores de los atributos. En la Tabla 11, se representa de forma detallada la instancia de la ontología Carro Autónomo.

Tabla 11. Descripción detallada de las instancias

| Nombre de la instancia | Nombre del concepto | Atributo | Valores |
|---|---|---|---|
| **Carlos** | Usuario | Facilidad_percibida_de_uso | 40 |
| | | Genero | Masculino |
| | | Edad | 30 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Medellín |
| | | Disfrute_percibido | 50 |
| **David** | Usuario | Facilidad_percibida_de_uso | 60 |
| | | Genero | Masculino |
| | | Edad | 32 |
| | | Nivel_de_estudio | Universidad |
| | | Ciudad_Origen | Bogotá |
| | | Disfrute_percibido | 50 |
| **Enrique** | Usuario | Facilidad_percibida_de_uso | 70 |
| | | Genero | Masculino |
| | | Edad | 70 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Cravo Norte |
| | | Disfrute_percibido | 40 |
| **Esperanza** | Usuario | Facilidad_percibida_de_uso | 90 |
| | | Genero | Femenino |
| | | Edad | 55 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Cravo Norte |
| | | Disfrute_percibido | 80 |
| **Estefanny** | Usuario | Facilidad_percibida_de_uso | 90 |
| | | Genero | Femenino |
| | | Edad | 28 |
| | | Nivel_de_estudio | Magister |
| | | Ciudad_Origen | Medellín |
| | | Disfrute_percibido | 75 |
| **Francy** | Usuario | Facilidad_percibida_de_uso | 40 |
| | | Genero | Femenino |
| | | Edad | 32 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Bogotá |
| | | Disfrute_percibido | 60 |
| **Jorge** | Usuario | Facilidad_percibida_de_uso | 90 |
| | | Genero | Masculino |
| | | Edad | 55 |
| | | Nivel_de_estudio | Magister |
| | | Ciudad_Origen | Pore |
| | | Disfrute_percibido | 25 |

| | | | |
|---|---|---|---|
| **Judith** | Usuario | Facilidad_percibida_de_uso | 40 |
| | | Genero | Femenino |
| | | Edad | 80 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Pore |
| | | Disfrute_percibido | 60 |
| **Lucrecio** | Usuario | Facilidad_percibida_de_uso | 30 |
| | | Genero | Masculino |
| | | Edad | 65 |
| | | Nivel_de_estudio | Profesional |
| | | Ciudad_Origen | Arauca |
| | | Disfrute_percibido | 30 |
| **Nancy** | Usuario | Facilidad_percibida_de_uso | 50 |
| | | Genero | Femenino |
| | | Edad | 50 |
| | | Nivel_de_estudio | Bachiller |
| | | Ciudad_Origen | Cravo Norte |
| | | Disfrute_percibido | 80 |
| **Natalia** | Usuario | Facilidad_percibida_de_uso | 20 |
| | | Genero | Femenino |
| | | Edad | 43 |
| | | Nivel_de_estudio | Doctorado |
| | | Ciudad_Origen | Cravo Norte |
| | | Disfrute_percibido | 90 |
| **Nury** | Usuario | Facilidad_percibida_de_uso | 35 |
| | | Genero | Femenino |
| | | Edad | 36 |
| | | Nivel_de_estudio | Doctorado |
| | | Ciudad_Origen | Medellín |
| | | Disfrute_percibido | 60 |
| **RTY456** | Carro | Iluminaciones_adaptables | Verdadero |
| | | Asistencia_de_atasco | Verdadero |
| | | Historico_reputacion | 40 |
| | | Freno_de_emergencia | Verdadero |
| | | GPS | Verdadero |
| | | Reputaccion_Carro | 40 |
| | | Control_Crucero | Verdadero |
| | | FSD | Verdadero |
| | | Deteccion_de_fatiga | Verdadero |
| | | LIDAR | Verdadero |
| | | Reconocimiento_de_señales_de_trafico | Verdadero |
| | | Percepción_inicial | 90 |
| | | Sistema_de_estacionamiento | Falso |
| **WE234** | Carro | Iluminaciones_adaptables | Falso |
| | | Asistencia_de_atasco | Verdadero |
| | | Historico_reputacion | 45 |
| | | Freno_de_emergencia | Falso |

| | | | |
|---|---|---|---|
| | | GPS | Verdadero |
| | | Reputaccion_Carro | 45 |
| | | Control_Crucero | Verdadero |
| | | FSD | Verdadero |
| | | Deteccion_de_fatiga | Falso |
| | | LIDAR | Verdadero |
| | | Reconocimiento_de_señales_de_trafico | Verdadero |
| | | Percepción_inicial | 60 |
| | | Sistema_de_estacionamiento | Falso |
| **ZX1922** | Carro | Iluminaciones_adaptables | Verdadero |
| | | Asistencia_de_atasco | Falso |
| | | Historico_reputacion | 19 |
| | | Freno_de_emergencia | Falso |
| | | GPS | Verdadero |
| | | Reputaccion_Carro | 19 |
| | | Control_Crucero | Falso |
| | | FSD | Verdadero |
| | | Deteccion_de_fatiga | Falso |
| | | LIDAR | Verdadero |
| | | Reconocimiento_de_señales_de_trafico | Verdadero |
| | | Percepción_inicial | 60 |
| | | Sistema_de_estacionamiento | Verdadero |
| **Carrera_27_con_calle_20** | Carretera, Viaje | ____ | ____ |
| **Carrera_27_con_calle_34** | Carretera, Viaje | ____ | ____ |
| **Carrera_27_con_calle_56** | Carretera, Viaje | ____ | ____ |
| **Carrera_27_con_calle_9** | Carretera, Viaje | ____ | ____ |

Fuente: Autor del proyecto.

Luego de aplicar los pasos de la metodología Methontology se consiguió como resultado la ontología mostrada en Anexo B y que puede ser visualizada en detalle en este enlace desarrollado en la herramienta OWLGrEd.

# 5. Implementación y validación de la ontología

La implementación de la ontología se realizó en el lenguaje de programación Python, debido a la facilidad de uso de la librería Owlready2 como se explicó en la sección 3.4, la cual permite el acceso a las ontologías, inferencias y el poder hacer una mezcla entre los métodos de Python y de la librería; los cambios realizados en las simulaciones se ven reflejados en la herramienta Protégé. Complementario a lo anterior, se utilizaron otras tecnologías que se describen en el presente capítulo para cumplir con los objetivos propuestos. La validación inicial de la ontología se realizó mediante la simulación de viajes en tres diferentes vehículos, cada vehículo cuenta con un video diferente que es utilizado para el reconocimiento de objetos, un conjunto de sensores incluido el FSD (Ver Sección 3.2.2), este sistema permite que el vehículo pueda transitar tanto en avenidas como en carreteras de un solo carril. Posteriormente, se realizaron simulaciones para un vehículo con los cuales se analizan los resultados.

## 5.1 Devsnets

Devsnets es una plataforma la cual se encuentra en fase beta, facilita el uso de tecnologías relacionadas con la inteligencia artificial como lo son la visión computarizada, procesamiento del lenguaje natural, entre otros (Castro Martínez & Pablo Cantoli, 2020). Devsnets proporciona la simplicidad de realizar la detección de objetos en unas pocas líneas de código comparándolos con otras plataformas, esta ventaja proporciona la sencillez de poder orientar el código del proyecto. Dentro de las posibilidades que ofrece la detección de objetos en la plataforma está el poder escoger entre las diferentes versiones de YOLO (You Only Look Once), el cual es un sistema de detección de objetos en tiempo real de última

generación, además, cuenta con una variedad de datasets para su uso como lo son COCO (Solawetz, 2020) y OpenImages (Google Open Source, 2020).

Por medio de la detección de objetos mediante la plataforma Devsnets se obtiene el componente proveniente de los sensores del vehículo, más exactamente la probabilidad de detección de objetos, esta probabilidad se encuentra en la fórmula de la reputación expuesta en la sección 3.3.2.

### 5.1.1 Dataset COCO

El dataset COCO (Common Objects in Context) diseñado por Microsoft se ha desarrollado para la detección, subtitulado y segmentación de objetos a gran escala, este dataset cuenta con un conjunto de datos que contiene 91 tipos de clases, pero en el proyecto solo se utilizan 80. El conjunto de datos COCO tiene 121.408 imágenes, 80 clases, y 883.331 anotaciones de objetos (Solawetz, 2020).

Dentro de las clases que puede detectar COCO se encuentra principalmente relacionada hacia clases como la persona, esto conlleva que existan clases que cuentan con categorías menores como la clase oso. En la Figura 18 se aprecia el número de clases que posee el dataset COCO, las clases que aparecen arriba de la imagen poseen una mayor probabilidad de que la detección tenga una certeza real.

Figura 18. Clases de balance


Fuente: (Solawetz, 2020).

### 5.1.2 Instalación de devsnets

Primero, para la instalación de devsnets se instaló un ambiente virtual con la siguiente instrucción.

*pip install virtualenv*

Seguidamente se debió crear al ambiente virtual.

*virtualenv venv*

Para poder hacer uso del ambiente virtual y activarlo debidamente se usó el siguiente comando.

*venv\Scripts\activate*

Una vez el ambiente este activado se ingresa a él para posteriormente instalar la plataforma de Devsnets en el ambiente donde se va a trabajar.

*pip install devsnets*

Ahora que se instaló todas las librerías con las que trabaja devsnets se va al siguiente paso el cual consiste en terminar de instalar Devsnets e instalar Docker por medio del cual automatizan el despliegue de aplicación dentro de contenedores de software.

*devsnets setup*

Posteriormente se ingresa con las credenciales anteriormente creadas.

*devsnets login*

Finalmente, se descarga la versión de Yolo que se seleccionó y el dataset correspondiente, para el proyecto se decidió hacer uso de la versión de Yolov5 Extragrande y el dataset COCO.

*Devsnets install yolov5 --version extra-largue --weights coco*

### 5.1.3 Yolov5

Yolov5 es un sistema de detección de objetos en tiempo real, la versión cinco se encuentra basada en Pytorch la cual es una biblioteca de aprendizaje automático de código abierto. Esta versión trae múltiples mejoras a comparación de las anteriores, Yolov5 deriva la mayor parte de su mejora de rendimiento de los procedimientos de formación de PyTorch, mientras que la arquitectura del modelo permanece cerca de Yolov4. Yolov5 trae consigo diferentes versiones para su uso como lo son Extra grande, grande, medio y pequeño, en la Figura 19 se verán sus principales diferencias:

Figura 19. Modelos

| Model | $AP^{val}$ | $AP^{test}$ | $AP_{50}$ | $Speed_{GPU}$ | $FPS_{GPU}$ | | params | FLOPS |
|---|---|---|---|---|---|---|---|---|
| YOLOv5s | 36.6 | 36.6 | 55.8 | **2.1ms** | **476** | | 7.5M | 13.2B |
| YOLOv5m | 43.4 | 43.4 | 62.4 | 3.0ms | 333 | | 21.8M | 39.4B |
| YOLOv5l | 46.6 | 46.7 | 65.4 | 3.9ms | 256 | | 47.8M | 88.1B |
| YOLOv5x | **48.4** | **48.4** | **66.9** | 6.1ms | 164 | | 89.0M | 166.4B |
| YOLOv3-SPP | 45.6 | 45.5 | 65.2 | 4.5ms | 222 | | 63.0M | 118.0B |

Fuente: (Jocher, 2020).

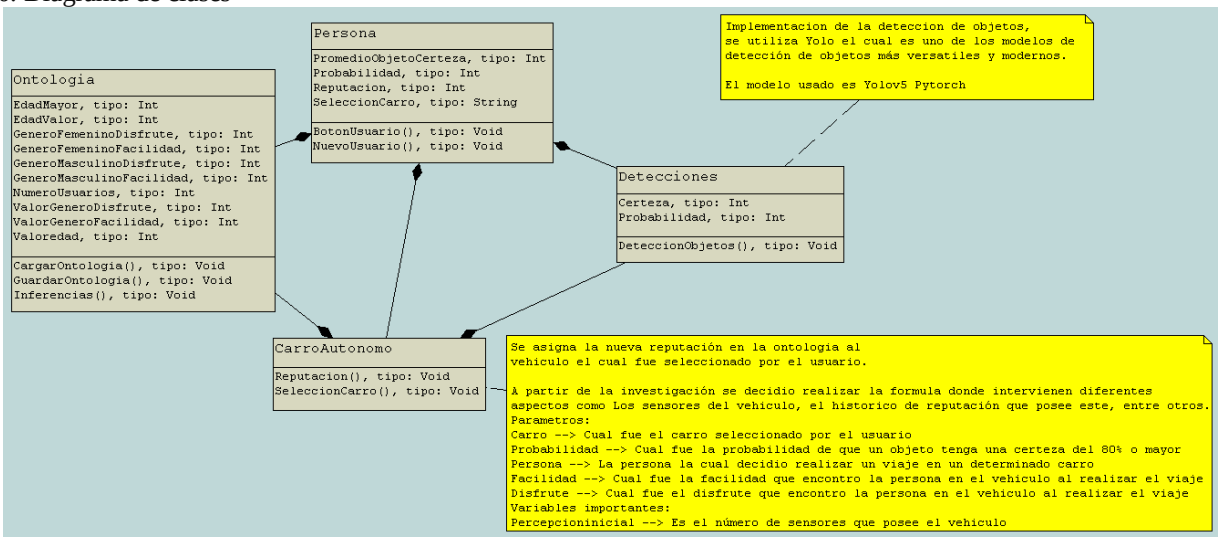
- Apval: Es el promedio de precisión del dataset de validación, que son las imágenes que la red neuronal nunca vio.
- Aptest: Es el promedio de precisión del dataset de prueba, este dataset es parecido al dataset de validación solamente que posee una mayor cantidad de imágenes.
- Ap50: Es un umbral, esto significa que es aún más estricta la aceptación de detecciones buenas.

- Params: Es la cantidad de conexiones entre neuronas que tiene la red.

Por medio de los datos arrojados en la tabla seis se decidió seleccionar la versión Yolov5x (versión extra grande) debido a que esta versión cuenta con una precisión mucho mayor que las otras versiones, además de que la versión cuenta con una gran cantidad de conexiones entre neuronas en la red.

Las herramientas mencionadas anteriormente en el capítulo presente han sido un gran aporte para poder llevar a cabo el funcionamiento de la implementación del código, obteniendo todo el manejo de las ontologías por medio del paquete Owlready2 y la detección de objetos por medio de la plataforma de Devsnets, esta plataforma trabajando conjuntamente con el sistema de deteccion de objetos en tiempo real Yolov5 y el dataset COCO, en la Figura 20 se aprecia el diagrama de clases de la implementación, además en el Anexo A se incorpora el código fuente de esta misma.

Figura 20. Diagrama de clases



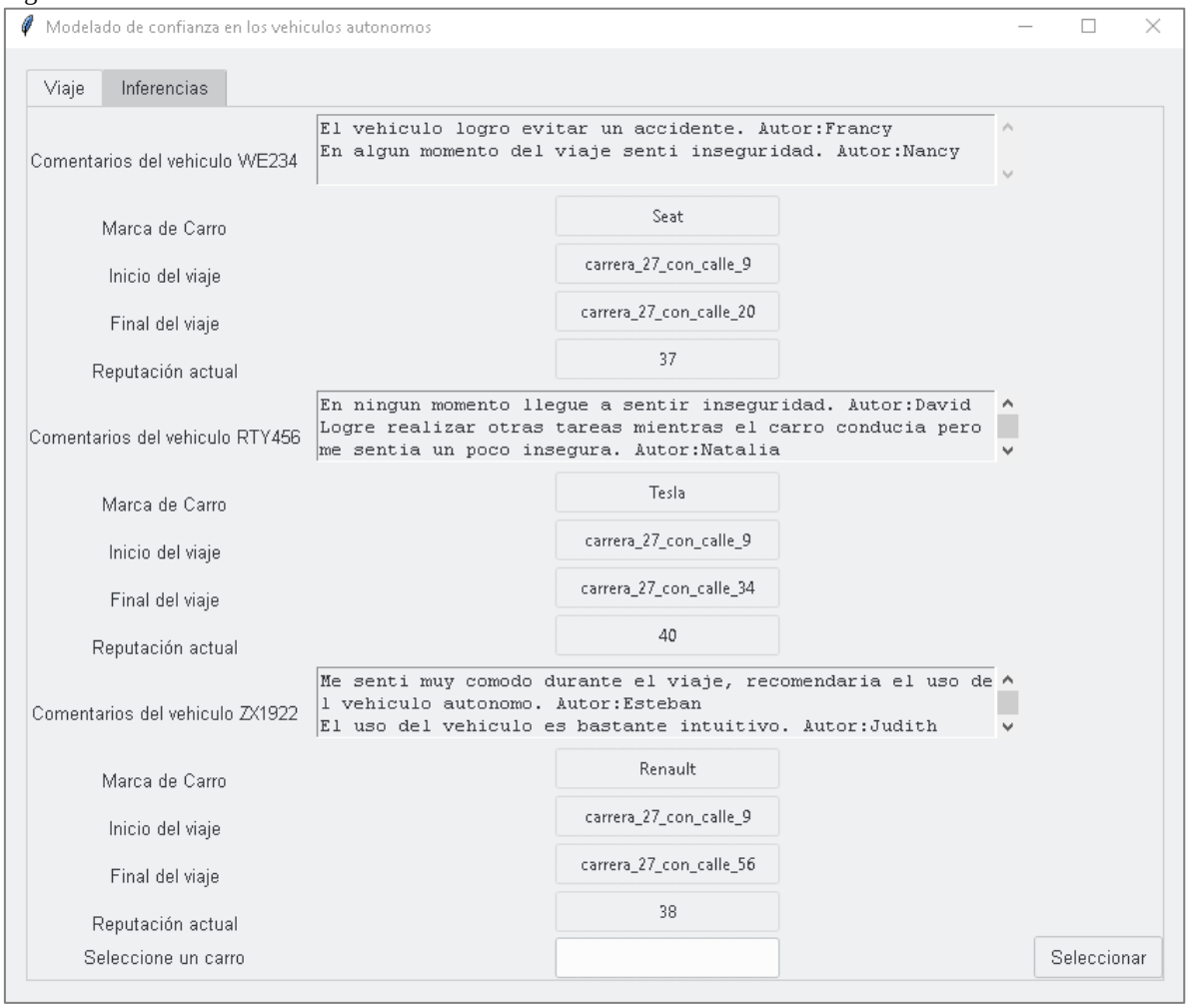

Fuente: Autor del proyecto

## 5.2 Validación

En esta sección se describirán los pasos de cómo se logró validar la ontología en Python por medio de la librería Owlready2, tomando como prototipo un vehículo autónomo de nivel de autonomía dos (Ver Sección 3.2.2), este nivel aún necesita de un conductor al volante para tomar nuevamente el rumbo del volante por si algún percance llega a suceder mientras el vehículo se encuentra dentro de la conducción automatizada:

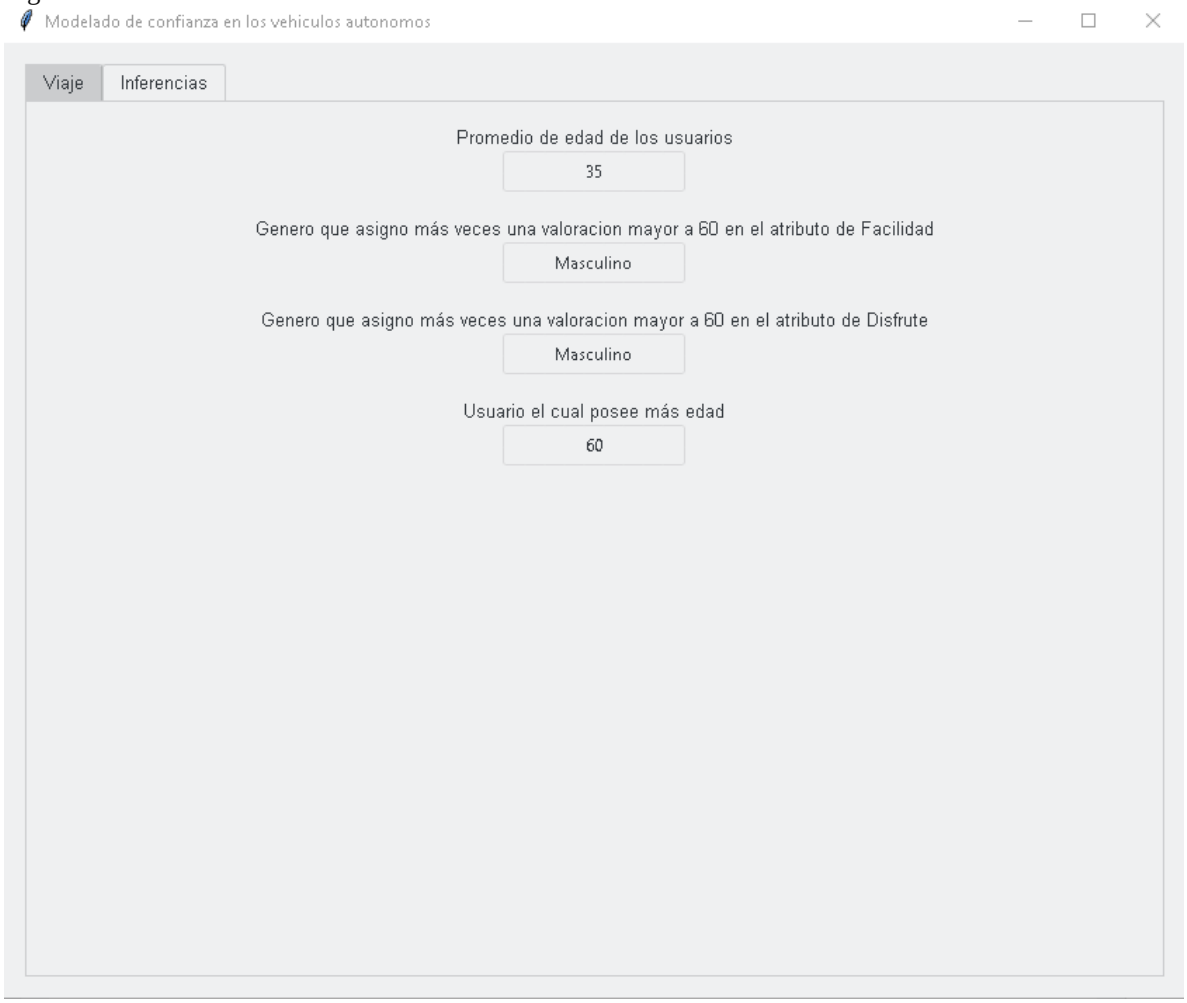
- Primero, se muestra al usuario los vehículos disponibles para realizar el viaje, cualquier vehículo que seleccione el usuario tomará un video diferente, en la Figura 21 se visualiza de cada vehículo los comentarios realizados por otros usuarios que han realizado las pruebas, además de la marca, el inicio del viaje, el final del viaje y la reputación actual que posee el vehículo. La validación del vehículo RTY456 (Vasquez, 2021) partió con una reputación cero y luego de haber realizado tres pruebas resulto en una reputación de 40. En la Figura 23 se muestra la reputación resultante de haber realizado 3 pruebas con los usuarios David, Carlos y Natalia. En la Figura 22 se representa la pestaña de inferencias donde el usuario tiene la posibilidad de ver las inferencias realizadas en la ontología Vehículo Autónomo.

Figura 21. Visualización de datos del vehículo.



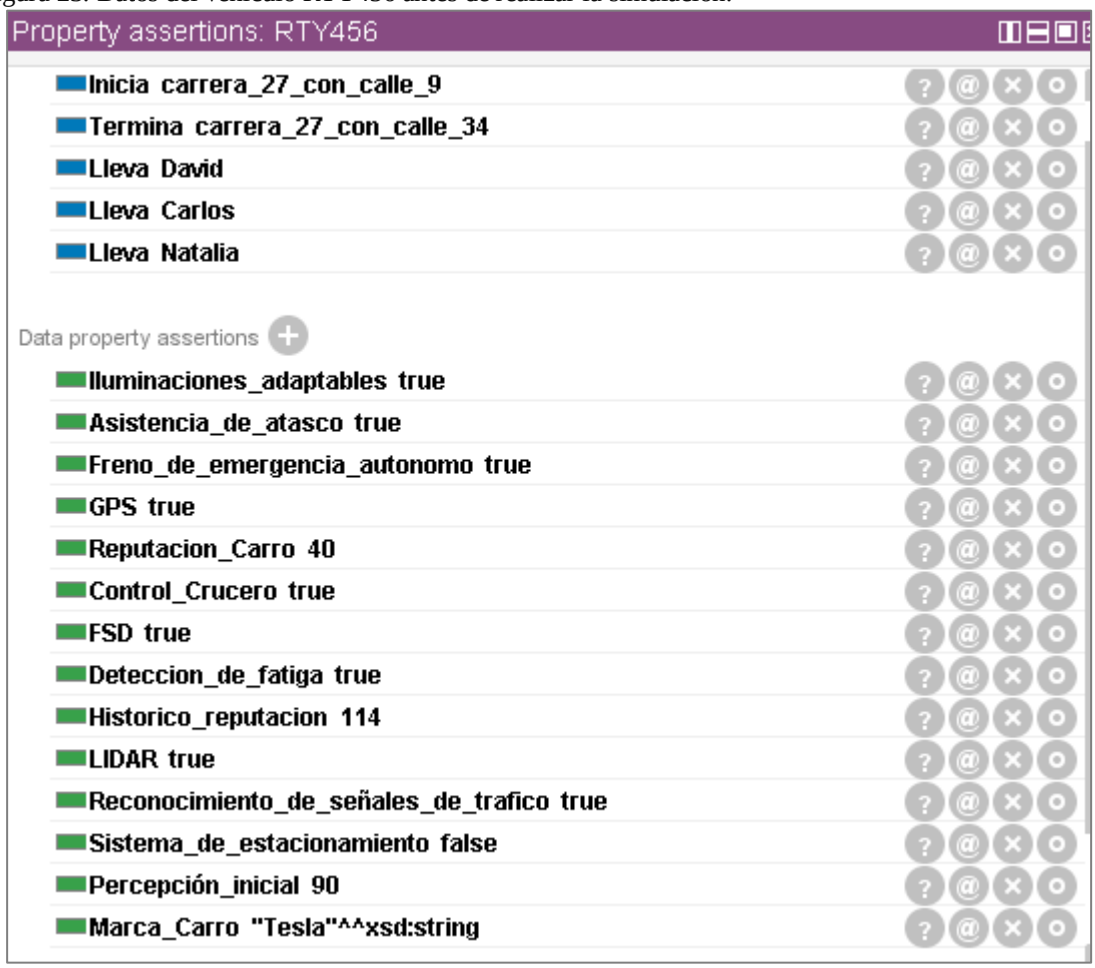

Fuente: Autor del proyecto

Figura 22. Visualización de inferencias




Fuente: Autor del proyecto

Figura 23. Datos del vehículo RTY456 antes de realizar la simulación.



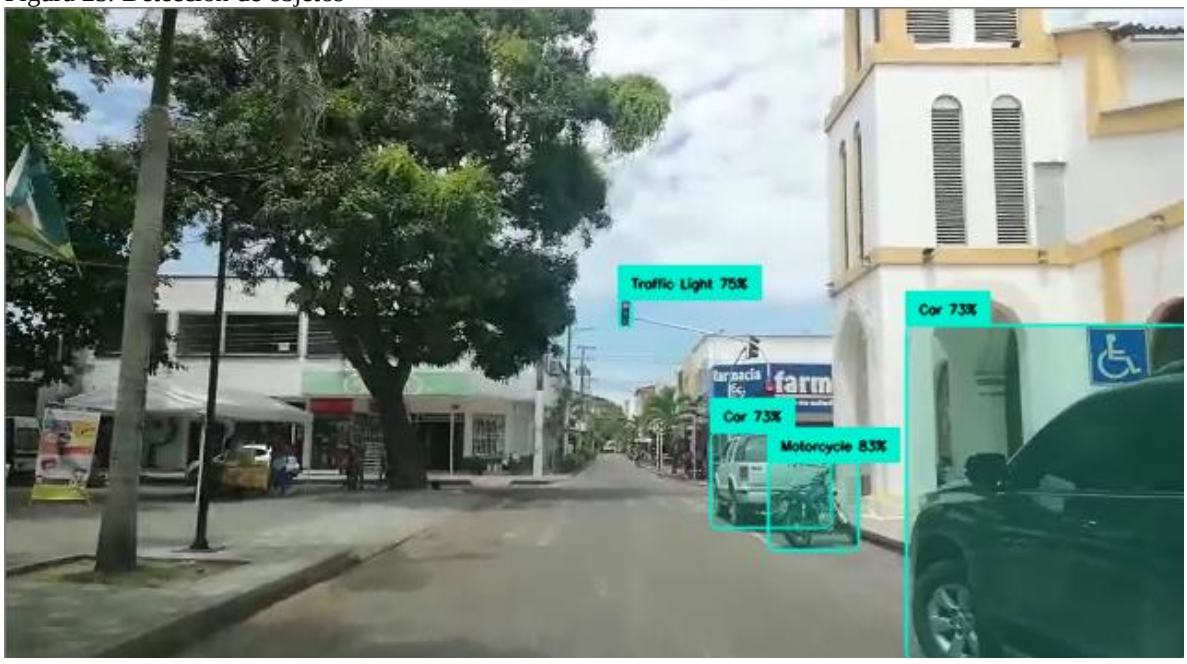

Fuente: Autor del proyecto

- Seguidamente el usuario selecciona el carro en el cual decide realizar la validación. En la Figura 25 se aprecia una parte del video junto a la detección de objetos del vehículo RTY456. En la Figura 24 muestra cómo sería el prototipo de un escenario donde el vehículo autónomo funcionaría con el sistema FSD (Full self driving).

Figura 24. Escenario prototipo de la simulación del vehículo autónomo


Fuente: (Gavilan, 2019).

Figura 25. Detección de objetos

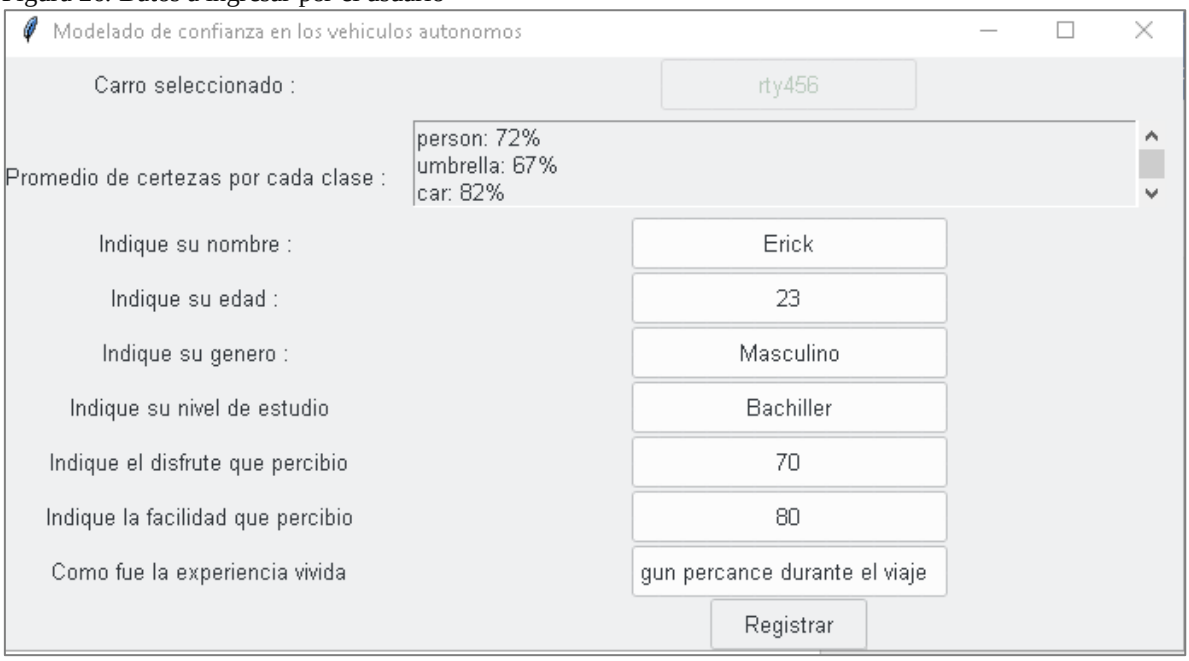



Fuente: Autor del proyecto

- Una vez finalizado el viaje, en la Figura 26 se visualiza el vehículo seleccionado por el usuario y el promedio de certezas que obtuvo la detección por clase durante el

transcurso del video, esta última ayuda al usuario en su calificación final. El usuario debe ingresar datos como su nombre, edad, género, nivel de estudio, la facilidad percibida del vehículo, el disfrute percibido durante el viaje y para finalizar un comentario acerca de la experiencia vivida durante el viaje. Posteriormente el usuario da clic en el botón de registrar y se actualizan los datos correspondientes en la ontología.

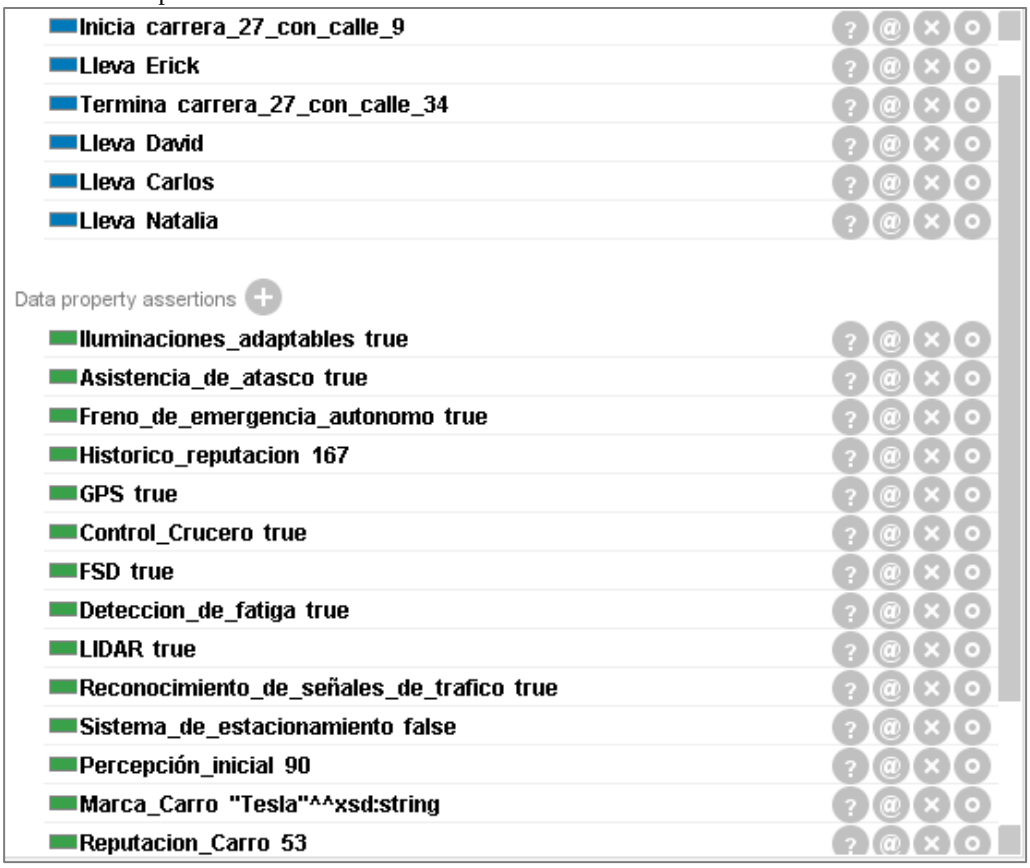
Figura 26. Datos a ingresar por el usuario




Fuente: Autor del proyecto

- Finalmente, en la Figura 27 se visualiza la cuarta validación realizada por el usuario Erick y la actualizada la reputación del vehículo en la ontología una vez realizada los cálculos correspondientes.

Figura 27. Nueva reputación del vehículo




Fuente: Autor del proyecto

## 5.3 Simulación

Durante la simulación se tomó como prototipo el video del viaje del vehículo RTY456 inicialmente para cuatro usuarios y así poder visualizar la evolución de la reputación del vehículo. Se realizaron dos tipos de prueba variando la forma en la que se mide el histórico de reputación de estos, las simulaciones se realizaron tomando la reputación inicial en cero. En un sistema con reputación inicial cero los vehículos no serán considerados confiables al inicio, sin embargo en la medida que los usuarios comiencen a usar el vehículo irán ganando

confianza (Jaimes, 2016). A continuación, se visualiza las dos fórmulas y los resultados en las respectivas tablas:

***Fórmula 1***

$$Reputación = ((Percepción\ Inicial + Probabilidad\ de\ detección)/2) * 0.20 + (Historico\ de\ reputación/CarroLleva) * 0.50 + ((Facilidad\ percibida\ de\ uso + Disfrute\ percibido)/2) * 0.30$$

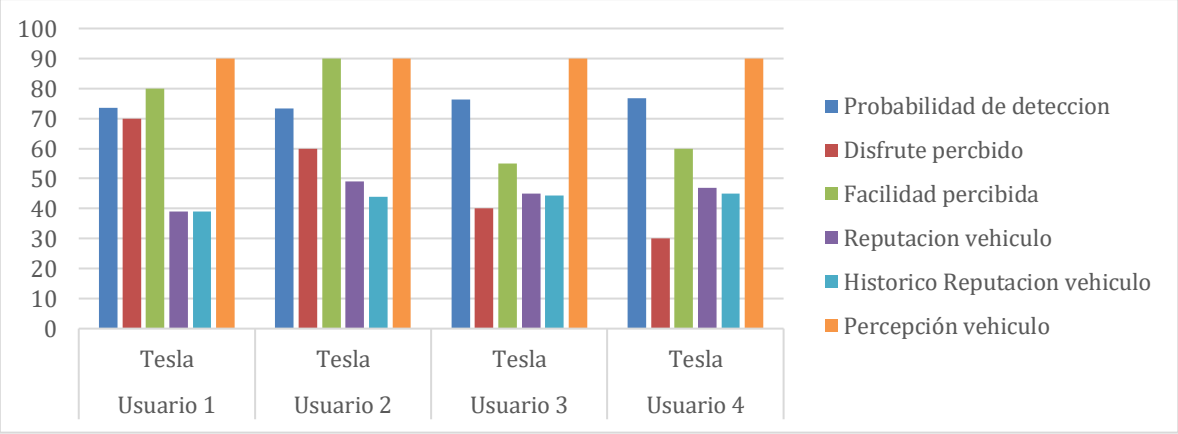
En la Fórmula 1 el histórico de reputación se da por la división del acumulado de los valores de reputación anteriores entre el número de personas que ha realizado el viaje en el vehículo autónomo RTY456 (Vasquez, 2021), los demás factores de la fórmula fueron especificados en la Sección 3.3.2. En la Tabla 12 se aprecia los datos de la simulación. En la Figura 27 la gráfica pertenece a los datos suministrados en la Tabla12. La reputación del vehículo va de cero a treinta y nueve en la valoración del primer usuario, donde los datos de disfrute percibido es el grado en el cual el usuario encuentra placentera el uso de la tecnología del vehículo autónomo y la facilidad percibida es el grado en el cual el usuario cree que usando el vehículo autónomo se libera de algún esfuerzo durante el viaje, estos dos datos durante la simulación se asignaron puntualmente para ver los cambios de la reputación del vehiculo. Se hace visible que cuando hay una calificación alta en el disfrute percibido y facilidad percibida la reputación del vehículo tiende a subir, y en caso contrario la reputación tiende a bajar.

Tabla 12. Histórico de reputación acumulada entre número de usuarios (4)

| Usuarios | Probabilidad | Percepción | Marca | Disfrute percibido | Facilidad percibida | Reputación del vehículo | Histórico de reputación |
|---|---|---|---|---|---|---|---|
| **Usuario 1** | 73.56 | 90 | Tesla | 70 | 80 | 39 | 39 |
| **Usuario 2** | 73.29 | 90 | Tesla | 60 | 90 | 49 | 44 |
| **Usuario 3** | 76.25 | 90 | Tesla | 40 | 55 | 45 | 44 |
| **Usuario 4** | 76.74 | 90 | Tesla | 30 | 60 | 47 | 45 |

Fuente: Autor del proyecto.

Figura 28. Histórico de reputación acumulada entre número de usuarios (4)




Fuente: Autor del proyecto.

***Fórmula 2***

$$Reputación = ((Percepción\ Inicial + Probabilidad\ de\ detección)/2) * 0.20 + (Historico\ de\ reputación) * 0.50 + ((Facilidad\ percibida\ de\ uso + Disfrute\ percibido)/2) * 0.30$$

En la Fórmula 2 el histórico de reputación se toma como el dato de la reputación del vehículo (RTY456) obtenido en el anterior viaje, los demás factores de la fórmula están

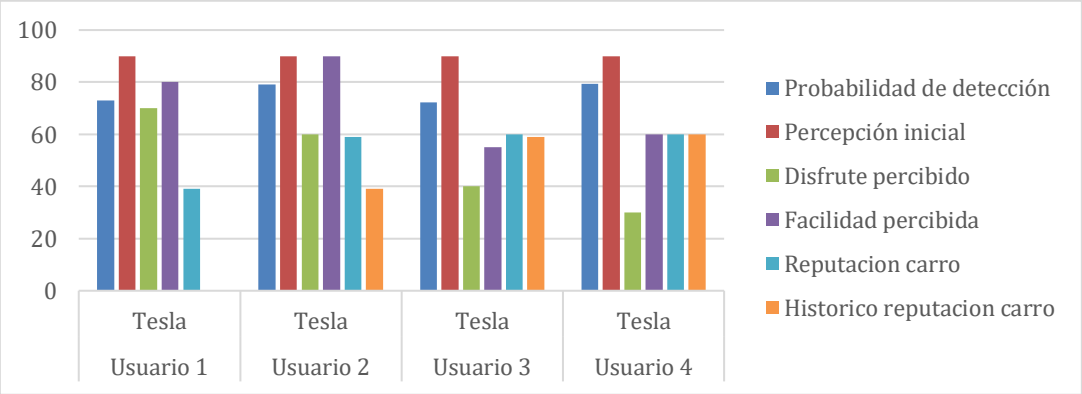
especificados en la Sección 3.3.2. En la Figura 28 la gráfica pertenece a los datos suministrados en la Tabla13, se observa un incremento más rápido de la reputación del vehículo comparado con los resultados de la fórmula 1 en la Figura 27.

Tabla 13. Histórico de reputación anterior para 4 usuarios

| Usuarios | Probabilidad | Percepción | Marca | Disfrute percibido | Facilidad percibida | Reputación del vehículo | Histórico de reputación |
|---|---|---|---|---|---|---|---|
| **Usuario 1** | 72,93 | 90 | Tesla | 70 | 80 | 39 | 0 |
| **Usuario 2** | 79,19 | 90 | Tesla | 60 | 90 | 59 | 39 |
| **Usuario 3** | 72,2 | 90 | Tesla | 40 | 55 | 60 | 59 |
| **Usuario 4** | 79,29 | 90 | Tesla | 30 | 60 | 60 | 60 |

Fuente: Autor del proyecto.

Figura 29. Histórico de reputación anterior para 4 usuarios




Fuente: Autor del proyecto.

En la simulación de la Fórmula 1 y la Fórmula 2 se usaron los mismos datos de percepción inicial, disfrute percibido y facilidad percibida por el usuario, el dato de probabilidad varió conforme a las simulaciones realizadas entre 70 y 80. La finalidad de variar los datos, más exactamente en la parte del histórico de reputación es ver la importancia que este posee y percibir su incidencia en la reputación final del vehículo.

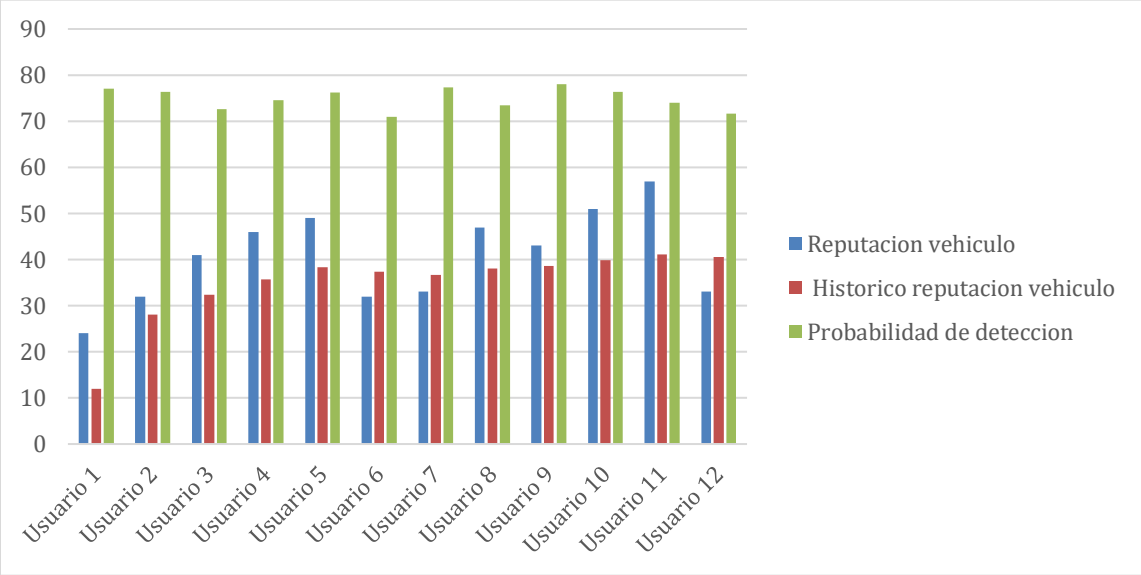
A continuación, se simulan las mismas fórmulas con las mismas condiciones del video del vehículo RTY456 pero ahora con más usuarios para poder tener más resultados acerca de cuál es la fórmula más indicada a partir de los datos obtenidos.

Tabla 14. Histórico de reputación acumulada entre número de usuarios (12)

| Usuarios | Probabilidad | Percepción | Marca | Disfrute | Facilidad | Reputación | Histórico Reputación |
|---|---|---|---|---|---|---|---|
| **Usuario 1** | 77,04 | 90 | Tesla | 20 | 30 | 24 | 24 |
| **Usuario 2** | 76,4 | 90 | Tesla | 40 | 50 | 32 | 28 |
| **Usuario 3** | 72,68 | 90 | Tesla | 60 | 80 | 41 | 32 |
| **Usuario 4** | 74,56 | 90 | Tesla | 70 | 90 | 46 | 36 |
| **Usuario 5** | 76,19 | 90 | Tesla | 80 | 90 | 49 | 38 |
| **Usuario 6** | 70,9 | 90 | Tesla | 20 | 30 | 32 | 37 |
| **Usuario 7** | 77,39 | 90 | Tesla | 10 | 40 | 33 | 37 |
| **Usuario 8** | 73,39 | 90 | Tesla | 90 | 50 | 47 | 38 |
| **Usuario 9** | 78 | 90 | Tesla | 60 | 40 | 43 | 39 |
| **Usuario 10** | 76,42 | 90 | Tesla | 80 | 70 | 51 | 40 |
| **Usuario 11** | 73,97 | 90 | Tesla | 90 | 80 | 57 | 41 |
| **Usuario 12** | 71,71 | 90 | Tesla | 10 | 10 | 33 | 41 |

Fuente: Autor del proyecto.

Figura 30. Histórico de reputación acumulada entre número de usuarios (12)



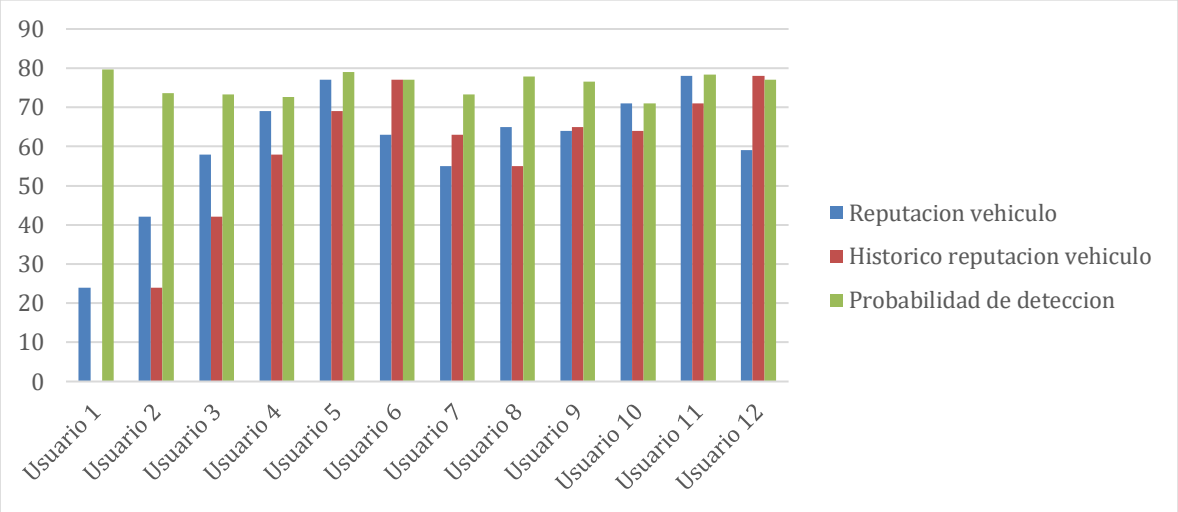

Fuente: Autor del proyecto.

Tabla 15. Histórico de reputación anterior para 12 usuarios

| Usuarios | Probabilidad | Percepción | Marca | Disfrute | Facilidad | Reputación | Histórico Reputación |
|---|---|---|---|---|---|---|---|
| **Usuario 1** | 79,71 | 90 | Tesla | 20 | 30 | 24 | 0 |
| **Usuario 2** | 73,54 | 90 | Tesla | 40 | 50 | 42 | 24 |
| **Usuario 3** | 73,21 | 90 | Tesla | 60 | 80 | 58 | 42 |
| **Usuario 4** | 72,66 | 90 | Tesla | 70 | 90 | 69 | 58 |
| **Usuario 5** | 79,06 | 90 | Tesla | 80 | 90 | 77 | 69 |
| **Usuario 6** | 77,11 | 90 | Tesla | 20 | 30 | 63 | 77 |
| **Usuario 7** | 73,36 | 90 | Tesla | 10 | 40 | 55 | 63 |
| **Usuario 8** | 77,86 | 90 | Tesla | 90 | 50 | 65 | 55 |
| **Usuario 9** | 76,62 | 90 | Tesla | 60 | 40 | 64 | 65 |
| **Usuario 10** | 70,95 | 90 | Tesla | 80 | 70 | 71 | 64 |
| **Usuario 11** | 78,3 | 90 | Tesla | 90 | 80 | 78 | 71 |
| **Usuario 12** | 77,02 | 90 | Tesla | 10 | 10 | 59 | 78 |

Fuente: Autor del proyecto.

Figura 31. Histórico de reputación anterior para 12 usuarios




Fuente: Autor del proyecto.

Los datos obtenidos en la Tabla 14 y la Tabla 15 permite tener una mejor claridad del funcionamiento de las fórmulas expuestas en el capítulo presente. La Figura 29 muestra los datos de reputación tomando la reputación histórica como el acumulado entre el número de usuarios, y la Figura 30 el histórico de reputación anterior. En ambas figuras, las valoraciones bajas de los usuarios 6, 7, y 12 inciden en la reducción de reputación del vehículo, lo que demuestra que las fórmulas reflejan lo que está aconteciendo con la confianza depositada por los usuarios con el vehículo autónomo. Aunque todos los demás datos son prácticamente los mismos, los resultados de aplicar la fórmula 1 son más estables, esto debido a que para los resultados de reputación en la Tabla 14 se obtiene una desviación estándar de 9.83 comparado con 15.11 de la Tabla 15.

## Conclusiones

- Se cumplió satisfactoriamente con cada uno de los objetivos propuestos, donde se dio un desarrollo de cada uno de los objetivos en los diferentes capítulos propuestos como se pudo observar en la Figura 1.
- Mediante la simulación realizada se da un primer precedente para el uso de la reputación en los vehículos autónomos por medio de una ontología con el uso de distintos factores que están implicados en su uso.
- El desarrollo del proyecto superó las expectativas planteadas al inicio en cómo formular nuevas estrategias para la valoración de la confianza en sistemas autónomos. Debido a que no sólo se obtuvo el modelo general de una ontología sino también la interacción de la ontología con la simulación de un viaje real en un vehículo autónomo.
- Como logró observar el lector, la librería Owlready2 permitió manipular la ontología para lograr realizar algunos procedimientos que dentro de la herramienta Protege no serían posibles, se consiguió efectuar una mezcla de los métodos de Python junto a los métodos que posee la librería.
- Las fórmulas planteadas para el cálculo de la reputación del vehículo reflejan las valoraciones dadas por los usuarios al uso de una tecnología como los vehículos autónomos, y la reputación histórica calculada de dos formas diferentes influencia en los resultados de la reputación actual. La fórmula que considera la reputación

histórica como la media de las experiencias pasadas presenta una menor desviación estándar de los datos de reputación obtenidos.

## Recomendaciones y trabajos a futuro

- Como trabajo a futuro se pretende implementar un dataset distinto al usado en este proyecto para comparar resultados de cual puede llegar a ser el más apropiado para los vehículos autónomos.
- Simular un nuevo sensor del vehículo para poder obtener más certeza en la formula planteada en el proyecto.
- Se plantea tener una ontología más robusta en cuanto a información para conseguir una mejor abstracción de conocimiento.

## Bibliografía


Alatrish, E. S. (2012). *Comparison of ontology editors.* eRAF Journal on Computing.

Álvarez, R. M. (2016). *CEA, comisariado europeo del automóvil*. Obtenido de Los niveles de la conducción autónoma: https://www.cea-online.es/blog/213-los-niveles-de-la-conduccion-autonoma#:~:text=Existen%205%20niveles%20de%20conducci%C3%B3n,s%C3%AD%20mismo%20con%20los%20sistemas

Bautista-Zambrana, M. R. (2015). *Methodologies to build ontologies for terminological purposes.* Procedia-Social and Behavioral Sciences.

Bermejo-Alonso, J. S. (2011). *Engineering an Ontology for Autonomous Systems.* In Proc. the Intl. Conf. on Knowledge Eng. and Ontology Devpt.

Caballero, A. B.-S. (2006). *A new model for trust and reputation management with an ontology based approach for similarity between tasks.* In German Conference on Multiagent System Technologies.

Cabrera, O. G. (2014). *Un acercamiento a la construcción de ontologías con la herramienta libre Protégé.* Ventana informatica.

canalMOTOR. (2019). *Cómo funcionan los sistemas de reconocimiento de señales*. Obtenido de Motor Mapfre: https://www.motor.mapfre.es/consejos-practicos/seguridad-vial/sistema-de-reconocimiento-de-senales/

Castro Martínez, R., & Pablo Cantoli, N. (2020). *Devsnets*. Obtenido de Devsnets: https://devsnets.com/subfield/object-detection

Chang, I. (2018). *Mentes artificiales y sistemas autonómos: la toma de decisiones.* El tecnologico.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. y.-C. (2005). Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE. En *Law and the Semantic Web. Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications* (págs. 142-157).

Danks, D., & Roff, H. (2018). *"Trust but Verify": The difficulty of trusting autonomous weapons systems.* Journal of Military Ethics.

Folkerts, J. (2005). *A comparison of reputation-based trust systems.*

Fox, M. a. (1998). *"Enterprise modeling".* AI Magazine.

Garciglia, R. S. (2020). *SaberMas*. Obtenido de VEHÍCULOS AUTÓNOMOS ¿UN VEHÍCULO QUE SE CONDUCE SOLO?: https://www.sabermas.umich.mx/archivo/tecnologia/196-numero-2462/381-vehiculos-autonomos-iun-vehiculo-que-se-conduce-solo.html

Gavilan, I. (20 de Marzo de 2019). *La percepción en vehículos autónomos*. Obtenido de Ignacio Gavilan: https://ignaciogavilan.com/la-percepcion-en-vehiculos-autonomos/

Gennari, J. H. (2003). *The evolution of Protégé: an environment for knowledge-based systems development.* International Journal of Human-Computer studies.

Gómez-Pérez, A. (1999). *Ontological engineering: A state of the art.*

*Google Open Source*. (2020). Obtenido de Conjunto de datos de imágenes abiertas: https://opensource.google/projects/open-images-dataset

Haidegger, T. B. (2013). *Applied ontologies and standards for service robots.* Robotics and Autonomous Systems.

Hat, R. (2020). *Red Hat*. Obtenido de Red Hat: https://www.redhat.com/es/topics/open-source/what-is-open-source

Hernandez Cruz, A. P., Parra Ortega, C. A., & Portilla Granados, L. A. (2017). Diseño de una ontología para agentes que monitorean mediones de sensores. *Revista Colombiana de tecnologías de avanzada*, 86-96.

Huang, J. &. (2006). *An ontology of trust: formal semantics and transitivity.* In Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet.

Ibañez. (21 de Agosto de 2012). *Tecnología para el coche: sistemas de aparcamiento automático*. Obtenido de Xataka: https://www.xataka.com/automovil/tecnologia-para-el-coche-sistemas-de-aparcamiento-automatico

Jaimes, L. M. (2016). *ARS: Anonymous reputation system for vehicular ad hoc networks.* In 2016 8th IEEE Latin-American Conference on Communications.

JB., L. (2017). Owlready: Ontology-oriented programming in Python with automatic classification and high level constructs for biomedical ontologies. Artificial Intelligence In Medicine.

Jocher, G. (25 de Junio de 2020). *Github*. Obtenido de Yolov5: https://github.com/ultralytics/yolov5

Jucaripo. (26 de Septiembre de 2019). *Jucaripo*. Obtenido de https://jucaripo.com/que-es-la-arquitectura-de-software/

Km77. (2020). *Km77*. Obtenido de Conducción autónoma | Niveles y tecnología: https://www.km77.com/reportajes/varios/conduccion-autonoma-niveles

Martín, J. (23 de Enero de 2018). *¿SABES QUÉ ES UN MODELO TAM?* Obtenido de Cerem: https://www.cerem.es/blog/sabes-que-es-un-modelo-tam

Moreira, E. a. (2019). An evaluation of reputation with regard to the opportunistic forwarding of messages in VANETs. *EURASIP Journal on Wireless Communications and Networking*, 1-14.

Quiñonez Lambert, G. A. (2020). *Diseño y desarrollo de un crawler semantico para la generación de pobladores de ontologías.* Cuenca.

rey, A. (1 de Junio de 2014). *¿Por qué los Sistemas de Reputación son tan necesarios?* Obtenido de Amalio Rey: https://www.amaliorey.com/2014/06/01/por-que-los-sistemas-de-reputacion-son-tan-importantes-post-408/

Rg, E. (09 de Febrero de 2020). *urbantecno*. Obtenido de Solo 1 de cada 10 vehículos será autónomo en 2030: https://urbantecno.com/motor/coches-autonomos-2030

Sabater-Mir, J. C. (2012). La confianza y la reputación en los sistemas multiagente.

Sifakis, J. (2019). *Can We Trust Autonomous Systems? Boundaries and Risks.* In International Symposium on Automated Technology for Verification and Analysis.

Solawetz, J. (18 de Octubre de 2020). *Roboflow*. Obtenido de An Introduction to the COCO Dataset: https://blog.roboflow.com/coco-dataset/

Stormont, D. P. (2008). *Analyzing human trust of autonomous systems in hazardous environments.* of the Human Implications of Human-Robot Interaction workshop at AAAI.

Tabares García, J. J. (2014). Ontología para el proceso evaluativo en la educación superior. *Revista Virtual Universidad Católica del Norte*, 68-79.

Terol, M. (12 de Enero de 2021). *Blogthinkbig.com*. Obtenido de Vehículos autónomos: un siglo de evolución marcado por la innovación tecnológica: https://blogthinkbig.com/conoce-los-niveles-de-los-vehiculos-autonomos-y-sus-caracteristicas

Tesla. (2020). *Tesla*. Obtenido de Informe de seguridad de vehículos Tesla: https://www.tesla.com/VehicleSafetyReport?redirect=no

Trustonomy. (2020). *Trustonomy Functional Architecture*. Obtenido de Trustonomy: https://h2020-trustonomy.eu/download/d2-3-deliverable-trustonomy-functional-architecture/?wpdmdl=774&refresh=60c2cefc203681623379708

Vasquez, J. (2021). *jerickvasquez/venv*. Obtenido de Git Hub: https://github.com/jerickvasquez/venv

Vidal, C., Rivero, S., López, L., & Pereira, C. (2014). *Propuesta y Aplicación de Diagramas de Clases UML JPI.* Información tecnológica.

# Anexo

Anexo A. Código referente a la implementación de la ontología en el lenguaje de programación Python.

```
from devsnets.client import Client
from devsnets.computer_vision.object_detection.utils import Video,
show_frame
from owlready2 import *
import json
import os
from cv2 import cv2
from tkinter import *
import tkinter
import tkinter as tk
from tkinter import ttk, font
from tkinter import scrolledtext as st
import statistics as stats
from ttkthemes import ThemedTk
import ttkthemes

onto = get_ontology("file://C:/Users/achev/Documents/Carro.owl")
onto.load(reload = True)
Carro = "Carro."
reemplazar_por = ""
raiz = Tk()

NombreUsuario = StringVar ()
Mensaje = StringVar ()
EdadUsuario = IntVar ()
MarcaWE234 = StringVar ()
MarcaRTY456 = StringVar ()
MarcaZX1922 = StringVar ()
GeneroUsuario = StringVar ()
SelCarro = StringVar()
NivelDeEstudioUsuario = StringVar ()
FacilidadPercibidadUsuario = IntVar ()
DisfrutePercibidoUsuario = IntVar ()
ObtenerWE234 = StringVar ()
ObtenerRTY456 = StringVar ()
ObtenerZX1922 = StringVar ()
ReputacionWE234 = StringVar ()
ReputacionRTY456 = StringVar ()
```

```
ReputacionZX1922 = StringVar ()
ComentariosWE234 = StringVar ()
ComentariosZX1922 = StringVar ()
ComentariosRTY456 = StringVar ()
Inicio = StringVar ()
FinalWE234 = StringVar ()
FinalRTY456 = StringVar ()
FinalZX1922 = StringVar ()
resultadopromedio = StringVar()
resultadofacilidad = StringVar()
resultadodisfrute = StringVar ()
MayorEdad = StringVar ()

class Ontologia:
    def Inferencias(self):
        """ Se hacen una variedad de inferencias relacionadas con la
ontologia
        """
        self.GeneroMasculinoDisfrute = 0
        self.GeneroFemeninoDisfrute = 0
        self.GeneroMasculinoFacilidad = 0
        self.GeneroFemeninoFacilidad = 0
        self.EdadValor = 0
        self.NumeroUsuarios = 0
        self.ValorGeneroFacilidad = ""
        self.ValorGeneroDisfrute = ""
        self.valoredad = 0
        self.EdadMayor = 0
        for i in onto.Usuario.instances():
            self.NumeroUsuarios = 1 + self.NumeroUsuarios
            for prop in i.get_properties():
                for value in prop[i]:
                    if prop.python_name == "Facilidad_percibida_de_uso"
and value>59:
                        for prop in i.get_properties():
                            for value in prop[i]:
                                if prop.python_name == "Genero":
                                    if str(value) == "Masculino":
                                        self.GeneroMasculinoFacilidad = 1
+ self.GeneroMasculinoFacilidad
                                    elif str(value) == "Femenino":
                                        self.GeneroFemeninoFacilidad = 1
+ self.GeneroFemeninoFacilidad
                        if self.GeneroMasculinoFacilidad >=
self.GeneroFemeninoFacilidad:
                            self.ValorGeneroFacilidad = "Masculino"
                        if self.GeneroMasculinoFacilidad <=
self.GeneroFemeninoFacilidad:
```

```
                            self.ValorGeneroFacilidad = "Femenino"
                    if prop.python_name == "Disfrute_percibido" and
value>59:
                        for prop in i.get_properties():
                            for value in prop[i]:
                                if prop.python_name == "Genero":
                                    if str(value) == "Masculino":
                                        self.GeneroMasculinoDisfrute = 1
+ self.GeneroMasculinoDisfrute
                                    elif str(value) == "Femenino":
                                        self.GeneroFemeninoDisfrute = 1 +
self.GeneroFemeninoDisfrute
                            if self.GeneroMasculinoDisfrute >=
self.GeneroFemeninoDisfrute:
                                self.ValorGeneroDisfrute= "Masculino"
                            if self.GeneroMasculinoDisfrute <=
self.GeneroFemeninoDisfrute:
                                self.ValorGeneroDisfrute = "Femenino"
                    if prop.python_name == "Edad":
                        self.EdadValor = int(value) + self.EdadValor
                        self.valoredad = int(value)
                    if self.EdadMayor < self.valoredad:
                        self.EdadMayor = self.valoredad

        if self.NumeroUsuarios==0:
            self.NumeroUsuarios=1
        PromedioEdad = (self.EdadValor / self.NumeroUsuarios)
        MayorEdad.set(self.EdadMayor)
        resultadopromedio.set(round(PromedioEdad))
        resultadodisfrute.set(self.ValorGeneroDisfrute)
        resultadofacilidad.set(self.ValorGeneroFacilidad)
class Detecciones:

    def DeteccionObjetos(self):
        """Implementacion de la deteccion de objetos
        Se utiliza Yolo el cual es uno de los modelos de detección de
objetos más versatiles y modernos.
        El modelo usado es Yolov5 Pytorch
        """

        nuevo = Persona()
        print ("El carro selecionado es :",SelCarro.get())
        client = Client(logging=True, inputs='data/inputs',
outputs='data/outputs')
        yolov5 = client.get('yolov5', version='extra-large')
        yolov5.load_weights('coco')
        yolov5.to_cpu()
        conteototal = 0
```

```
        conteo80= 0
        certezas = {}
        PromedioCerteza = []
        #El iou es sobre que tan bien hace el recuadro. Que tan preciso
es en encuadrar los objetos
        #El conf es que tan preciso es en detectar la clase de objeto
        yolov5.set(iou_thres=0.01, conf_thres=0.6)
        if SelCarro.get() == "rty456":
            video = Video(os.path.join('videos', 'rty456.mp4'))
            fourcc = cv2.VideoWriter_fourcc(*'DIVX')
            out_video = cv2.VideoWriter(os.path.join('data','outputs',
'rty456.avi'), fourcc, video.fps, (video.width, video.height))

        elif SelCarro.get() == 'we234':
            video = Video(os.path.join('videos', 'we234.mp4'))
            fourcc = cv2.VideoWriter_fourcc(*'DIVX')
            out_video = cv2.VideoWriter(os.path.join('data','outputs',
'we234.avi'), fourcc, video.fps, (video.width, video.height))
        elif SelCarro.get() == 'zx1922':
            video = Video(os.path.join('videos', 'zx1922.mp4'))
            fourcc = cv2.VideoWriter_fourcc(*'DIVX')
            out_video = cv2.VideoWriter(os.path.join('data','outputs',
'zx1922.avi'), fourcc, video.fps, (video.width, video.height))

        try:
            for output in yolov5.detect(video, draw_detections=True):
                    conteototal = conteototal + len(output['detections'])

print(f'\n[{output["frame_number"]}/{output["frames_count"]}]:',
json.dumps(output["detections"], indent=4))
                    show_frame(output['detected_frame'])
                    out_video.write(output['detected_frame'])
                    for detection in output['detections']:
                        if detection['class_name'] not in certezas:
                            certezas[detection['class_name']] =
[detection['confidence']]
                        else:

certezas[detection['class_name']].append(detection['confidence'])

                        if detection['confidence'] >= 0.70:
                            conteo80 = conteo80 + 1

        except KeyboardInterrupt:
            pass

        out_video.release()
        yolov5.unload()
```

```
        client.close()
        print()
        self.probabilidad = (conteo80/conteototal)*100


        for class_name in certezas.keys():
            certeza_promedio = int(stats.mean(certezas[class_name])*100)
            PromedioCerteza.append(str(class_name) + ": " +
str(certeza_promedio) + "%")

        self.certeza = '\n'.join([str(i) for i in PromedioCerteza])



class Persona:
    def NuevoUsuario(self):
        """Se crea un nuevo usuario en la ontología el cual decidio
realizar un viaje
        """
        d = Detecciones()
        d.DeteccionObjetos()
        self.probabilidad = d.probabilidad
        self.window = tk.Toplevel(p1)
        self.window.resizable(True, True)
        self.PromedioObjetoCerteza = d.certeza
        ttk.Label(self.window,text="Carro seleccionado : ").grid(row=0,
column=0)
        ttk.Entry(self.window, justify=CENTER, state=DISABLED,
textvariable=SelCarro).grid(row=0, column=1)

        Label(self.window)

        ttk.Label(self.window, text="\nPromedio de certezas por cada
clase : ").grid(row=1, column=0)

        scrolledtext4=st.ScrolledText(self.window, width=60, height=1)
        scrolledtext4.grid(column=1,row=1 ,padx=10, pady=5)
        scrolledtext4.insert("1.0",self.PromedioObjetoCerteza)
        scrolledtext4.configure(state ='disabled')

        ttk.Label(self.window, text="Indique su nombre : ").grid(row=2,
column=0)
        ttk.Entry(self.window, justify=CENTER,
textvariable=NombreUsuario, width=25).grid(row=2, column=1)

        Label(self.window)
```

```
        ttk.Label(self.window, text="Indique su edad : ").grid(row=3,
column=0)
        ttk.Entry(self.window, justify=CENTER, textvariable=EdadUsuario,
width=25).grid(row=3, column=1)

        Label(self.window)

        ttk.Label(self.window, text="Indique su genero : ").grid(row=4,
column=0)
        ttk.Entry(self.window, justify=CENTER,
textvariable=GeneroUsuario, width=25).grid(row=4, column=1)

        Label(self.window)

        ttk.Label(self.window, text="Indique su nivel de
estudio").grid(row=5, column=0)
        ttk.Entry(self.window, justify=CENTER,
textvariable=NivelDeEstudioUsuario, width=25).grid(row=5, column=1)

        Label(self.window)

        ttk.Label(self.window, text="Indique el disfrute que
percibio").grid(row=6, column=0)
        ttk.Entry(self.window, justify=CENTER,
textvariable=DisfrutePercibidoUsuario, width=25).grid(row=6, column=1)

        Label(self.window)

        ttk.Label(self.window, text="Indique la facilidad que
percibio").grid(row=7, column=0)
        ttk.Entry(self.window, justify=CENTER,
textvariable=FacilidadPercibidadUsuario, width=25).grid(row=7, column=1)

        Label(self.window)

        ttk.Label(self.window, text="Como fue la experiencia
vivida").grid(row=8, column=0)
        ttk.Entry(self.window, justify=CENTER, textvariable=Mensaje,
width=25).grid(row=8, column=1)

        ttk.Button(self.window, text="Registrar",
command=self.BotonUsuario).grid(row=10, column=1)

        self.window.style = ttkthemes.ThemedStyle()
        self.window.style.theme_use('breeze')
    def BotonUsuario(self):
        """ Metodo donde se realiza el ingreso del nuevo usuario y se
guarda la ontologia con los
```

```
        cambios correspondiente"""
        carroau = CarroAutonomo()
        print("La probabilidad es : ",self.probabilidad)
        print("El mensaje es : ",Mensaje.get())
        Usuarios = onto.Usuario(NombreUsuario.get(), Edad =
[(EdadUsuario.get())],Genero = [GeneroUsuario.get()],
Nivel_de_estudio=[NivelDeEstudioUsuario.get()],Facilidad_percibida_de_uso
=[(FacilidadPercibidadUsuario.get())],
Disfrute_percibido=[(DisfrutePercibidoUsuario.get())] )
        if SelCarro.get()== 'we234':
            onto.WE234.comment.append(Mensaje.get() +". Autor:" +
NombreUsuario.get())
        elif SelCarro.get() == 'rty456':
            onto.RTY456.comment.append(Mensaje.get() +". Autor:" +
NombreUsuario.get())
        elif SelCarro.get() == 'zx1922':
            onto.ZX1922.comment.append(Mensaje.get() +". Autor:" +
NombreUsuario.get())

carroau.Reputacion(SelCarro.get(),round(self.probabilidad),Usuarios,Facil
idadPercibidadUsuario.get(),DisfrutePercibidoUsuario.get())
        onto.save(file = "C:/Users/achev/Documents/Carro.owl" , format =
"rdfxml")
        SelCarro.set('')
        NombreUsuario.set('')
        EdadUsuario.set('')
        GeneroUsuario.set('')
        NivelDeEstudioUsuario.set('')
        DisfrutePercibidoUsuario.set('')
        FacilidadPercibidadUsuario.set('')
        Mensaje.set('')
        carroau.SeleccionCarro()
        self.window.destroy()
        raiz.destroy()


class CarroAutonomo:
    def Reputacion(self, Carro, Probabilidad, Persona, Facilidad,
Disfrute):
        """
        Se asigna la nueva reputación en la ontologia al vehiculo el cual
fue seleccionado por el usuario.

        A partir de la investigación se decidio realizar la formula donde
intervienen diferentes
        aspectos como Los sensores del vehiculo, el historico de
reputación que posee este, entre otros.
```

```
        Parametros:
        Carro --> Cual fue el carro seleccionado por el usuario
        Probabilidad --> Cual fue la probabilidad de que un objeto tenga
una certeza del 80% o mayor
        Persona --> La persona la cual decidio realizar un viaje en un
determinado carro
        Facilidad --> Cual fue la facilidad que encontro la persona en el
vehiculo al realizar el viaje
        Disfrute --> Cual fue el disfrute que encontro la persona en el
vehiculo al realizar el viaje

        Variables importantes:
        Percepcioninicial --> Es el número de sensores que posee el
vehiculo
        """
        PercepcionInicial = 0
        CarroWE234Lleva = 0
        CarroRTY456Lleva = 0
        CarroZX1922Lleva = 0
        if Carro == 'rty456':
            onto.RTY456.Lleva.append(Persona)
            for prop in onto.RTY456.get_properties():
                for value in prop[onto.RTY456]:
                    if value is True:
                        PercepcionInicial = PercepcionInicial +1
                    if prop.python_name == "Lleva":
                        CarroRTY456Lleva = CarroRTY456Lleva + 1
            onto.RTY456.Percepción_inicial.clear()
            if CarroRTY456Lleva ==0:
                CarroRTY456Lleva =1
            onto.RTY456.Percepción_inicial.append(PercepcionInicial*10)
            HistoricoReputacion = onto.RTY456.Historico_reputacion[0]
            Reputacion = onto.RTY456.Reputacion_Carro[0]
            onto.RTY456.Reputacion_Carro.clear()
            Reputacion = ((((PercepcionInicial*10)+
round(Probabilidad))/2) * 0.20) +
round(HistoricoReputacion/CarroRTY456Lleva) * 0.50 +
(((Facilidad)+(Disfrute))/2) * 0.30
            onto.RTY456.Reputacion_Carro.append(round(Reputacion))
            onto.RTY456.Historico_reputacion.clear()
            print("La reputacion es ", Reputacion)
            HistoricoReputacion =  Reputacion + HistoricoReputacion

onto.RTY456.Historico_reputacion.append(round(HistoricoReputacion))
            print("La nueva reputación del carro RTY456 es :",Reputacion)

        if Carro == 'we234':
            onto.WE234.Lleva.append(Persona)
```

```
            for prop in onto.WE234.get_properties():
                for value in prop[onto.WE234]:
                    if value is True:
                        PercepcionInicial = PercepcionInicial +1
                    if prop.python_name == "Lleva":
                        CarroWE234Lleva = CarroWE234Lleva + 1
            onto.WE234.Percepción_inicial.clear()
            if CarroWE234Lleva ==0:
                CarroWE234Lleva =1
            onto.WE234.Percepción_inicial.append(PercepcionInicial*10)
            HistoricoReputacion = onto.WE234.Historico_reputacion[0]
            Reputacion = onto.WE234.Reputacion_Carro[0]
            onto.WE234.Reputacion_Carro.clear()
            Reputacion = ((((PercepcionInicial*10)+
round(Probabilidad))/2) * 0.20) +
round(HistoricoReputacion/CarroWE234Lleva) * 0.50 +
(((Facilidad)+(Disfrute))/2) * 0.30
            onto.WE234.Reputacion_Carro.append(round(Reputacion))
            onto.WE234.Historico_reputacion.clear()
            HistoricoReputacion =  Reputacion + HistoricoReputacion

onto.WE234.Historico_reputacion.append(round(HistoricoReputacion))
            print("La nueva reputación del carro WE234 es :",Reputacion)

        if Carro == 'zx1922':
            onto.ZX1922.Lleva.append(Persona)
            for prop in onto.ZX1922.get_properties():
                for value in prop[onto.ZX1922]:
                    if value is True:
                        PercepcionInicial = PercepcionInicial +1
                    if prop.python_name == "Lleva":
                        CarroZX1922Lleva = CarroZX1922Lleva + 1
            onto.ZX1922.Percepción_inicial.clear()
            if CarroZX1922Lleva ==0:
                CarroZX1922Lleva =1
            onto.ZX1922.Percepción_inicial.append(PercepcionInicial*10)
            HistoricoReputacion = onto.ZX1922.Historico_reputacion[0]
            print("Historico de reputación viejo :",HistoricoReputacion)
            Reputacion = onto.ZX1922.Reputacion_Carro[0]
            onto.ZX1922.Reputacion_Carro.clear()
            Reputacion = ((((PercepcionInicial*10)+
round(Probabilidad))/2) * 0.20) +
round(HistoricoReputacion/CarroZX1922Lleva) * 0.50 +
(((Facilidad)+(Disfrute))/2) * 0.30
            onto.ZX1922.Reputacion_Carro.append(round(Reputacion))
            onto.ZX1922.Historico_reputacion.clear()
            HistoricoReputacion =  Reputacion + HistoricoReputacion
            print("Historico de reputación nuevo :",HistoricoReputacion)
```

```
onto.ZX1922.Historico_reputacion.append(round(HistoricoReputacion))
            print("La nueva reputación del carro ZX1922 es :",Reputacion)

    def SeleccionCarro(self):
        """El usuario selecciona el vehiculo el cual usara para el viaje
        """
        for i in list(onto.individuals()):
            if str(i) == "Carro.WE234":
                pcomenwe234 = '\n'.join([str(i) for i in
onto.WE234.comment])
                ComentariosWE234.set(pcomenwe234)
                for prop in onto.WE234.get_properties():
                    for value in prop[onto.WE234]:
                        if prop.python_name == "Inicia":
                            valorinicio = str(value)
                            Inicio.set(valorinicio.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Termina":
                            valorfinal = str(value)
                            FinalWE234.set(valorfinal.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Reputacion_Carro":
                            valorreputacion = int(value)
                            ReputacionWE234.set(valorreputacion)
                        elif prop.python_name== "Marca_Carro":
                            valorMarca = str(value)
                            MarcaWE234.set(valorMarca)
                print("")
            if str(i) == "Carro.RTY456":
                pcomenrty456 = '\n'.join([str(i) for i in
onto.RTY456.comment])
                ComentariosRTY456.set(pcomenrty456)
                for prop in onto.RTY456.get_properties():
                    for value in prop[onto.RTY456]:
                        if prop.python_name == "Inicia":
                            valorinicio = str(value)
                            Inicio.set(valorinicio.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Termina":
                            valorfinal = str(value)
                            FinalRTY456.set(valorfinal.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Reputacion_Carro":
                            valorreputacion = int(value)
                            ReputacionRTY456.set(valorreputacion)
                        elif prop.python_name== "Marca_Carro":
                            valorMarca = str(value)
```

```
                            MarcaRTY456.set(valorMarca)
                print("")
            if str(i) == "Carro.ZX1922":
                pcomenzx1922 = '\n'.join([str(i) for i in
onto.ZX1922.comment])
                ComentariosZX1922.set(pcomenzx1922)
                for prop in onto.ZX1922.get_properties():
                    for value in prop[onto.ZX1922]:
                        if prop.python_name == "Inicia":
                            valorinicio = str(value)
                            Inicio.set(valorinicio.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Termina":
                            valorfinal = str(value)
                            FinalZX1922.set(valorfinal.replace(Carro,
reemplazar_por))
                        elif prop.python_name== "Reputacion_Carro":
                            valorreputacion = int(value)
                            ReputacionZX1922.set(valorreputacion)
                        elif prop.python_name== "Marca_Carro":
                            valorMarca = str(value)
                            MarcaZX1922.set(valorMarca)


d = Detecciones()
onty = Ontologia()
per = Persona ()
car = CarroAutonomo()
onty.Inferencias()
car.SeleccionCarro()

raiz.title("Modelado de confianza en los vehiculos autonomos")
fuente = font.Font(weight='bold')
raiz.config(bd=15)
raiz.resizable(True, True)
raiz.grid_columnconfigure((0,1), weight=1)

""" Incluimos el panel de pestañas"""
nb = ttk.Notebook(raiz)
nb.pack(fill='both', expand='yes')

p1 = ttk.Frame(nb)
p1.pack()
p2 = ttk.Frame(nb)

nb.add(p1,text='Viaje')
nb.add(p2,text='Inferencias')
```

```
"""Campos pertenecientes al vehiculo WE234 """
ttk.Label(p1, text="\nComentarios del vehiculo WE234").grid(row=0,
column=0)

scrolledtext1=st.ScrolledText(p1, width=60, height=1)
scrolledtext1.grid(column=1,row=0 ,padx=10, pady=5)
scrolledtext1.insert("1.0", ComentariosWE234.get())
scrolledtext1.configure(state ='disabled')

ttk.Label(p1, text="\nMarca de Carro").grid(row=1, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=MarcaWE234,
width=25).grid(row=1, column=1)

ttk.Label(p1, text="\nInicio del viaje").grid(row=2, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=Inicio,
width=25).grid(row=2, column=1)

ttk.Label(p1, text="\nFinal del viaje").grid(row=3, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=FinalWE234,
width=25).grid(row=3, column=1)

ttk.Label(p1, text="\nReputación actual").grid(row=4, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED,
textvariable=ReputacionWE234, width=25).grid(row=4, column=1)

Label(p1)

"""Campos pertenecientes al vehiculo RTY456 """
ttk.Label(p1, text="\nComentarios del vehiculo RTY456").grid(row=5,
column=0)

scrolledtext2=st.ScrolledText(p1, width=60, height=1)
scrolledtext2.grid(column=1,row=5 ,padx=10, pady=5)
scrolledtext2.insert("1.0", ComentariosRTY456.get())
scrolledtext2.configure(state ='disabled')

ttk.Label(p1, text="\nMarca de Carro").grid(row=6, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=MarcaRTY456,
width=25).grid(row=6, column=1)

ttk.Label(p1, text="\nInicio del viaje").grid(row=7, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=Inicio,
width=25).grid(row=7, column=1)
```

```
ttk.Label(p1, text="\nFinal del viaje").grid(row=8, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=FinalRTY456,
width=25).grid(row=8, column=1)

ttk.Label(p1, text="\nReputación actual").grid(row=9, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED,
textvariable=ReputacionRTY456, width=25).grid(row=9, column=1)

Label(p1)

"""Campos pertenecientes al vehiculo ZX1922 """
ttk.Label(p1, text="\nComentarios del vehiculo ZX1922").grid(row=10,
column=0)

scrolledtext3=st.ScrolledText(p1, width=60, height=1)
scrolledtext3.grid(column=1,row=10 ,padx=10, pady=5)
scrolledtext3.insert("1.0", ComentariosZX1922.get())
scrolledtext3.configure(state ='disabled')

ttk.Label(p1, text="\nMarca de Carro").grid(row=11, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=MarcaZX1922,
width=25).grid(row=11, column=1)

ttk.Label(p1, text="\nInicio del viaje").grid(row=12, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=Inicio,
width=25).grid(row=12, column=1)

ttk.Label(p1, text="\nFinal del viaje").grid(row=13, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED, textvariable=FinalZX1922,
width=25).grid(row=13, column=1)

ttk.Label(p1, text="\nReputación actual").grid(row=14, column=0)
ttk.Entry(p1, justify=CENTER, state=DISABLED,
textvariable=ReputacionZX1922, width=25).grid(row=14, column=1)

ttk.Separator(p1, orient='horizontal')

ttk.Label(p1, text="Seleccione un carro").grid(row=15, column=0)


CampoSeleccionVehiculo = ttk.Entry(p1, justify=CENTER,
textvariable=SelCarro, width=25)

CampoSeleccionVehiculo.grid(row=15, column=1)
ttk.Button(p1, text="Seleccionar", command=per.NuevoUsuario).grid(row=15,
column=2)

""" Campos pertenecientes a la pestaña inferencias """
```

```
ttk.Label(p2, text="\nPromedio de edad de los usuarios").pack()
ttk.Entry(p2, justify=CENTER, state=DISABLED,
textvariable=resultadopromedio).pack()

ttk.Label(p2, text="\nGenero que asigno más veces una valoracion mayor a
60 en el atributo de Facilidad").pack()
ttk.Entry(p2, justify=CENTER, state=DISABLED,
textvariable=resultadofacilidad).pack()

ttk.Label(p2, text="\nGenero que asigno más veces una valoracion mayor a
60 en el atributo de Disfrute").pack()
ttk.Entry(p2, justify=CENTER, state=DISABLED,
textvariable=resultadodisfrute).pack()

ttk.Label(p2, text="\nUsuario el cual posee más edad").pack()
ttk.Entry(p2, justify=CENTER, state=DISABLED,
textvariable=MayorEdad).pack()

raiz.style = ttkthemes.ThemedStyle()
raiz.style.theme_use('breeze')
raiz.mainloop()
```
Fuente: Autor del proyecto.

Anexo  B. Vista detallada de la ontología modelada.