UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE UN SISTEMA DE TASACIÓN Y PREDICCIÓN DE PRECIOS DE AUTOS USADOS EN CHILE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

GABRIEL ANDRÉS AZÓCAR CÁRCAMO

PROFESOR GUÍA:
NELSON BALOIAN TATARYAN

MIEMBROS DE LA COMISIÓN:
JOSE PINO URTUBIA
PABLO GONZALEZ JURE

SANTIAGO DE CHILE
2020

Resumen

La venta de vehículos es en Chile uno de los mercados más sólidos y extensos, tanto en el comercio de autos nuevos como en el de autos usados. En el último de estos existe la necesidad, como en toda venta de artículos de segunda mano, de hacer una tasación del vehículo para poder comprarlo o venderlo. Sin embargo, ésta no es directa y es efectuada comúnmente por personas conocedoras del mercado y con experiencia en el rubro.

Dada esta problemática, en este trabajo se buscó una forma de poder automatizar el proceso, utilizando ejemplos del mercado para poder predecir cómo el precio de un vehículo se comporta a través de los años, de manera que un usuario pueda obtener una buena estimación del vehículo que quiere comprar o vender.

Para poder obtener datos del mercado actual, se construyó un *scraper* que recorre diariamente la mayoría de las páginas de ventas de autos usados en Chile. Luego, con el fin de formatear e indexar estos datos para poder usarlos para construir una tasación, se implementó un sistema basado en palabras claves y en la agrupación de las entradas a través de su marca, modelo, año y versión específica. Ésta última es el dato principal de la solución propuesta y está basada en la información pública que el Servicio de Impuestos Internos libera sobre todos los autos existentes en el país. Como un modelo de auto puede tener varias versiones específicas con precios muy diferentes entre ellas, se hizo un análisis minucioso sobre cuáles versiones de un mismo auto se pueden considerar parecidas y cuáles no, a través del análisis de sus precios de lanzamiento al mercado, formando lo que se denominó grupos de versiones. Estos grupos son los importantes al momento de generar la predicción, ya que todos los autos pertenecientes al mismo grupo del vehículo consultado se toman como datos para construir el predictor.

Además, se implementó una aplicación web que permite hacer consultas sobre vehículos específicos entregando el precio actual y el que tendrá a través de los años.

Finalmente, se obtuvieron resultados y validaciones tanto para el predictor como para la aplicación web. El primero se validó comparándolo con datos reales y el segundo se validó a través de encuestas de usabilidad, dando ambos resultados positivos y satisfactorios.

Para mis padres, que me lo han dado todo

Agradecimientos

Se me hace muy difícil el escribir esta sección porque tengo mucha gente a la cual me encantaría poder agradecerle. Para empezar, me gustaría dedicarles unas palabras a quienes nunca han dejado de creer en mi, mis padres Jorge y Claudia. Es imposible no usar el cliché de "no estaría aquí si no fuese por ustedes", porque se los debo todo y más. No hubiese logrado nada sin su apoyo incondicional ni su sabiduría, la cual siempre ha sido oportuna y para mi una de las cosas más importantes al momento de tomar decisiones. Aún así, siempre me han dejado construir mi propio camino hacia la persona que soy actualmente, aún cuando sé que en ocasiones no estaban del todo de acuerdo con mis elecciones. Todo por lo que hemos pasado se puede interpretar como "lo que nos trajo hasta aquí", y es por eso que no me arrepiento de nada y estoy sinceramente agradecido. De igual manera quiero agradecer a mis hermanos Carlos y Nacho. Al primero, gracias por acompañarme en mis años de universidad, siempre dispuesto a ayudar, apoyar y acompañar sea cual sea el motivo. Al segundo, gracias por tu vitalidad y apañe a lo que sea que se me ocurra, ya sea jugar algo o ver anime. También quiero agradecer a la hermana adoptiva Consuelo, quien siempre se preocupó por mi aún no teniendo ninguna obligación.

Quiero agradecer a las organizaciones de las cuales formé parte en mi paso por la universidad. Primero, quiero agradecer al equipo de baloncesto de ingeniería donde conocí muchas personas valiosas con las cuales me entretenía mucho jugar. En especial quisiera mencionar a Julio quien siempre me apoyó y me devolvió las ganas de jugar basket.

Segundo, me gustaría agradecer a todos los integrantes del Laboratorio de Robótica, donde pasé física y mentalmente la mayoría de mi tiempo en la universidad. Conocí gente que me marcó tanto profesional como humanamente y creé muchos lazos de amistad que sé que conservaré por muchos años. Agradecer especialmente a la gente que confió en mí en mi periodo como capitán, su apoyo fue vital en ese entonces para mí. También a José Miguel, quien fue el que creyó en mi para entrar a trabajar siendo mechón y le estoy muy agradecido por eso. Por último al profesor Javier Ruiz-del-Solar, quien es el creador y sostenedor del proyecto.

Quiero agradecer también a todos mis amigos quienes me acompañaron en esta etapa: todos en Rayo Basket, Billie, Carba, Kay, Nico, Kenzo, Dattari, Leiva, Cele, Mattayas, Thibo, Elías, Rayo y Pinchi (si se me queda alguien después me cobran sentimientos no más).
Emilio, Pablo F., Pablo C. y Coni, gracias por estar en las más feas.

Obviamente no puede quedar fuera la gente linda de la familia Startnet, quienes me acogieron en mi primera práctica profesional, luego en la segunda y después me ofrecieron mi primer trabajo del cual nace esta memoria. Muchas gracias por haber confiado en mí, siempre digo que no podría haber "caído" a un mejor lugar (digo caído porque fue de casualidad que leí su anuncio en u-cursos) y se les estima bastante.

Finalmente, agradecer al profesor Nelson por su tiempo y guía.

Tabla de Contenido

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto y problema

El mercado automotriz, tanto de autos nuevos como usados, es uno de los más sólidos que existen en nuestro país, de hecho es un negocio que no para de crecer [12]. En 2019, Chile tenía 5.5 millones de autos en circulación [7], lo que da una proporción de aproximadamente 1 auto cada 4 habitantes. Si bien la cantidad de autos que ingresan al parque automotriz es alta [14], también lo es la rotación de los ya existentes a través del mercado de los autos usados. Como en toda venta de un artículo de segunda mano, existe dificultad en la tasación de los autos, la que actualmente es realizada por personas expertas en el campo ya que depende de varios factores, como la marca, el modelo, el año, el kilometraje, el estado, etc. Esto genera un problema para las personas ajenas a este mercado a la hora de saber cuánto vale un vehículo, ya sea para saber a qué precio venderlo o comprarlo.

Considerando lo descrito anteriormente, en este trabajo se presenta un sistema computacional de tasación de autos usados que tiene como objetivo entregar el valor de mercado actual del auto así como también la evolución esperada de su precio en los años venideros. El sistema permite a los usuarios consultar la tasación de un auto ya sea ingresando la patente del mismo junto a su kilometraje o ingresando la marca, modelo, año y kilometraje. Luego de ingresar los datos, el sistema da a elegir versiones del auto que ingresó, y si los datos que se tienen sobre ese modelo son suficientes, el sistema entrega una tasación actual del auto junto con una curva de tendencia del precio en los años que siguen (suponiendo un uso de 20.000 kilómetros por año). De esta manera, el sistema apoya al usuario a tomar la decisión, por ejemplo, de comprar o vender un auto ahora o en un futuro. El sistema está basado en la extracción automática de datos de las principales páginas web de ventas de autos en Chile.

El proyecto nace en la empresa Startnet, una startup creada en el año 2017 dedicada al company builder de empresas ligadas al campo automotriz en Chile. Su última incursión en el campo es la creación de una plataforma de créditos automotrices, donde es posible solicitar un crédito para comprar un auto o bien utilizar un vehículo propio como prenda para obtener un crédito de consumo. En este último caso, el sistema descrito en este documento se vuelve relevante por la necesidad de tasar de manera eficaz los autos que llegan como solicitudes de préstamo, para poder determinar cuánto dinero es el máximo disponible para éste.

1.2. Objetivos

1.2.1. Objetivo General

Construir un sistema de predicción de precios de autos usados basado en las características de éstos (marca, modelo, año, versión, kilometraje, tipo de combustible, tipo de transmisión, etc), que permita al usuario realizar tasaciones de autos para fines comerciales. Esto se hace principalmente con el objetivo de calcular viabilidades de créditos con autos como prenda y para poder evaluar inversiones a mediano plazo, como compra de autos en especifico según su devaluación anual. El producto resultante es útil para la empresa donde se realiza el trabajo ya que cuenta con una plataforma de solicitudes de préstamos financieros.

1.2.2. Objetivos Específicos

Para lograr el objetivo general expuesto en este trabajo de título, se propone cumplir con los siguientes objetivos específicos:

1. Descargar datos de las principales páginas web de venta de autos usados, tanto concesionarias como particulares, utilizando *web scraping*. De ahora en adelante a este sistema se le denominará scraper. El objetivo de este sistema es obtener los datos necesarios para alimentar el predictor de precios de los autos usados.

2. Contar con datos formateados e indexados a partir de los datos obtenidos en el objetivo anterior. El objetivo de esto es lograr filtrar los datos y normalizarlos para facilitar su procesamiento.

3. Contar con un modelo de predicción utilizando los datos procesados por el objetivo anterior. Este sistema se basará en técnicas de *machine learning*. El objetivo es poder tasar autos usados.

4. Crear una interfaz que permita interactuar con el tasador de autos.

1.3. Estructura del documento

El resto de este documento se estructura como sigue. En el capítulo 2 se presenta y discute la literatura relacionada que sustenta el trabajo realizado en esta memoria de título. El capítulo 3 está dedicado a presentar el diseño de la solución propuesta. A continuación, en el capítulo 4 se describen las decisiones de diseño e ingeniería tomadas para poder implementar la solución descrita en el capítulo anterior. En el capítulo 5 se presentan los resultados de la validación del tasador realizada a través de comparaciones con datos reales y la validación de la interfaz gráfica realizada a través de pruebas de usabilidad con personas externas al proyecto. Finalmente, en el capítulo 6 se presentan las conclusiones y se ofrecen perspectivas de trabajo futuro.

Capítulo 2

Estado del Arte

Dado el ámbito de este trabajo, tenemos los siguientes temas para los cuales se hizo una revisión con el objeto de recopilar información relevante para tomar las decisiones pertinentes a este desarrollo. Primero se comentará sobre sitios web que implementan funciones similares a las que se quieren desarrollar en este trabajo, para revisar el diseño y eventualmente repetir algunas buenas prácticas. Segundo, como este trabajo es principalmente un desarrollo de software, se expondrá la literatura acerca de metodologías pertinentes para escoger la más adecuada. También se hará una revisión de los frameworks (plataformas de apoyo al desarrollo) para decidir cuál se ajusta mejor a las condiciones de este trabajo.

2.1. Trabajos previos en el Tema

En Chile existen dos plataformas que realizan un trabajo similar a lo que se quiere desarrollar:

1. **Véndenos tu Auto** [23]: Esta página ofrece el servicio de comprar autos de particulares. Si bien no se especializa en tasar autos, tienen un tasador de valor actual que ayuda a la empresa a proponer un valor al auto que está ofreciendo el usuario. Sin embargo, este tasador solo muestra un precio actual del vehículo y no una evolución a través del tiempo, lo que no permite al usuario analizar si le conviene vender ahora o a mediano plazo.

2. **Autobook** [3]: Esta página presenta un sistema que es más parecido al que se quiere implementar en este trabajo de título. Genera un informe completo sobre el auto tasado, incluyendo precio esperado, evolución de este y detalles de los diferentes ajustes que este precio podría tener, por ejemplo, ubicación geográfica del auto o estado del mismo. Para ello utilizan información de autos publicados tanto en medios digitales como impresos y actualizan sus informes semanalmente. Para tasar un auto el usuario puede introducir tanto su patente como sus características individuales (marca, modelo, versión y año).

Figura 2.1: Ejemplos de vistas de Véndenos tu Auto (arriba) y Autobook (abajo).

Con esto, se pueden obtener las siguientes conclusiones aplicables al trabajo propuesto:

- Ambos tasadores tienen en común que la información que piden es marca, modelo, versión, año y kilometraje, lo que señala que una tasación necesita al menos esos datos para poder realizarse. Cabe destacar que la versión del auto implícitamente entrega más información, por ejemplo tipo de combustible, tipo de transmisión, tipo de tracción, entre otros.

- En el caso de Autobook, ellos afirman que sus datos los obtienen desde sitios de ventas de autos, por lo que se infiere que si es posible obtener esos datos, también es posible realizar tasaciones.

2.2. Ingeniería de Software

En sus inicios, el desarrollo de software era tarea de una sola persona, que resolvía un problema bien acotado y para uso personal. Esto hacía que el ciclo de la creación del software sea sencillo ya que consistía solo en escribir el código, revisar errores y corregirlos. Sin embargo, con el paso del tiempo, el hardware aumentó sus capacidades y disminuyó sus costos, generando una masificación del uso de computadores lo que a su vez creó nuevos ámbitos en donde un software podía ser solución a un problema. Con esto, se hizo necesario crear estándares para generar y mantener software a gran escala, ya que los proyectos ahora consistían de muchos programadores trabajando simultáneamente en códigos de miles de líneas [11]. Esto dio paso a la creación de la Ingeniería de Software, que se puede resumir en el estudio del proceso de creación de software confiable y de calidad, basándose en métodos y técnicas de ingeniería. Entre las muchas áreas que aborda esta rama de las ciencias de la computación está el estudio de modelos y ciclos de vida para desarrollo de software, llamados también metodologías de desarrollo de software, que se discutirán con más detalle en la sección siguiente.

2.2.1. Metodologías de desarrollo de software

Una metodología de desarrollo de software es un marco de trabajo utilizado para estructurar, planificar y controlar el proceso de desarrollo de software. Las principales razones para trabajar utilizando esta estructura en un proceso son:

- Ayuda a generar un entendimiento común de lo que se quiere hacer.

- Ayuda encontrar inconsistencias, redundancias u omisiones en las partes del proceso.

- Ayuda a marcar hitos o metas del desarrollo.

- Ayuda a modificar el proceso fácilmente si es necesario.

Si bien existe una gran cantidad de metodologías, estas se pueden encasillar principalmente en cuatro modelos o enfoques:

1. **Modelo de Cascada** [19]: Es un modelo que se asemeja a una cascada, empezando con las fases de concepción del proyecto hasta llegar al termino de éste. Se caracteriza por ser estricto en el sentido del flujo del trabajo, ya que una fase debe ser terminada

para poder pasar a la siguiente. La principal crítica a este modelo es su rigidez que muchas veces hace que los proyectos tarden más de lo necesario.

2. **Modelo de Espiral** [4]: Es un modelo de tipo iterativo, que se caracteriza por agregar el análisis de riesgo al proyecto. Pasa por etapas sucesivas de planificación, evaluación, análisis de riesgo e ingeniería, para luego volver a repetir mientras se considere necesario. Si bien es un poco más flexible que el Modelo de Cascada, también tiene desventajas en el sentido de que el análisis de riesgo puede demorar el proyecto.

3. **Modelo de Desarrollo en Fases** [2]: Es un modelo basado en lanzamientos periódicos del software. Existen dos tipos:

   - Incremental: El sistema se especifica y luego es dividido en subsistemas asociados a una funcionalidad. Primero se comienza con un subsistema pequeño y en cada *release* se van agregando más funciones.

   - Iterativo: Se entrega el sistema completo y se van haciendo cambios y mejoras en la funcionalidad en cada *release*.

   Sus principales desventajas son que el proceso puede perder visibilidad y que agregar una nueva funcionalidad puede "romper" una ya existente.

4. **Modelo de Desarrollo Ágil o Metodologías Ágiles** [1]: Es un modelo de Desarrollo en fases flexible donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. Sus principales características son ciclos iterativos breves, diseño simple, organización libre, procesos adaptables y reacción rápida ante los cambios. Estas características hacen que el desarrollo del software sea más expedito, por eso se le denomina "ágil". Sus principales desventajas son las mismas del Modelo de Desarrollo en Fases.

Como el proyecto que se propone en esta propuesta debe ser realizado en un periodo corto de tiempo, se opta por adoptar Metodologías Ágiles para el desarrollo de este. En la siguiente sección se discutirán los principales marcos de trabajo que utilizan este modelo.

2.2.2. Metodologías Ágiles

Como se introdujo en la sección anterior, este tipo de metodología está pensada principalmente para proyectos que necesitan ser terminados rápidamente. Por lo tanto, es ideal para el desarrollo que se propone en el presente trabajo de título. A continuación se revisarán los tres modelos más utilizados de esta metodología.

1. **Scrum** [1]: Este modelo tiene como característica principal el desarrollo de tareas por parte de grupos pequeños (3 a 9 personas) auto-organizados. El desarrollo es incremental distribuido en fases, llamadas *sprints*. Estos son periodos de desarrollo de tiempo fijo (un mes normalmente) en los cuales se busca desarrollar una funcionalidad específica. A diario se debe realizar una reunión corta para explicar en que está cada grupo y sincronizar tareas para el día. El énfasis del desarrollo está determinado principalmente por los requerimientos del cliente, dando mayor prioridad a lo que este considera

importante. Sus principales beneficios son la flexibilidad al cambio, reducción del *Time to Market*[1], mayor productividad y buena predicción de tiempos.

2. **Kanban** [18]: Este modelo se basa en la definición de tareas específicas, denominadas "tarjetas", ordenadas como una cola de prioridad. De esta forma, los desarrolladores van adquiriendo tareas según la tarjeta que siga en la cola. Ésta suele ser representada por un tablero, donde hay columnas como lista de objetivos o *backlog*, trabajos en proceso, trabajos terminados, etc. Esto hace que el proceso actual sea visible para todo el equipo, lo que ayuda a la transparencia del trabajo. El método se basa en cuatro principios principales: comenzar por lo que se va a hacer ahora, perseguir el cambio incremental y evolutivo, respetar el proceso actual y sus roles y existencia de liderazgo en todos los niveles. Sus principales beneficios son evitar la producción excesiva, lograr tiempos de entrega cortos, flexibilidad en el desarrollo y el fácil control de material a través de la visibilidad de los procesos.

3. **Extreme Programming** [1]: Este modelo es considerado el más adecuado para equipos pequeños que quieran desarrollar rápidamente proyectos a largo plazo. Sus principales características son el desarrollo incremental, la existencia de pruebas unitarias continuas, el uso de programación en parejas y la existencia de roles específicamente asignados. Existen cinco reglas, llamadas valores, que el equipo debe respetar: simplicidad, comunicación, retroalimentación, valentía (saber cuando desechar código, cuando refactorizar, etc.) y respeto (luchar por alta calidad de código, no romper código existente, entre otros). Sus principales beneficios son eficiencia en el proceso de pruebas y planificación, tasa de error pequeña, programación bien organizada y buena coordinación con el cliente.

Si bien las tres opciones antes expuestas sirven para lo que se necesita, el modelo escogido es Scrum porque permite definir tareas especificas en periodos de tiempos bien definidos. Otra ventaja es que en el contexto en el cual se desarrollará el proyecto se utiliza esta metodología.

2.3. Desarrollo web

El desarrollo web es un concepto que hace referencia a la creación de sitios web, ya sea para internet o para una intrared. Normalmente es una comunicación entre un cliente, un servidor y una base de datos con el fin de realizar una acción o consultar información. En este contexto, la arquitectura comúnmente utilizada en la actualidad es conocida como Modelo-Vista-Controlador, la cual es la escogida también para este proyecto. Su principal característica es el de separar los datos de la lógica de negocio[2] y, como su nombre lo indica, consta de tres partes:

- **Modelo**: Representa la información que opera el sistema y gestiona los accesos y actualizaciones a ella. Las peticiones de acceso o manipulación de información llegan a través del Controlador. Esta parte suele asociarse a la base de datos del sistema.

[1]Tiempo que transcurre desde que se concibe un producto hasta que está disponible para la venta.

[2]Parte de un sistema que se encarga de codificar las reglas de negocio del mundo real que determinan cómo la información puede ser creada, almacenada y cambiada.

- **Vista**: Parte gráfica del sistema donde se muestra el estado del Modelo y permite interactuar con él. Al dar una orden de modificación del Modelo, se envía una petición al Controlador. Esta parte suele asociarse al *frontend* del sistema.

- **Controlador**: Recibe y responde a instrucciones enviadas por la vista, normalmente desencadenando peticiones al Modelo para consultar información o para que la información sea modificada. Esta parte suele asociarse al *backend* del sistema.

En la Figura 2.2 se puede observar un ejemplo de interacción entre los componentes.

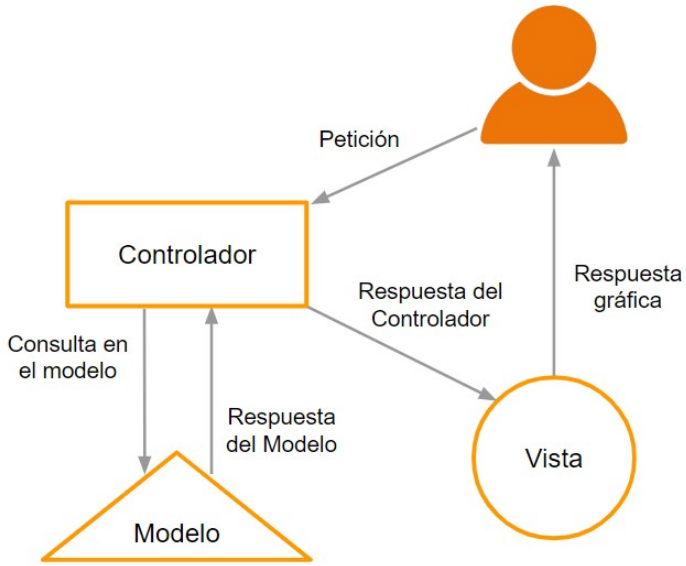

Figura 2.2: Ejemplo de Modelo-Vista-Controlador.

2.3.1. Frontend

En el contexto de desarrollo de web, el *frontend* es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. La interfaz de usuario es el componente principal de esta parte, para la cual existen varios frameworks que ayudan a su creación. En esta sección se presentarán dos de los más utilizados: Angular y React.

- **Angular** [13]: Es un framework de desarrollo web escrito en TypeScript y mantenido por Google. Está enfocado principalmente a la creación de aplicaciones de una sola página. Su funcionamiento se basa en la creación de componentes que son porciones de código que pueden ser reutilizados en todo el proyecto. Sus principales características son un databinding[3] sólido y ser fácilmente testeable. Además, cuenta con una herramienta llamada "Angular CLI" la cual es una interfaz de la línea de comandos que permite inicializar, desarrollar y mantener las aplicaciones Angular directamente desde la terminal.

- **React** [10]: Es un framework de desarrollo de aplicaciones web de una sola página escrito en JavaScript y mantenido por Facebook. Al igual que Angular, se basa en

[3]Técnica general que une y sincroniza las fuentes de datos del proveedor y el consumidor.

el uso de componentes. Su principal característica es la de poder trabajar de buena manera con datos que son modificados constantemente, contando con herramientas que permiten que la página no deba renderizarse completamente de nuevo al ocurrir un cambio en los datos. Para esto cuenta con "Redux", una librería que se encarga de imponer ciertas restricciones sobre cómo y cuándo pueden producirse las actualizaciones para que la consistencia sea posible.

En el caso del proyecto a realizar, ningún framework presenta grandes ventajas por sobre otro, por lo que se escoge Angular, ya que todos los demás proyectos que se realizan en la empresa lo utilizan.

2.3.2. Backend

En el contexto de desarrollo web, el *backend* es la parte que se encarga de la lógica de la plataforma, como también de la conexión a la base de datos y el servidor utilizado, a la cual el usuario no tiene acceso directamente. Existen varios lenguajes y frameworks para el desarrollo de backend de sistemas Web, sin embargo, en la empresa a realizar el proyecto trabajan principalmente con dos: Django y Symfony, por lo tanto en esta sección se mencionará ambas.

- **Django** [9]: Es un framework de desarrollo web de código abierto para el lenguaje Python. Sigue el patrón de diseño de Modelo Vista Controlador. Contiene herramientas para crear y configurar elementos recurrentes en el desarrollo web como formularios, autenticación, endpoints, entre otros. Sus ventajas son entrega un rápido desarrollo, es seguro contra SQL injections o CSRF[4], cuenta con un buen ORM[5] y es muy escalable.

- **Symfony** [20]: Es un framework de desarrollo web de código abierto para el lenguaje PHP. Sigue el patrón de diseño de Modelo Vista Controlador. En su globalidad, tiene las mismas características y ventajas que Django, aunque es preferible que se implemente en grandes proyectos.

Como se aprecia, ambos frameworks ofrecen soluciones con funcionalidades similares, diferenciándose principalmente en el lenguaje en el que se desarrollan los sistemas. Es por esto que Django será el utilizado en este proyecto, ya que Python cuenta con buenas librerías de Machine Learning, con las cuales además se tiene experiencia previa. En adición, existe un proyecto anterior en Django, desarrollado por la empresa, cuyo código se puede reutilizar como base.

2.3.3. Base de datos

Al trabajar con bases de datos, la primera decisión a tomar es si se trabajará con una base de datos de tipo relacional o una de tipo no relacional. A continuación se expondrán ambas.

- **Relacionales**: También llamadas SQL, ya que la principal interacción con ellas es a

[4]Es un tipo de exploit malicioso de un sitio web en el que comandos no autorizados son transmitidos por un usuario en el cual el sitio web confía.

[5]Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la utilización de una base de datos relacional.

través del Lenguaje de Consultas Estructuradas (SQL). En este tipo de bases de datos la información está organizada en un conjunto de tablas formalmente descritas, las cuales se identifican con nombres e índices. Sus principales ventajas son la robustez, la integridad referencial y la fácil comprensión de los datos al estar ordenados en tablas. Sus principales desventajas son posibles problemas al cambiar la estructura, tiempos de respuesta cuando crecen mucho y problemas al almacenar datos multimedia o datos de información geográfica. Algunos ejemplos son MySQL, PostgresSQL y MariaDB.

- **No Relacionales**: También llamadas no SQL, ya que no soportan la interacción con ese lenguaje. En este tipo de bases de datos la información está organizada a través de documentos sin identificadores específicos, lo cual es muy útil cuando no se tiene de antemano la estructura de los datos a almacenar. Sus principales ventajas son la versatilidad, el crecimiento horizontal y baja necesidad de recursos computacionales para funcionar. Las principales desventajas son la poca documentación del software, la no existencia de un estándar en el lenguaje y la no existencia de garantías de atomicidad en la información. Algunos ejemplos son MongoDB, Cassandra y Redis.

Como los datos que se van a utilizar son estructurados y se necesita integridad referencial, se decide utilizar una base de datos relacionales.

2.4. Regresión

Como este trabajo tratará de buscar una línea de tendencia a través de ejemplos, se introducirá el concepto de regresión. En el contexto de la estadística, una regresión es un proceso para estimar relaciones entre variables, por ejemplo, obtener una curva de tendencia a partir de un grupo de puntos. Si bien existen muchos modelos para ese fin, como se trabajará con la librería de Python Sklearn [8], se mencionarán solo los dos principales métodos que tiene implementados.

- **Método de Mínimos cuadrados**: Este método intenta minimizar la suma de cuadrados de las diferencias (llamadas residuos) de valores en el eje de las ordenadas entre los puntos generados por la función elegida y los correspondientes valores en los datos iniciales.

- **Regularización de Tikhonov o Regresión de arista (Ridge)**: Este método es un ajuste al método de mínimos cuadrados e intenta resolver:

$$\text{mín}_w \, ||Xw - y||^2 + \alpha||w||^2$$

  Donde $||x||$ es distancia euclidiana y α un ponderador arbitrario mayor a 0. Se puede observar que si α es igual a 0, se parece bastante al método de mínimos cuadrados.

Como la gran mayoría de problemas de machine learning, la mejor solución o el mejor método no pueden ser decididos puramente por teoría, ya que los resultados empíricos son los más importantes y no siempre coinciden con ésta. Es por esto que los dos métodos antes mencionados serán utilizados como solución y comparados uno con el otro para poder tomar la decisión final.

Capítulo 3

Solución propuesta

La solución propuesta contempla cuatro etapas: la extracción de datos utilizando *web scraping*[1], el formateo e indexación de los datos obtenidos, la construcción de un modelo de predicción utilizando técnicas de *machine learning* y la interfaz gráfica como frontend web. Dentro de estas etapas, es la segunda la que presenta mayor relevancia, ya que es esencial que los datos estén bien indexados para lograr una buena predicción. No basta con poder identificar la marca, modelo y año de un auto ya que existen muchos casos donde la versión del mismo es la que determina el valor de este. Por ejemplo, la camioneta Mitsubishi L200[2] en su edición 2018 presentó 9 versiones, en donde la diferencia de precio entre la más barata y la más cara es de casi 6 millones de pesos. Esto es un desafío ya que muchas veces las páginas de venta de autos contienen las versiones de los autos de manera distinta o simplemente no la tienen.

Para comenzar con la descripción de la solución propuesta, se presentará la arquitectura escogida, que consta de cuatro elementos: scraper, backend, base de datos y frontend. Como se aprecia en la Figura 3.1, la interacción entre estos cuatro elementos se puede dividir en dos grandes etapas: extracción, procesamiento y almacenamiento de los datos (flechas naranjas) y consulta y predicción (flechas azules). Cada una de estas etapas se subdividen en los siguientes pasos.

Extracción, procesamiento y almacenamiento de los datos:

1. Paso A: El scraper obtiene datos de publicaciones y los envía a un endpoint del backend.

2. Paso B: El backend recibe los datos del scraper, los procesa (formatea e indexa) y los guarda en la base de datos.

[1]Técnica para extraer información de sitios web mediante programas de software. Usualmente, estos programas simulan la navegación de un humano en una página determinada.

[2]https://mitsubishi-motors.cl/nuevaL200

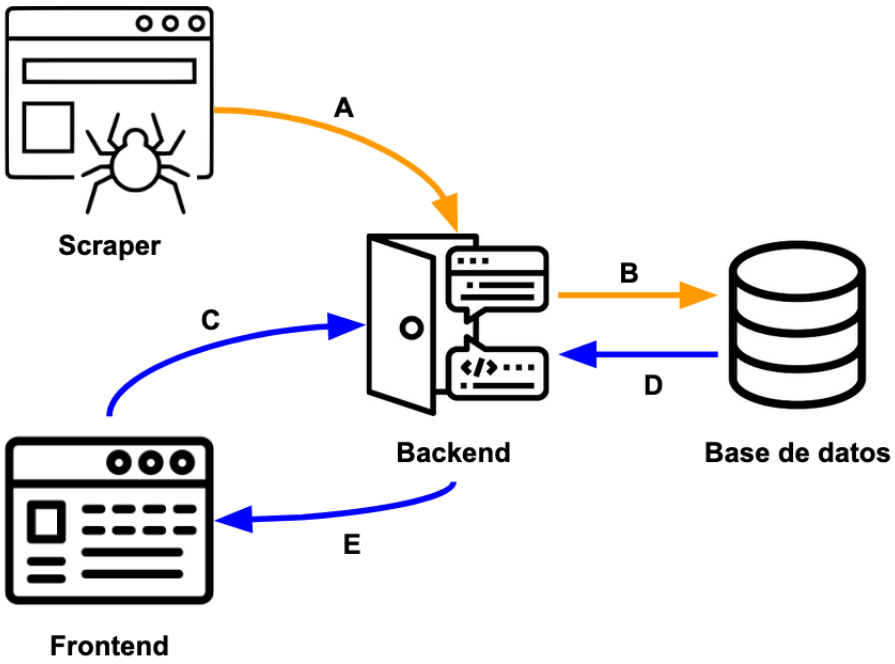

Figura 3.1: Arquitectura propuesta.

Consulta y predicción:

1. Paso C: El usuario consulta sobre un auto. El frontend envía los datos a consultar a un endpoint de backend.

2. Paso D: El backend consulta la base de datos para obtener las entradas asociadas al auto consultado. Genera la predicción y envía la respuesta al frontend.

3. Paso E: El frontend recibe respuesta del backend y muestra la información en pantalla.

Dado esto, se puede relacionar a los elementos con los objetivos específicos de la sección anterior.

- El objetivo número 1 tiene relación con el elemento scraper.
- Los objetivos número 2 y 3 tienen relación con los elementos backend y base de datos.
- El objetivo número 4 tiene relación con el elemento frontend.

A continuación se detallarán las soluciones propuestas para cada punto de los objetivos específicos.

3.1. Scraper

Este subsistema será el encargado de revisar periódicamente los sitios web nacionales de ventas de autos usados. En Chile existen más de cien páginas del rubro, algunas dedicadas a la publicación de autos particulares y otras dedicadas a publicación exclusiva de automotoras. Todas las páginas tienen un patrón en común en sus interfaces: existe una vista con el listado de todos los autos disponibles y existe una vista donde se muestran los detalles de un auto en específico (Figuras 3.2 y 3.3). Por lo tanto se pueden identificar dos pasos en este proceso, primero recopilar información sobre los autos disponibles y segundo revisar la información especifica de cada uno de los autos encontrados, como se ilustra en la Figura 3.4. Luego, la solución propuesta para este objetivo se describe de la siguiente forma.

Scraper de la vista de listado (flechas naranjas en Figura 3.4):

1. Paso A: Desde una cola que contiene urls de páginas de listado, el scraper consume los elementos uno por uno.

2. Paso B: Visita la página de listado de la url actual y extrae las urls de la página de detalle de cada auto.

3. Paso C: Envía las urls encontradas a una nueva cola que contiene urls de detalles de autos.

Scraper de la vista de detalle (flechas azules en Figura 3.4):

1. Paso D: Desde la cola que contiene urls de detalles de autos, el scraper consume los elementos uno por uno.

2. Paso E: Visita la página de la url actual y extrae los detalles específicos de aquel auto.

3. Paso F: Envía los detalles del auto al backend para su procesamiento.

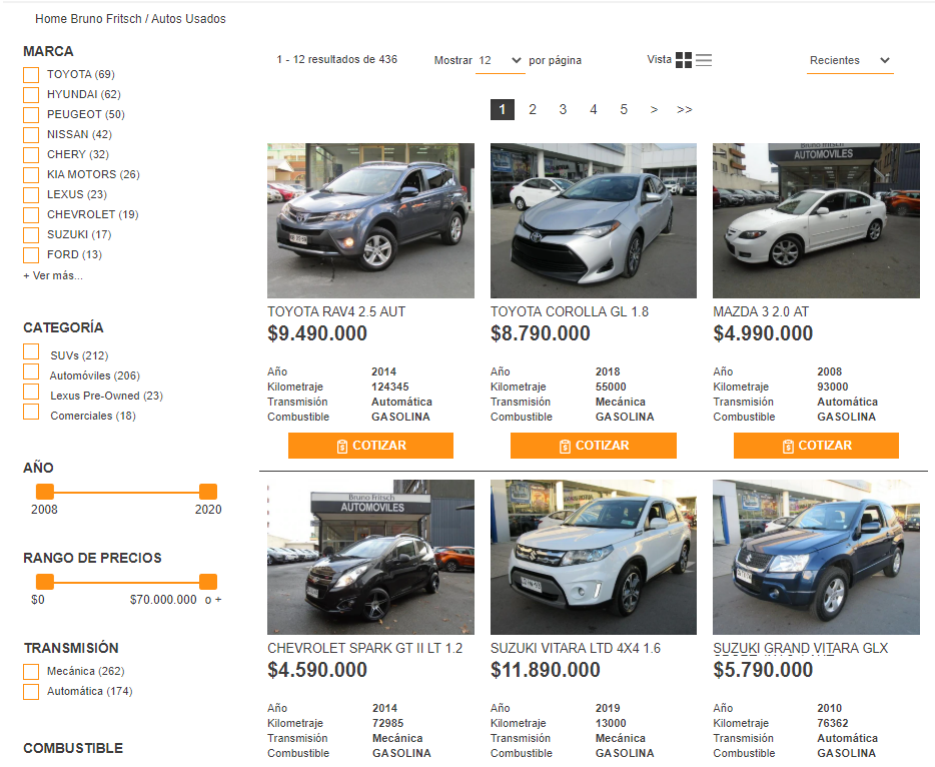

Figura 3.2: Ejemplo de vista de listado de autos.

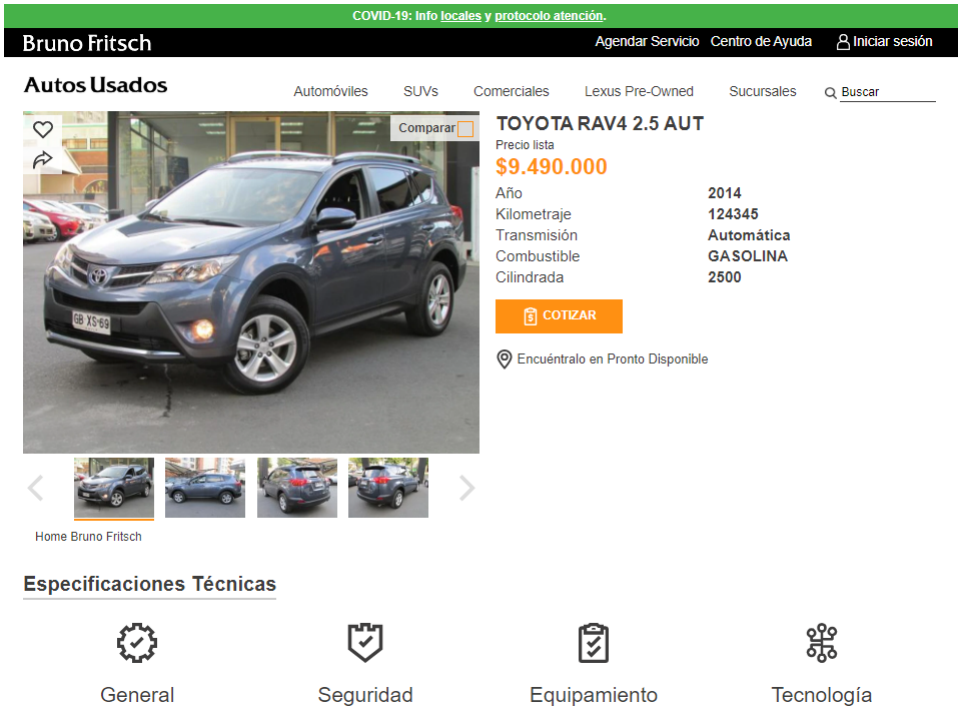

Figura 3.3: Ejemplo de vista de detalle de auto.

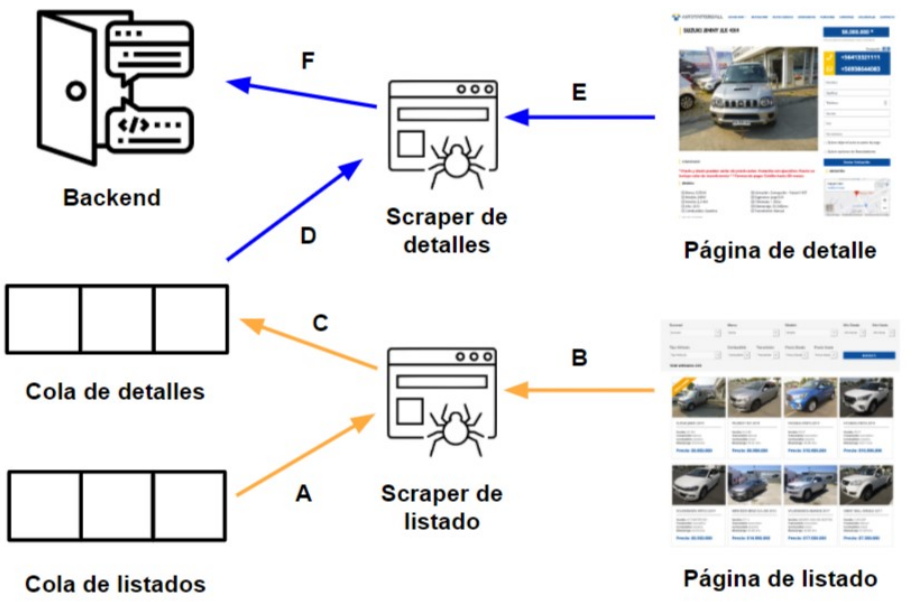

Figura 3.4: Arquitectura propuesta para el scraper.

3.2. Formateo e indexación

Este subsistema estará contenido en el backend y será el encargado de hacer que la información proveniente del scraper sea registrada de forma correcta. Para esto, se necesitarán tres tablas de datos: marcas de vehículos existentes en Chile, modelos de esas marcas y versiones de esos modelos por año. Esta información es de carácter pública y es distribuida por el Servicio de Impuestos Internos [22]. Con esta información es posible encontrar las versiones y las palabras claves que se utilizan en ellas (por ejemplo D/C o DC para Doble Cabina). Como fue explicado anteriormente, no basta con identificar marca, modelo y año de un auto para poder identificarlo, también es necesario obtener la versión específica de este. Sin embargo, hay versiones que si bien son diferentes, se pueden considerar como iguales por la similitud de sus precios. A estos grupos parecidos se les llamará Grupo de Versiones. Esto será explicado con mayor detalle más adelante.

De esta forma, cada vez que llega una entrada nueva desde el scraper se le podrá asignar un Grupo de Versión según su marca, modelo, año y versión, para hacer que su identificación posterior sea más sencilla. Este proceso de dar formato a la información no es directo, ya que nada asegura que estos tres atributos lleguen de forma separada desde el scraper, ya que muchas veces en las páginas de ventas de autos existe un solo campo llamado "modelo" que es realmente una concatenación del modelo y la versión. Además, este subsistema se encargará de uniformar los valores de otros campos, por ejemplo, que todos los sinónimos de "bencina" obtengan el mismo valor, que ocurra lo mismo con los sinónimos de "diésel", igualmente con los sinónimos de "4x4", etc. Finalmente, deberá agregar la entrada con la información filtrada a la base de datos.

3.3. Predicción

Para esta parte del trabajo, se propone realizar una regresión polinomial en varias dimensiones, utilizando como datos los autos obtenidos por el scraper que tengan relación con el auto cuyo precio se quiere predecir. Esta relación hace referencia a otras versiones del modelo que son similares y será determinada por el parecido de sus precios en el momento en que salieron al mercado (Grupo de Versiones mencionado anteriormente). Además, se agregarán los autos relacionados a la versión pero lanzados al mercado en años anteriores o posteriores, ya que tienden a comportarse de forma similar, solo que con desfase temporal. Estos datos se situarán en un espacio de varias dimensiones, las cuales deben ser al menos las siguientes:

1. Kilometraje.
2. Tipo de combustible.
3. Tipo de transmisión.
4. Tipo de tracción.
5. Año de lanzamiento.
6. Año de publicación de la entrada de venta (en la base de datos se tienen datos del scraper del 2017, 2019 y 2020).

3.4. Interfaz gráfica

La interfaz gráfica propuesta para este trabajo consiste en dos vistas simples, una donde el usuario pueda ingresar los datos del vehículo a evaluar y la otra donde se puedan visualizar los datos de la tasación. En la primera, habrán dos opciones: buscar por marca, modelo, versión, año y kilometraje o buscar por patente y kilometraje. La información sobre las patentes emitidas en Chile es de carácter público y es distribuida por el Servicio de Impuestos Internos [21]. En la segunda, se mostrará el valor actual del auto junto con un gráfico que indique como su precio irá variando a través de los años, suponiendo un aumento fijo de kilometraje por año. En las figuras 3.5 y 3.6 se pueden apreciar bocetos de lo explicado anteriormente.

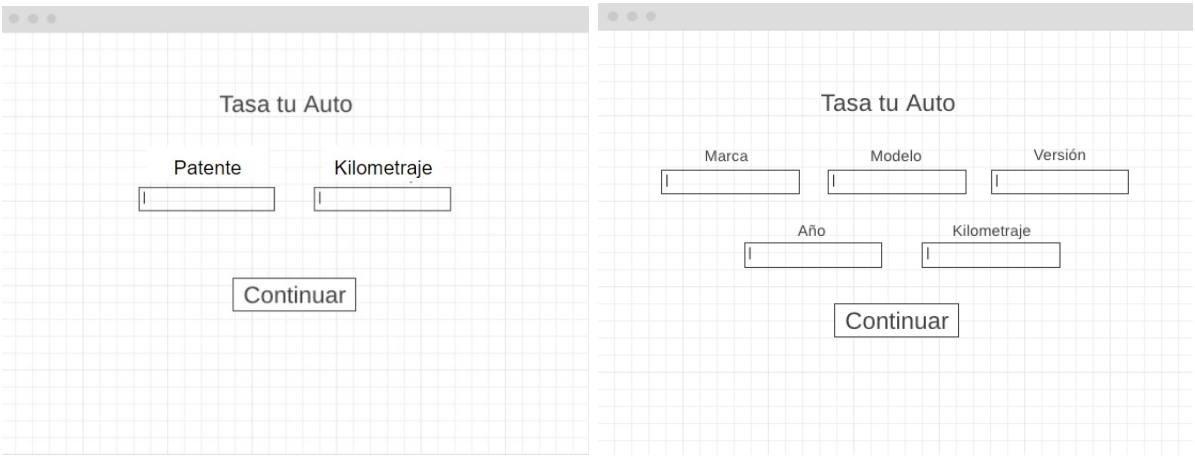

Figura 3.5: Boceto vista consultar vehículo.

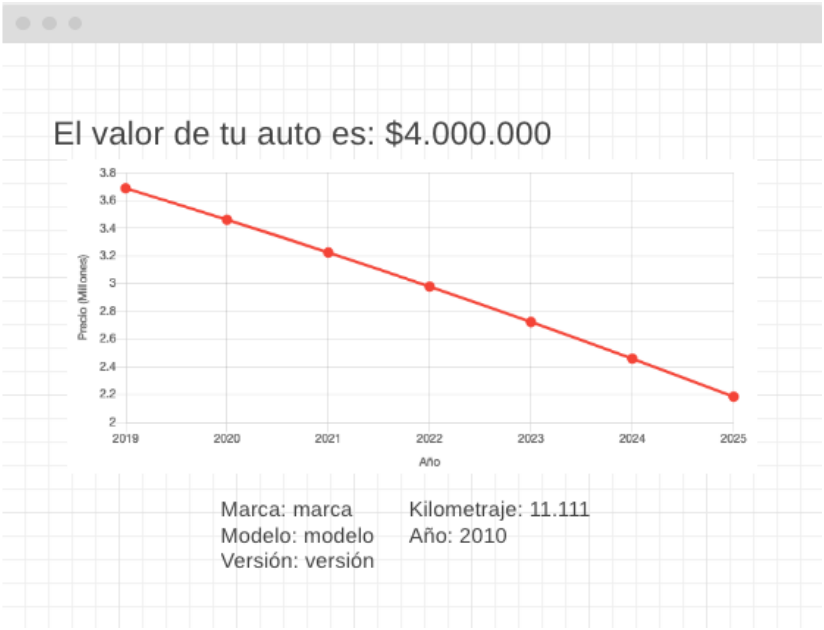

Figura 3.6: Boceto vista resultados.

Capítulo 4

Implementación

4.1. Scraper

Al momento de empezar este proyecto existía un esqueleto muy básico de este subsistema escrito en el lenguaje de programación Javascript que hacía scraping de 4 páginas, el cual fue mejorado y modificado casi en su totalidad por el autor de esta memoria. Para las colas de mensajes utiliza un *message broker*[1] llamado RabbitMQ [17] y para la parte de *web scraping* utiliza la librería Cheerio [6]. El trabajo realizado en el scraper fue principalmente modificación del flujo, extensión de alcance (actualmente hace *web scraping* de 75 páginas) y arreglo de errores. En esta sección se detallará el trabajo recién mencionado.

El scraping de páginas web es un proceso con poca capacidad de generalización, ya que cada página web es distinta y se deben hacer diferentes cosas para obtener la información relevante de cada una de ellas. Es por esto que es necesario tener por cada página a visitar un scraper dedicado a ella. Sin embargo, como fue mencionado en la sección 3.1, la mayoría de las páginas de ventas de autos siguen el patrón de tener dos vistas principales: una con la lista de autos y otra para el detalle de un auto en particular. Por lo tanto deben existir dos funciones de scraping por página, las cuales serán llamadas *procesarUrl* y *procesarDetalles* respectivamente. Finalmente, es necesario tener un archivo de configuración que indique ciertas características de la página a hacer scraping, por ejemplo, la url de su página de listado y los parámetros que utiliza. Este archivo es de tipo JSON y contiene la siguiente información:

- siteName: nombre del sitio.
- initialPage: número de su página inicial de listado.
- limitPage: número de páginas de listado a la que se quiere llegar.
- url: dirección de la página de listado.
- qs: queryset de la página de listado.

[1] Programa intermediario que traduce los mensajes de un sistema a otro.

- pageVariable: identificador que recibe el número de página actual en la url de listado.
- coef: este valor es el coeficiente de cuánto se le suma al pageVariable para cambiar de página. Por ejemplo, Yapo[2] cuenta las páginas de uno en uno, pero Salazar Israel[3] cuenta las páginas por offset de autos a mostrar. Esto quiere decir que la primera página muestra los primeros 20 autos, la segunda muestra los autos desde el 20 hasta el 40, la tercera del 40 al 60 y así sucesivamente, por lo que su pageVariable es 20.

Para ilustrar mejor el uso de este archivo de configuración, se tiene el siguiente ejemplo de la configuración de página Yapo.cl:

```
{
    "siteName": "yapo",
    "initialPage": 0,
    "limitPage": 500,
    "url": "https://www.yapo.cl/chile/autos",
    "qs": {
        "st": "s",
        "rs": 2006,
        "cg": 2020
    },
   "pageVariable": "o"
   "coef": 1
}
```

Esto significa que se va a hacer scraping de la página 0 a la 500 de la url ahí indicada, utilizando la variable "o" como indicador de página y los parámetros listados en "qs", que en este caso son que solo sean publicaciones de venta ("s" de *sell*) y que los autos mostrados estén entre los años 2006 y 2020. Más concretamente, este archivo va a generar las siguientes urls al ser leído:

```
https://www.yapo.cl/chile/autos?st=s&rs=2006&cg=2020&o=0
https://www.yapo.cl/chile/autos?st=s&rs=2006&cg=2020&o=1
https://www.yapo.cl/chile/autos?st=s&rs=2006&cg=2020&o=2
...
https://www.yapo.cl/chile/autos?st=s&rs=2006&cg=2020&o=500
```

Esto corresponde a todas las urls de listado de autos que se desea hacer scraping del sitio Yapo.cl. El módulo encargado de leer los archivos de configuración es llamado "productor" y será explicado a continuación.

El scraper se divide en tres módulos principales independientes: el productor, el consumidor de listas y el consumidor de detalles. Basados en la Figura 3.4, el primero es el encargado de poblar la cola de listados con urls de páginas de listados de autos, el segundo es el "scraper de listado" en la figura y el tercero es el "scraper de detalles" en la figura. A continuación se detallará su funcionamiento.

[2]https://www.yapo.cl/chile/autos/
[3]https://www.salazarisraelusados.cl/autos-usados/

4.1.1. Productor

Este módulo es el encargado de poblar la cola de listado con las urls que se encuentran especificadas en los archivos de configuración de las páginas, de la forma explicada anteriormente. Se gatilla cada 8 horas para páginas de automotoras y cada 24 para páginas de publicación de particulares. Si bien esta diferencia no tiene mayor relevancia para el trabajo propuesto en este documento, si lo tiene para otra plataforma de la empresa que necesita que los autos de páginas de automotoras estén lo más actualizados posible. Los mensajes producidos por el productor son de tipo JSON y contienen dos campos: el nombre del sitio de origen y la url respectiva a revisar. Por ejemplo:

```
{
  "origin": "gtautos",
  "url": "https://gtautos.cl/web/autos-usados?page=2",
}
```

4.1.2. Consumidor de listas

Este módulo es el encargado de poblar la cola de detalles, lo cual realiza consumiendo la cola de listados. Una vez que recibe un mensaje, identifica la página de origen a través del campo *origin* y utiliza la función *processarUrl* asociada a esa página para hacer scraping del listado de autos.

Hay dos posibles escenarios para una url de listado de autos. El primero es que el *endpoint* donde la página hace la consulta para obtener sus autos es conocido. Si este es el caso, no es necesario hacer scraping de la página del detalle del auto, ya que solo hace falta consultar el *endpoint* para obtener los detalles de los autos. A este tipo de páginas se les llamará de scraping rápido. El segundo caso es el contrario, por lo que es necesario visitar las páginas de detalles de cada auto listado para obtener su información. A este tipo de páginas se les llamará de scraping normal. Por lo tanto, los mensajes producidos por este módulo pueden ser de dos tipos:

1. Mensaje de scraping rápido: Este tipo de mensaje ya contiene todo lo necesario para que el backend registre la entrada en la base de datos. Es un JSON estructurado con los campos que se muestran a continuación.

    - origin: nombre del sitio web.
    - url: url.
    - name: id de la entrada.
    - brand: marca.
    - model: modelo.
    - version: versión.
    - year: año.

- cylinder_capacity: cilindrada.
- city : ciudad.
- address: dirección de venta.
- price : precio.
- kilometers : kilometraje.
- description: descripción de la entrada.
- date_created: fecha de publicación.
- transmission: tipo de transmisión.
- gas_type: tipo de combustible.
- url_principal_img: url imagen principal.
- doors: cantidad de puertas.
- color: color del auto.
- extra_image: urls imágenes extra.

2. Mensaje de scraping normal: Este tipo de mensaje contiene lo necesario para poder ingresar a una página de detalle de auto para poder realizarle el scraping y contiene la siguiente información.
   - origin: nombre del sitio web.
   - url: página de url de detalles de la entrada.
   - name: id de la entrada.

Estos mensajes, tanto de scraping rápido como normal, son encolados en la lista de detalles para su procesamiento por parte del Consumidor de detalles.

4.1.3. Consumidor de detalles

Este módulo es el encargado de enviar entradas al backend, a través de consumir la cola de detalles. Una vez que recibe un mensaje, identifica la página de origen a través del campo origin y utiliza la función *procesarDetalles* asociada a esa página para hacer scraping de la entrada. Su comportamiento varia según si el mensaje que recibe es uno de scraping rápido o uno de scraping normal. En el primer caso, solo toma el mensaje y lo reenvía al backend para que sea almacenado en la base de datos. En el segundo caso, visita la url del auto a revisar y extrae la información necesaria, creando un JSON igual al de un mensaje de scraping rápido. Finalmente lo envía al backend para su almacenamiento.

4.2. Formateo e indexación

En esta sección se describe como el backend realiza el procesamiento posterior de los datos entregados por el scraper. El backend está escrito en Python utilizando Django[9].

4.2.1. Modelos fiscales

Todos lo años el Servicio de Impuestos Internos publica una lista con los modelos de autos existentes en Chile y su tasación fiscal [22], lo cual es útil para conocer dos cosas: cuáles son exactamente los autos que circulan en el país y cuáles son las distintas versiones existentes de un mismo auto, lo que se usa para determinar cuales se podrían considerar similares. Por ejemplo, en la Figura 4.1 se ilustran las versiones de la camioneta Mitsubishi L-200 del año 2019, junto con sus nombres y valores de tasación de ese mismo año. Como se puede observar, a pesar de ser el mismo modelo de vehículo, la diferencia entre el valor entre el más barato (versión WORK CR) y el más caro (versión DAKAR HP) es de casi 7 millones de pesos, lo que sugiere que diferentes versiones deben ser tratadas de diferente forma a pesar de ser el mismo modelo.

Sin embargo, también existen versiones que tienen diferencias mínimas de precio entre ellas, lo que sugiere que pueden ser consideradas como parecidas y por ende como parte de un mismo grupo, es decir pertenecerán al mismo Grupo de Versiones. Estos grupos son definidos por el parecido en el precio que tuvieron esas versiones al momento de ser lanzadas al mercado y son nombrados por la unión de las palabras claves de cada versión que conforma el grupo, lo cual se verá con mayor detenimiento más adelante. En la Figura 4.2 se muestran los grupos de versiones de la Figura 4.1 representados en colores.

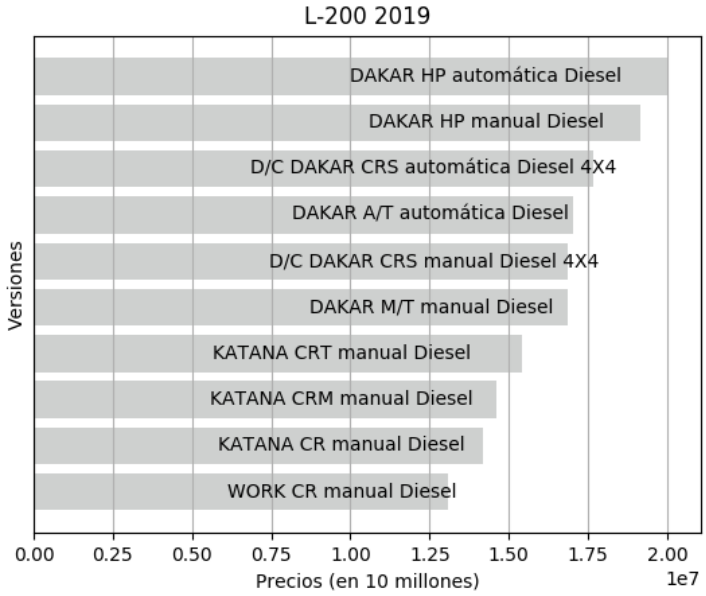

Figura 4.1: Versiones de la camioneta Mitsubishi L-200 del 2019.

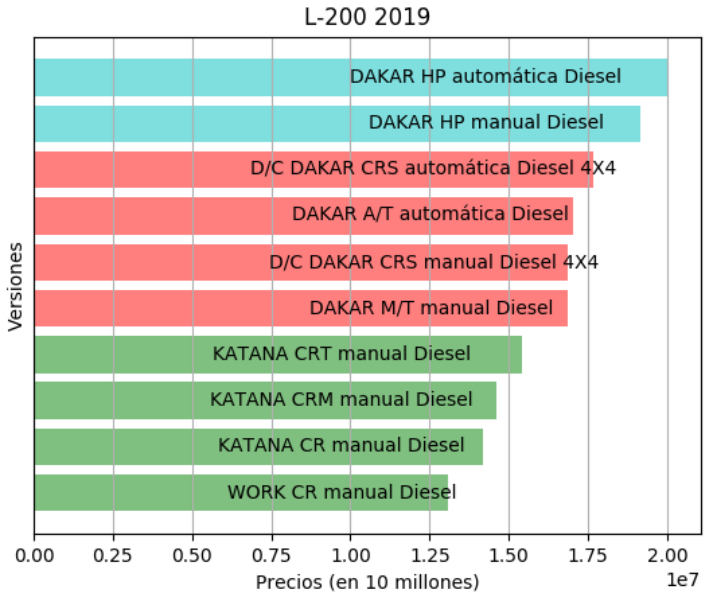

Figura 4.2: Grupos de versiones de la camioneta Mitsubishi L-200 del 2019.

El proceso de búsqueda de un grupo de versiones es realizado automáticamente a través de funciones creadas específicamente y es el siguiente:

1. Se ordena por precio ascendente la lista de versiones a agrupar. Luego, se obtiene la versión con menor precio y se genera un grupo de versiones donde esta es la única integrante. Utilizando como ejemplo el caso de la Figura 4.1, se crea un grupo de versiones de un solo integrante con la versión "WORK CR manual Diesel". Con esto creado, definimos un valor que llamaremos *precio umbral* que será útil para decidir si las versiones restantes pertenecen o no a este grupo. Este umbral es calculado como la mediana de los precios de las versiones que componen el grupo multiplicado por un ponderador, específicamente por 1.1, el cual se obtuvo de forma empírica. Siguiendo con el ejemplo, como hay una sola versión por ahora, el precio umbral se calcula como su precio por el ponderador, por lo tanto:

$$\text{Precio umbral} = \$13.080.000 * 1.1 = \$14.388.000.$$

2. Para decidir si la siguiente versión en la lista ordenada por precios pertenece al grupo que se está formando actualmente, se compara el precio de la versión con el precio umbral del grupo. Si el precio es menor al umbral, entonces la versión pertenece al grupo y se recalcula el precio umbral. Si ocurre lo contrario, el grupo se da por cerrado y se empieza uno nuevo. En el ejemplo, la versión que sigue es "KATANA CR manual Diesel" con un precio de \$14.170.000, lo que la hace perteneciente al grupo según el precio umbral calculado en el punto anterior. Se realiza el ajuste al precio umbral ahora con la nueva mediana, por lo tanto:

$$\text{precio umbral} = 1.1 * (\$13.080.000 + \$14.170.000) \ / \ 2 = \$14.987.500$$

3. Se repite el paso anterior hasta que la versión revisada no cumpla con el precio umbral. En el ejemplo esto ocurre al revisar la versión "DAKAR M/T manual Diesel" que tiene un precio de $16.840.000, ya que en ese punto el precio umbral es:

   Mediana de [$13.080.000, $14.170.000, $14.620.000, $15.430.000] = $14.395.000
   Luego, Precio umbral = 1.1 * $14.395.000 = $15.834.500

   Finalmente, como el precio umbral es menor al precio de la versión, el grupo se cierra (y queda formado como el grupo verde de la Figura 4.2) y la versión "DAKAR M/T manual Diesel" inicia un grupo nuevo con ella como única integrante (que desemboca en el grupo rojo de la Figura 4.2).

Como se mencionó anteriormente, un grupo de versiones es identificado por un nombre que se obtiene de la concatenación de las palabras de cada versión por separado, las que llamaremos palabras clave. En este conjunto pueden encontrarse dos tipos de palabras claves: las que aparecen en todas las versiones del grupo, las cuales serán llamadas "palabras fuertes" y las que no aparecen en todas las versiones, las cuales serán llamadas "palabras débiles". Las primeras se representan en mayúsculas en el nombre del grupo y las segundas en minúsculas. En el caso de que una palabra fuerte esté presente en más de un grupo, solo se mantiene como fuerte en el grupo que tenga menor cantidad de palabras fuertes y en los demás grupos se transforma en una débil. Esto sigue la siguiente lógica: si una palabra es fuerte en varios grupos, entonces no es característica en los grupos que tienen más palabras fuertes. Por ejemplo, al darles nombre a los grupos celeste y rojo de la Figura 4.2, estos serían "DAKAR HP" y "DAKAR d/c crs a/t m/t", siendo "DAKAR" palabra fuerte en ambos. Sin embargo, el primer grupo tiene otra palabra fuerte lo que sugiere que, si bien "DAKAR" es importante, lo es más "HP" porque es exclusiva del grupo. Luego, los grupos quedan nombrados "dakar HP" y "DAKAR d/c crs a/t m/t". Todo este proceso es realizado automáticamente al momento de crear los Grupos.

Como la mayoría de los autos tienen un nuevo lanzamiento año a año, también tienen nuevas versiones. Por esto, se debe analizar cuáles grupos de versiones antiguos corresponden a los actuales. Esto es importante ya que los autos tienden a mantenerse en sus mismos segmentos de mercado y hacer correctamente esta conexión entrega información relevante en cuanto a la depreciación de los mismos. En la Figura 4.3 se muestran las versiones de la camioneta Mitsubishi L-200 lanzadas en el 2018 y sus precios de ese año. Como se puede observar, hay tres grupos al igual que en las versiones 2019, pero hay 9 versiones distintas y no 10. Se puede apreciar claramente que los grupos de ambos años están relacionados, prácticamente son los mismos grupos sólo que con años de lanzamiento diferente. Un conjunto de Grupos de Versiones relacionados recibirá el nombre de *Clan de Grupos de Versiones*. Por ejemplo los grupos "dakar HP 2018" y "dakar HP 2019" forman parte de un mismo Clan (grupos azules en Figuras 4.2 y 4.3). En la Figura 4.4 se muestra un ejemplo de Clan de Grupos de Versiones, donde cada grupo está representado por un color diferente.

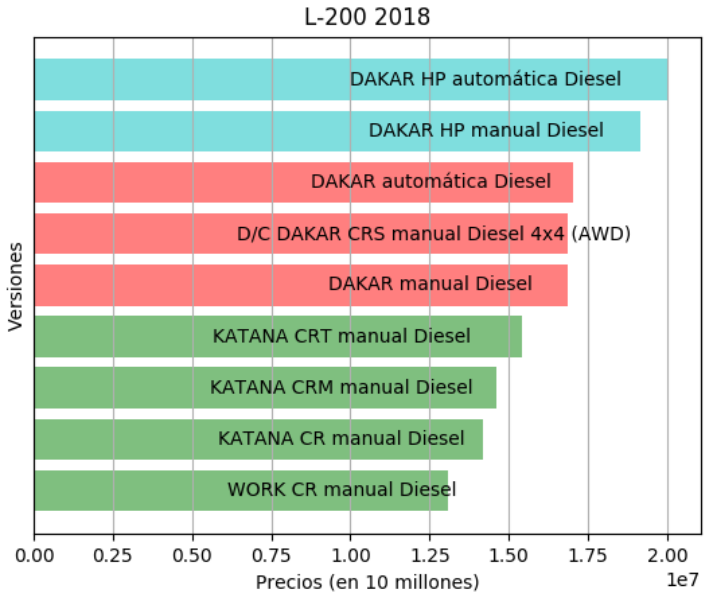

Figura 4.3: Grupos de versiones de la camioneta Mitsubishi L-200 del 2018.

Los Clanes se determinan por el parecido en el nombre de los grupos, donde se calcula un puntaje de semejanza a través del número de palabras que comparten normalizado por el total de palabras sin repetición disponibles. De esta forma, si los nombres son exactamente iguales, el puntaje será de 1, por otra parte si no existe intersección, el puntaje será de 0. Este puntaje se calcula entre todos los grupos de un auto en un año con los grupos del mismo auto en el año anterior. Si el vehículo no tiene versiones en el año anterior, se calculan con las versiones del año más cercano existente. Dos grupos pertenecen al mismo Clan si el puntaje calculado entre ellos es el mayor entre todos los puntajes calculados.

A modo de ejemplo, el grupo Mitubishi L-200 2019 "DAKAR d/c crs a/t m/t" (rojo en la Figura 4.2) puede formar Clan con los grupos Mitsubishi L-200 2018 "dakar HP", "DAKAR d/c crs" o "katana work crt crm cr" (azul, rojo y verde en Figura 4.3 respectivamente). Con la fórmula anteriormente descrita, los puntajes de semejanza serían 1/6, 3/5 y 0/10 respectivamente, por lo que "DAKAR d/c crs a/t m/t" y "DAKAR d/c crs" pertenecen al mismo Clan. Al igual que los Grupos, todo el procedimiento de búsqueda de clanes es realizado automáticamente a través de la definición de una serie de funciones.

Cabe mencionar que las cerca de 35.000 entradas de versiones extraídas desde el Servicio de Impuestos Internos [22] fueron revisadas a mano para poder unificar términos comunes. Por ejemplo, el término "Cabina Simple" puede ser encontrado como "S/C", "SC", "C/S", "CS" y "CABINA SIMPLE". Por lo tanto, se decidió dejar solo "S/C" como término válido, por lo que cada ocurrencia de las otras opciones fueron cambiadas por "S/C". Esta unificación es vital para el funcionamiento del sistema de clanes de grupos de versiones, ya que es necesario que los términos que definen a los grupos calcen completamente a través de los años para el cálculo correcto del puntaje de semejanza. Se denominaron a estos términos *palabras de adaptación*, y existen de dos tipos: los que tienen una marca asociada y los que no. Se puede encontrar la lista completa de las palabras de adaptación definidas en el Apéndice.

Además, se listaron palabras que si bien están en las versiones, no forman parte como tal de ellas ya que se refieren más a una característica que a una palabra clave. Por ejemplo palabras como “diésel”, “automático”, “4x4”, “mecánico”, “petróleo”, entre muchas otras. Estas palabras se bautizaron como *sinónimos* y se puede encontrar la lista completa en el Apéndice.

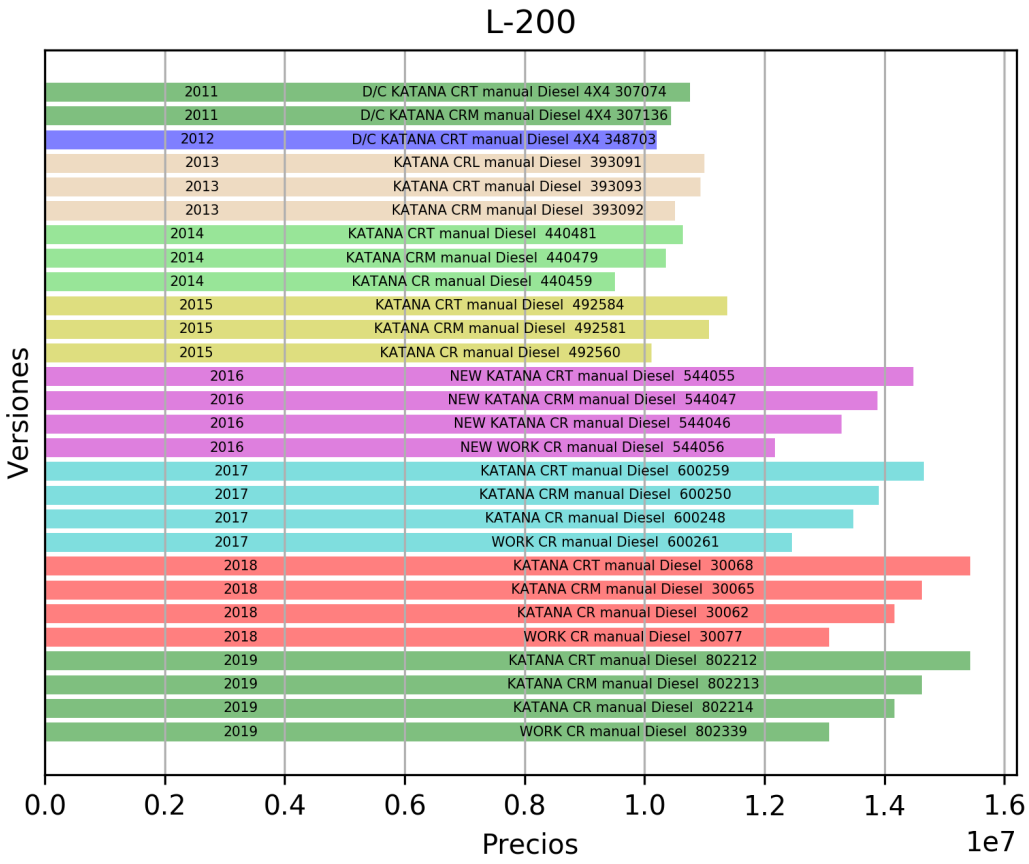

Figura 4.4: Ejemplo de clan de grupos de versiones. Diferentes colores indican diferentes años.

4.2.2. Formateo

En ocasiones, para poder indexar las entradas provenientes del scraper se necesita primero formatear los datos, ya que estos pueden venir desorganizados o con datos errados. Por ejemplo, veamos la siguiente entrada:

Marca	Modelo	Versión	Año
Jeep	Grand Cherokee Limited 4x4 3.6 AT		2017

Transmisión	Combustible	Tracción	Cilindrada
Automática	Bencina		

Como se puede observar, hay columnas que están vacías como versión, tracción y cilindrada. Por lo tanto, antes de poder asignarle un modelo fiscal a esta entrada, es necesario tener las columnas con la información correcta, que en este caso sería:

Marca	Modelo	Versión	Año
Jeep	Grand Cherokee	Limited	2017

Transmisión	Combustible	Tracción	Cilindrada
Automática	Bencina	4x4	3.6

El principio para lograr lo anterior es el revisar las frases claves de la unión entre marca, modelo y versión de una entrada. En el ejemplo anterior, la unión sería "Jeep Grand Cherokee Limited 4x4 3.6 AT" y las frases importantes serían "Jeep", "Grand Cherokee", "Limited", "4x4", "3.6" y "AT". Para obtener las frases se realizan los pasos que se explican a continuación.

1. Concatenación de marca, modelo y versión, convertidos a mayúsculas. Utilizando el ejemplo antes mencionado:
   JEEP GRAND CHEROKEE LIMITED 4x4 3.6 AT

2. Adaptación de palabras claves neutras. Con neutro se hace referencia a que es una palabra de adaptación que no tiene asociada una marca o modelo en específico. Dando más ejemplos, algunos casos son las palabras "Grand" o "New", a las cuales se les cambia el espacio que les sigue por un guión lo que las hace "pegarse" a la palabra que las sigue. Otros casos son las marcas "Mercedes Benz" y "Land Rover", que cambian a "Mercedes-Benz" y "Land-Rover". En el ejemplo, este paso genera:
   JEEP GRAND-CHEROKEE LIMITED 4x4 3.6 AT

3. Se hace una división por espacios de la oración, para tener las frases importantes separadas en forma de arreglo.
   [JEEP, GRAND-CHEROKEE, LIMITED, 4x4, 3.6, AT]

4. Entre todas las frases obtenidas se hace una búsqueda de la marca del vehículo. Esto se hace comparando las marcas existentes en la base de datos con los elementos del

arreglo. La comparación se hace a través de la distancia de Levenshtein[4] y se escoge la marca que tuvo menor distancia, siempre y cuando esta distancia sea menor a 2, de lo contrario, se da por infructuosa la búsqueda. En el ejemplo, se encuentra que la marca es Jeep.

5. Para buscar el modelo, las frases restantes se vuelven a concatenar por espacios. Luego, se hace la adaptación de palabras claves según la marca antes escogida, por ejemplo si la marca es Toyota se adaptarán solo las palabras que sean especiales para Toyota, como "RAV 4" a "RAV-4" o "4 RUNNER" a "4-RUNNER". Una vez hecha la adaptación, se vuelven a dividir las frases en un arreglo según los espacios y se busca el modelo entre los modelos de la marca existentes en la base de datos, utilizando el mismo principio del paso anterior. En el ejemplo, el modelo encontrado será Grand Cherokee.

6. Como ya se conoce la marca y modelo del auto, las frases restantes en el arreglo pueden corresponder a parte de la versión o a características del auto (transmisión, cilindrada, combustible, etc). Como se mencionó previamente, se cuenta con una base de datos de este tipo de palabras para poder realizar la adaptación.
   Primero se buscan palabras claves de combustible (petroleo, gasolina, diésel, etc), luego palabras claves de tracción (4WD, 2WD, AWD, 4x2, 4x4, etc) y finalmente palabras claves de transmisión (AT, MT, mecánico, etc). Si se identifican algunas de estas palabras claves, se registran en el campo correspondiente de la base de datos (si este ya tiene datos, lo sobrescribe) y se eliminan del arreglo de frases. En el ejemplo, se observa que se obtiene tracción 4x4 y transmisión automática (AT). Luego, las frases claves restantes son:

   [LIMITED, 3.6]

7. Si alguna de las frases restantes en el arreglo tiene forma de expresión regular "X.Y" con X e Y números, se entiende como la cilindrada por lo que se registra y elimina del arreglo. En este caso, el valor 3.6.

8. Todas las frases que hayan quedado en el arreglo luego de los pasos anteriores, se entienden como la versión. Por lo tanto se concatenan con espacios y se marca como la versión del auto. En el ejemplo, la versión quedaría solo como "LIMITED", dando el resultado expuesto al comienzo de la sección.

[4]Algoritmo de comparación entre dos strings que entrega la cantidad de operaciones de distancia que hay entre uno y otro. Las operaciones pueden ser reemplazo, adición o eliminación de un carácter.

4.2.3. Indexación

Luego de tener la entrada correctamente formateada, se procede a indexarla. Esto significa encontrar el Grupo de Versiones al cual pertenece, para así poder utilizar el Clan de Versiones asociado en la predicción. Dicho de otra manera, se le asigna un Grupo de Versiones a una entrada para saber que otras versiones de distintos años de ese modelo son compatibles con él, pudiendo así utilizar todas las entradas que pertenecen al Clan en la construcción del predictor.

El proceso es simple, primero se obtienen los grupos del año al cual corresponde la entrada, grupos del año anterior y también del año siguiente. Por ejemplo si el vehículo es del año 2016, se buscan grupos de los años 2015, 2016 y 2017. Luego, se asigna un puntaje para cada grupo calculado a través de comparar las apariciones de las palabras de la versión con las palabras que conforman los nombres de los grupos. Cada aparición da un puntaje que se suma a un puntaje global, si hay un calce con una palabra débil se adiciona 0.3 mientras que si hay un calce con una palabra fuerte se adiciona 1.0. Además, si el grupo es del mismo año que el vehículo, se agrega 1.0 extra. Si el valor del puntaje global excede el umbral de 1.2, ese grupo es candidato a ser el grupo de la entrada. Como se puede observar, si el grupo es del mismo año del vehículo basta con que calce una palabra para poder ser candidato. Finalmente entre todos los grupos candidatos se escoge el que tenga mayor puntaje. En el caso de que exista solo un grupo en el año del vehículo, ese grupo es escogido sin importar si existen o no calces en las palabras claves.

4.2.4. Mantenimiento a través del tiempo de Grupos y Clanes

Como ya ha sido mencionado anteriormente, todos los años se agregan nuevos modelos de autos al mercado, los cuales pueden ser vehículos completamente nuevos o una nueva versión de uno ya existente. Esto sugiere que una vez al año se debe realizar mantenimiento a los Clanes definidos el año anterior. El mantenimiento no es necesario para los Grupos ya que estos por definición fueron formados a través de los precios que las versiones tuvieron al momento de ser lanzadas al mercado, lo que no cambia a través del tiempo. Por lo tanto el proceso se limita a crear nuevos Grupos con los nuevos vehículos y agregarlos a un Clan existente o generar uno nuevo en su defecto. Como el proceso de generación de Grupos y Clanes es automatizado siguiendo el algoritmo descrito en 4.2.1, lo único que se debe hacer manualmente es el revisar si en las nuevas entradas existen nuevos modelos o nuevas palabras claves y agregarlas a las tablas que contienen esa información. Como al inicio de este trabajo de título esas tablas no existían, se tuvieron que construir a partir de revisar cerca de 35.000 entradas, sin embargo las entradas nuevas cada año no superan las 2.500, lo cual es sustancialmente menor. Por lo tanto, gracias al modelo implementado, actualizar la información a través de los años es un trabajo menor, relativamente rápido y directo, ya que basta agregar las nuevas palabras claves encontradas a la base de datos y luego correr una función a la cual solo se le indica el año que se desea analizar. Ésta se encarga de generar los nuevos Grupos y agregarlos a sus respectivos Clanes de manera automática.

4.3. Predicción

Para realizar la predicción se necesitan dos datos: la versión fiscal del modelo a evaluar y su kilometraje actual. Lo primero que se realiza es buscar todas las entradas existentes de vehículos que pertenezcan al mismo Clan de la versión entregada. Luego, todas las entradas son transformadas a vectores, de la forma:

1. Año de publicación del vehículo (año en que fue ingresada la entrada a la base de datos).
2. Año de salida al mercado del vehículo.
3. Kilometraje.
4. Combustible (0 si es bencina, 1 si es diésel).
5. Transmisión (0 si es manual, 1 si es automático).
6. Tracción (0 si es 4x2, 1 si es 4x4).
7. Precio.

No se agregaron otras variables como la cilindrada o la cantidad de puertas, ya que esta información no existe en la mayoría de las entradas (porque en las páginas de venta no están disponibles). Con los vectores creados, se genera una regresión para luego evaluar esa función en el punto que se desea saber, obteniendo así el valor actual del auto. Para obtener los valores futuros del auto, se generan consultas del mismo auto pero sumando años al "Año de publicación del vehículo" y sumando 20.000 kilómetros más al kilometraje por año añadido. Con estos datos se puede generar una linea de tendencia del auto consultado. Todo esto es realizado en Python utilizando la librería Sklearn[8].

En la Figura 4.5 se muestra un ejemplo de predicción, utilizando como método de regresión Ridge, para un Jeep Grand Cherokee Limited 2015, bencina y automático con 90.000 kilómetros y en la Figura 4.6 se muestra un ejemplo de predicción, utilizando el mismo método, para un Chevrolet Sail NB 2013, bencina y manual con 110.000 kilómetros. En el primer ejemplo fueron utilizadas 720 entradas y en el segundo ejemplo fueron utilizadas 671 entradas, extraídas de los clanes de grupos de versiones correspondientes.

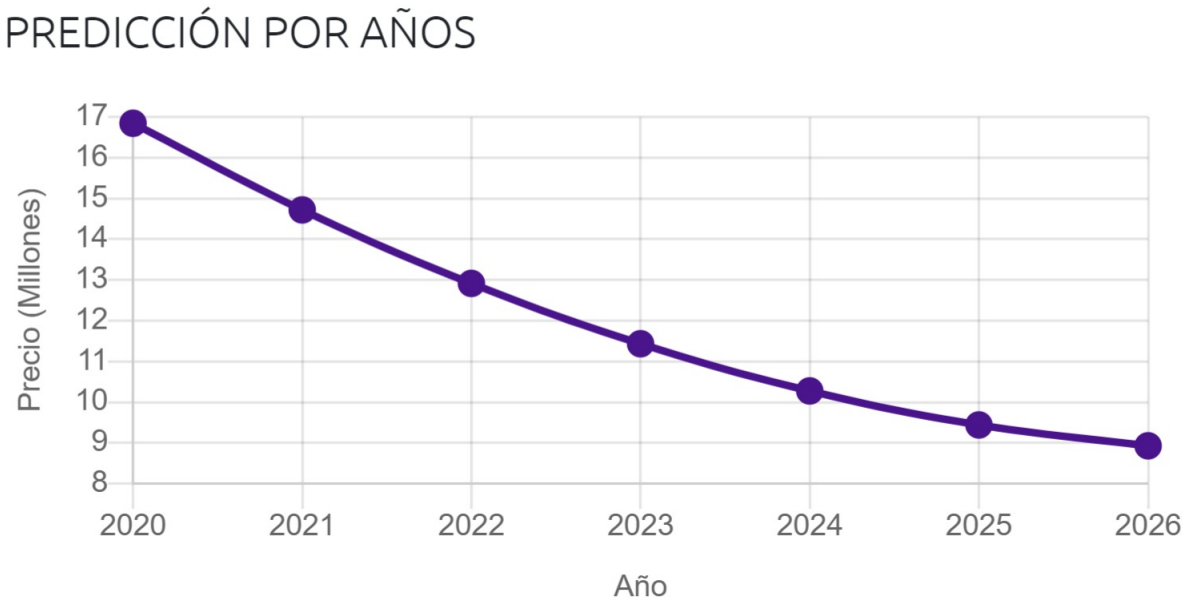

Figura 4.5: Ejemplo de predicción para un Jeep Grand Cherokee Limited 2015, bencina y automático con 90.000 kilómetros.

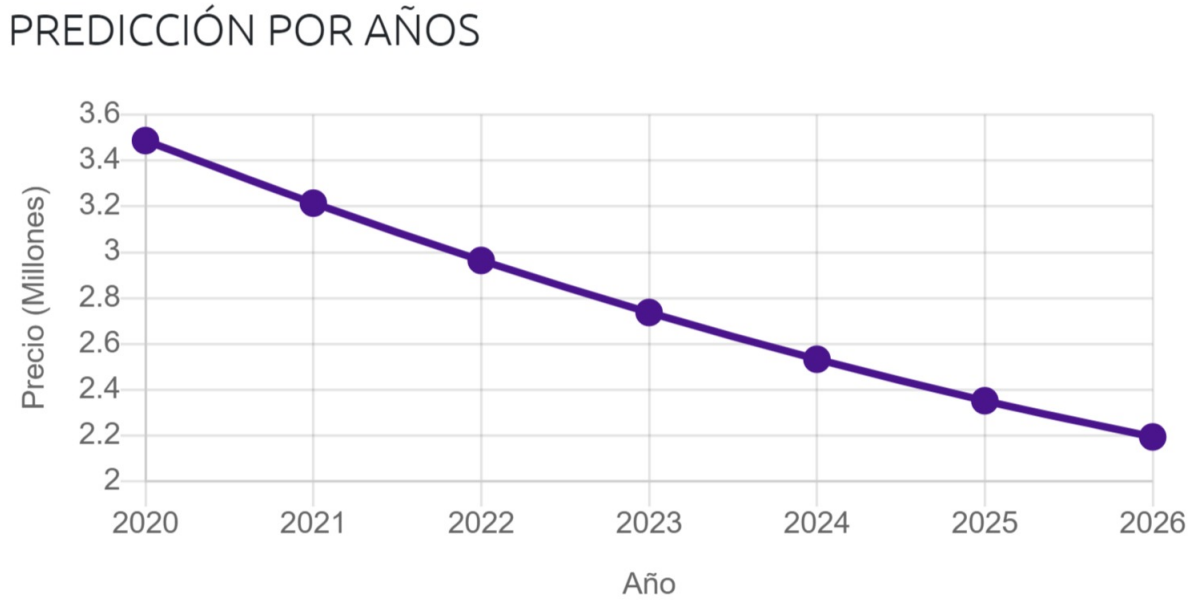

Figura 4.6: Ejemplo de predicción para un Chevrolet Sail NB 2013, bencina y manual con 110.000 kilómetros.

4.4. Base de datos

En esta sección se describirán las entidades más importantes de la base de datos: Maker, Model, FiscalModel, FiscalGroup, Car, Synonym y WordAdapter. En la Figura 4.7 se muestra un gráfico UML de estas.

- **Maker**: Como su nombre lo indica, esta entidad representa un fabricante de autos. Sus campos son:
  - id (integer): identificador único.
  - name (string): nombre.
  - active (boolean): verdadero si el fabricante está activo.
  - date_created (date): fecha de ingreso a la base de datos.
- **Model**: Esta entidad representa un modelo específico de auto. Sus campos son:
  - id (integer): identificador único.
  - name (string): nombre.
  - active (boolean): verdadero si el modelo está activo.
  - date_created (date): fecha de ingreso a la base de datos.
  - maker_id: llave foránea a su Maker asociado.
- **FiscalModel**: Esta entidad representa un modelo (con su versión) fiscal. Sus campos son:
  - id (integer): identificador único.
  - brand (string): nombre de la marca del modelo.
  - model (string): nombre del modelo.
  - version (string): versión del modelo.
  - year (integer): año de salida al mercado.
  - type (string): tipo de vehículo (camioneta, SUV, camión, moto, etc).
  - gas_type (string): tipo de combustible.
  - transmission (string): tipo de transmisión.
  - traction (string): tipo de tracción (4x4, 4x2, AWD, etc).
  - cylinder_capacity (integer): cilindrada.

  - equipment (string): equipamiento con el que cuenta (airbag, aire acondicionado, doble cabina, etc).
  - fiscal_year (integer): año fiscal de esta información. Es importante porque se tiene la información del mismo modelo a través de diferentes años fiscales.
  - fiscal_price (integer): precio fiscal de ese modelo en ese año fiscal.
  - permission_price (integer): precio del permiso de circulación de ese modelo en ese año fiscal.
  - maker_id: llave foránea a su Maker asociado.
  - model_id: llave foránea a su Model asociado.
- **ModelVersionGroup**: Esta entidad representa un grupo de versiones fiscales. Sus campos son:
  - id (integer): identificador único.
  - group_number (integer): número del grupo que representa.
  - group_year (integer): año del grupo.
  - fiscal_model_id: llave foránea a un FiscalModel que pertenece al group_number.
- **Car**: Esta entidad representa un auto publicado en una página de compra/venta. Sus campos son:
  - id (integer): identificador único.
  - name (string): identificador proveniente de la página de origen.
  - origin (string): nombre de la página de origen.
  - brand (string): nombre de la marca del auto.
  - model (string): nombre del modelo del auto.
  - version (string): versión del auto.
  - year (integer): año de salida al mercado.
  - type (string): tipo de vehículo (camioneta, SUV, camión, moto, etc).
  - gas_type (string): tipo de combustible.
  - transmission (string): tipo de transmisión.
  - traction (string): tipo de tracción (4x4, 4x2, AWD, etc).

  - cylinder_capacity (integer): cilindrada.
  - equipment (string): equipamiento con el que cuenta (airbag, aire acondicionado, doble cabina, etc).
  - price (integer): precio listado en la página de origen.
  - kilometers (integer): kilometraje.
  - city (string): ciudad de dónde se publicó el auto.
  - url (string): dirección web de la publicación del auto en la página original.
  - url_principal_image (string): dirección web de la imagen principal de la publicación del auto en la página original.
  - description (string): descripción entregada por el vendedor en la publicación del auto.
  - date_created (date): fecha de creación de la publicación en la página original.
  - date_added (date): fecha de adición a la base de datos de la entrada.
  - date_last_seen (date): fecha de la última vez que la entrada fue revisada por el scraper.
  - deleted (boolean): verdadero si la publicación en la página de origen ya no existe.
  - fiscal_group (integer): número identificador del grupo de versiones fiscales al cual pertenece la entrada.
  - maker_id: llave foránea a su Maker asociado.
  - model_id: llave foránea a su Model asociado.

- **Synonym**: Esta entidad representa palabras claves y su significado o como deben ser interpretadas. Su utilidad está en encontrar características de un auto en su versión o descripción. Sus campos son:
  - id (integer): identificador único.
  - text (string): texto a buscar.
  - meaning (string): si el "text" es encontrado, se debe interpretar con el valor de este campo.
  - type (string): característica a la cual hace referencia el sinónimo. Puede ser gas_type, transmission o traction.

- **WordAdapter**: Esta entidad representa una palabra clave que debe ser reemplazada por otra. Puede o no estar ligada a un fabricante en específico. Sus campos son:
  - id (integer): identificador único.
  - old (string): palabra a encontrar.
  - new (string): palabra a reemplazar si el "old" es encontrado.
  - maker_id: llave foránea a su Maker asociado. Puede no tener uno.

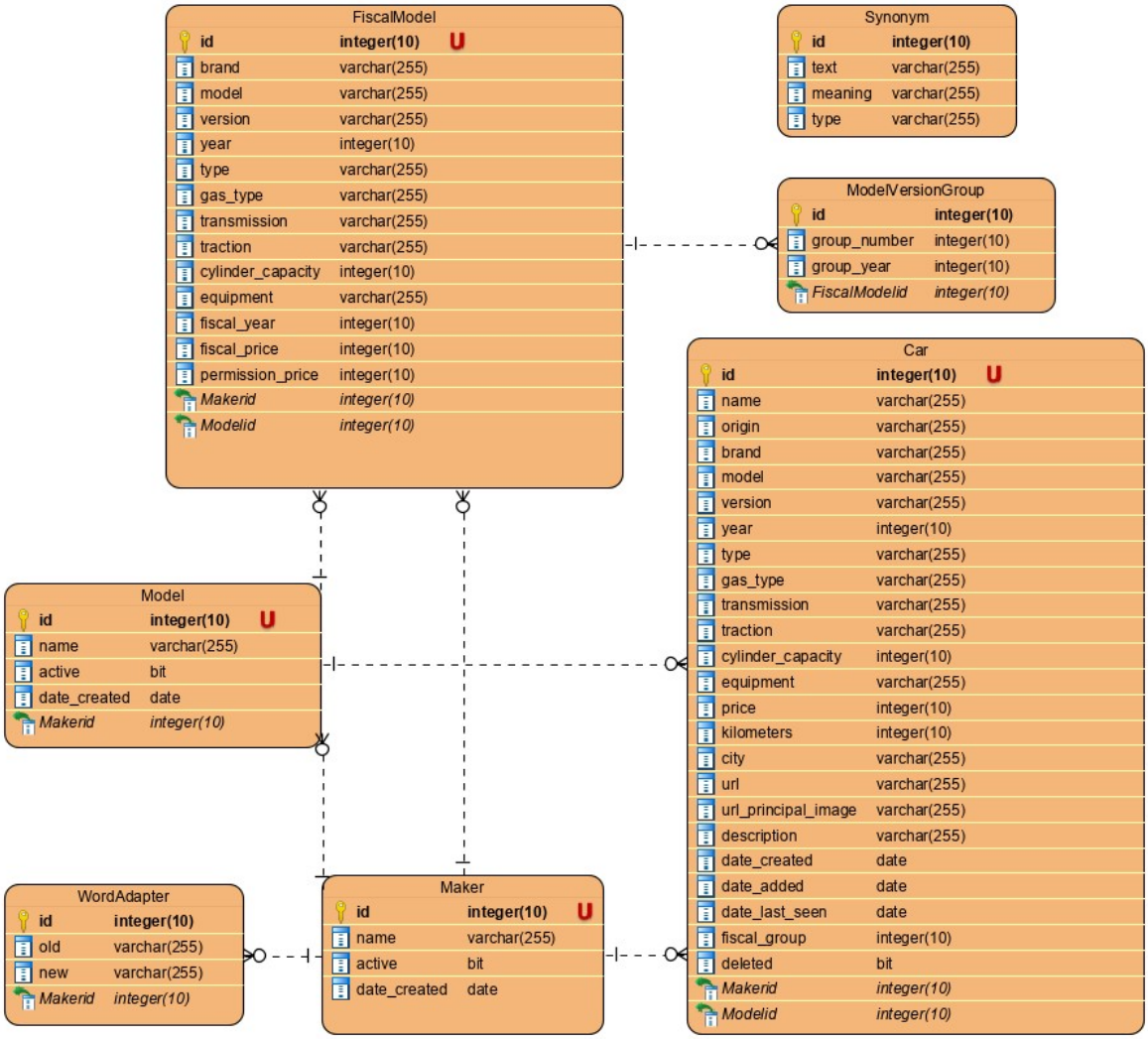

Figura 4.7: UML de entidades más importantes en la base de datos.

4.5. Interfaz gráfica

Para la interfaz gráfica del proyecto, se construyó una aplicación web utilizando el framework Angular [13], la cual cuenta con dos vistas: la de ingreso de datos y la de muestra de resultados. La primera otorga la posibilidad de ingresar los datos del vehículo a tasar de dos formas diferentes: por patente y kilometraje o por marca, modelo, año y kilometraje. Por defecto, la forma de ingreso es la primera mencionada, como se puede observar en la Figura 4.8. Si se desea utilizar la otra forma de ingreso de datos, se debe presionar la frase "O si prefieres, busca tu auto por marca y modelo" para que los *inputs* cambien a como se puede apreciar en la Figura 4.9.

Las entradas de marca, modelo y año son listas de opciones. La primera es una lista que muestra todas las marcas disponibles, la segunda es una lista que muestra los modelos de los autos asociados a la marca escogida en la entrada anterior y la tercera es una lista que muestra los años posibles de la combinación marca y modelo escogida en las entradas anteriores. De esta forma se obliga al usuario a ingresar datos válidos, dicho de otra manera, restringe al usuario a ingresar un trío marca, modelo y año consistente. La información de estas listas son obtenidas a través de solicitudes GET sucesivas al backend. Al detectarse un cambio en el dato de la marca, se hace una solicitud al backend para obtener los modelos de esa marca. Luego, al detectarse un cambio en el dato del modelo, se hace una solicitud para obtener los años en los que ese modelo ha sido lanzado.

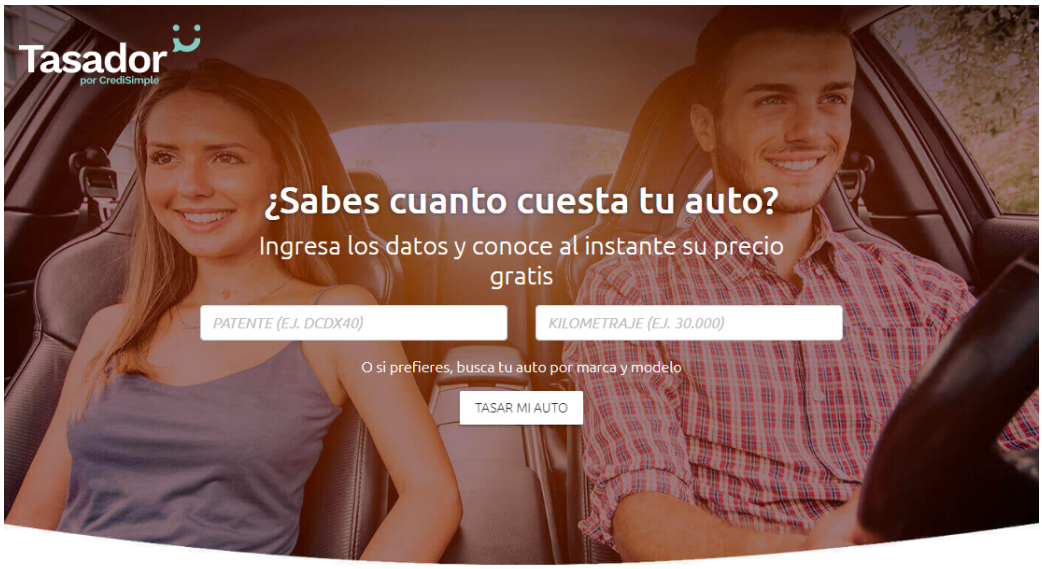

Figura 4.8: Vista ingreso patente y kilometraje.

Figura 4.9: Vista ingreso patente y kilometraje.

El siguiente paso en el flujo es escoger la versión específica del vehículo a tasar, información que es clave para el proceso de predicción como fue explicado en las secciones anteriores. Para esto, independiente de si la información fue ingresada de una u otra forma, se debe presionar el botón "TASAR MI AUTO" que abrirá una ventana de diálogo *modal* con todas las versiones existentes en la base de datos sobre ese auto. Esta lista de versiones es también obtenida a través de una solicitud GET al backend (gatillada al apretar el botón) enviando como parámetro la patente o la marca, el modelo y el año, según corresponda.

Como se puede observar en la Figura 4.10, las versiones son mostradas en cuadros donde se lista su información, que consiste en tipo transmisión, combustible, número de puertas, cilindrada, entre otros. Esto ayuda a que un usuario que no sabe exactamente su versión pueda inferirla a través de las características del vehículo que desea tasar. Para escoger una versión basta con seleccionarla con un *click* sobre ella lo que desencadena un cambio de vista y una solicitud POST al backend con el id de la versión escogida y el kilometraje ingresado, ya que solo esos dos datos son necesarios para realizar la tasación. Al recibir respuesta del backend, se muestran los resultados como se ilustra en la Figura 4.11, donde se puede apreciar el nombre completo del auto buscado, el precio actual, un gráfico de su precio calculado a través de los años, un cuadro con las características del vehículo y un cuadro con los rangos de precios calculados. Finalmente, si se desea tasar otro auto, en la esquina superior derecha existe un botón que lleva a la vista anterior.

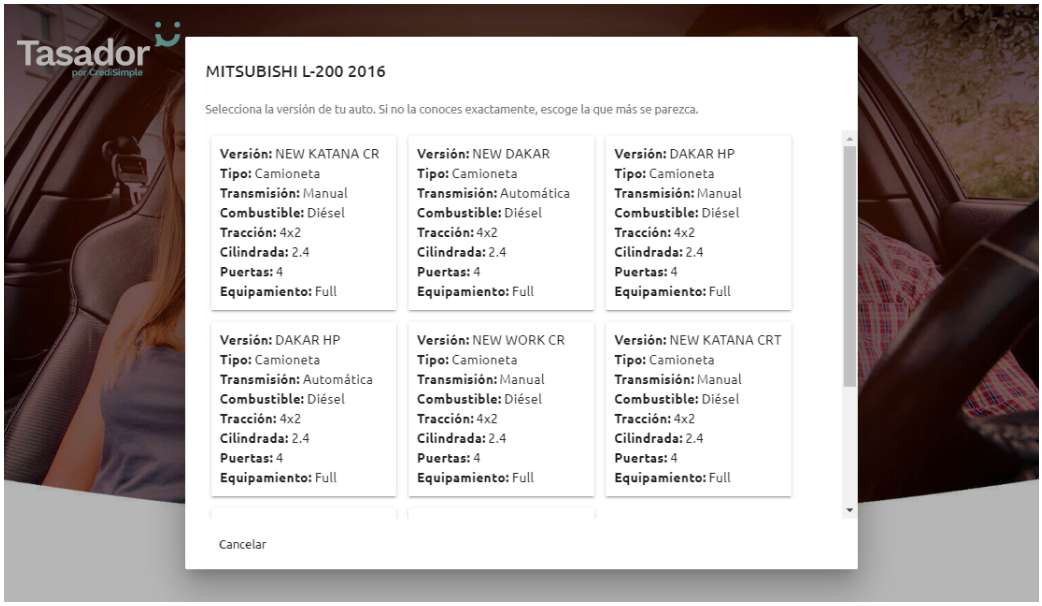

Figura 4.10: Vista elección de versión especifica.

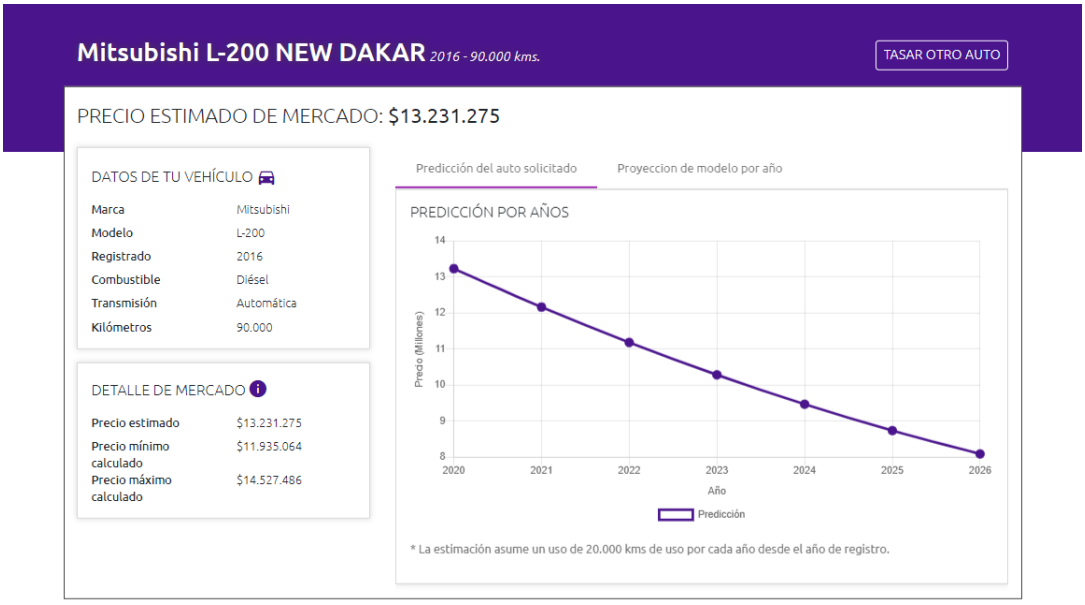

Figura 4.11: Vista de resultados.

Capítulo 5

Resultados y Validación

5.1. Predictor

Para la validación del predictor se obtuvieron estadísticas de cuáles son los Clanes con mayor cantidad de autos disponibles en la base de datos[1], que contiene los autos obtenidos por el scraper. La cantidad total de Grupos y Clanes se pueden observar en la Tabla 5.1 y la cantidad de datos junto con la "cantidad usable", que hace referencia a entradas que tienen asociada un grupo de versiones, se pueden observar en la Tabla 5.2.

	Cantidad
Grupos de versiones	14.817
Clanes de versiones	4.857

Tabla 5.1: Cantidad de grupos y clanes existentes.

Año de "scrapeo"	Cantidad total	Cantidad usable
2017	385.977	244.548
2018	0	0
2019	862.885	468.280
2020	295.037	173.970

Tabla 5.2: Cantidad de entradas presentes en la base de datos.

Como se puede apreciar, hay datos que fueron obtenidos en el año 2017 y corresponden al scraper antiguo que tenía la empresa al momento de iniciar este trabajo, el cual ya se había mencionado en 4.1. Los datos obtenidos en 2019 y 2020 corresponden al scraper actual. En total se tienen 886.798 autos usables de un total de 1.543.899.

Con estos datos, se obtuvieron los 30 Clanes con mayor cantidad de datos disponibles. Cabe destacar que sólo se permite un Clan por modelo. Por ejemplo existían dos clanes de Chevrolet Sail con más de 10.000 entradas, pero sólo se dejo el que tenía más (11.373). En la Tabla 5.3 se puede apreciar el detalle de los clanes escogidos y su cantidad asociada.

[1]Con fecha Julio 2020

Marca del Clan	Modelo del Clan	Cantidad de entradas
Chevrolet	Sail	11.373
Toyota	Yaris	11.180
Chevrolet	Spark	9.897
Mitsubishi	L-200	9.330
Suzuki	Grand Nomade	8.630
Hyundai	Accent	8.475
Hyundai	Tucson	8.426
Kia	Morning	8.221
Hyundai	Santa fe	7.498
Suzuki	Swift	7.365
Kia	Sorento	6.210
Chevrolet	Corsa	5.666
Mazda	3	5.607
Peugeot	208	5.192
Chevrolet	D-Max	4.846
Mitsubishi	Montero	4.788
Suzuki	Celerio	4.779
Kia	Rio 5	4.378
Peugeot	206	4.292
Kia	Cerato	4.186
Chevrolet	Aveo	4.186
Toyota	Rav-4	4.097
Ford	Ranger	4.017
Suzuki	Alto	3.978
Mazda	CX-5	3.713
Jeep	Compass	3.704
Nissan	Terrano	3.635
Samsung	SM3	3.633
Kia	Rio 4	3.630
Nissan	Navara	3.570

Tabla 5.3: Clanes escogidos para la validación junto a su cantidad de entradas disponibles.

Las pruebas de validación se hicieron de la siguiente forma:

a. Se escoge al azar un auto "scrapeado" que sea parte del clan que se quiere probar y que **haya sido ingresado este año 2020 a la base de datos**. Esto será considerado el *ground truth*. La lista de vehículos seleccionados se puede apreciar a continuación. Para efectos prácticos, su ubicación en esta lista será también su nombre de prueba, por ejemplo, si su ubicación es la quinta posición, su prueba es la Prueba 5.

    1. Chevrolet Sail 2011 con 109.500 kilómetros, grupo: **NB ls pwr**.
    2. Toyota Yaris 2011 con 150.000 kilómetros, grupo: **sport xli sedan abs std gli dab**.
    3. Chevrolet Spark 2017 con 60.000 kilómetros, grupo: **GT LT effect dh e5 ac bt**.
    4. Mitsubishi L-200 2011 con 276.752 kilómetros, grupo: **KATANA crm diésel crt**.
    5. Suzuki Grand Nomade 2016 con 36.051 kilómetros, grupo: **GLX sport e4 se nav**.
    6. Hyundai Accent 2007 con 176.000 kilómetros, grupo: **mc gls 2ab 16v ac gl ps crdi diésel 3p full**.
    7. Hyundai Tucson 2014 con 43.800 kilómetros, grupo: **GL lm ac 2ab fl a wce1 wce2 crdi diésel**.
    8. Kia Morning 2014 con 72.500 kilómetros, grupo: **EX dh ac dab sport abs**.
    9. Hyundai Santa Fe 2011 con 135.000 kilómetros, grupo: **GLS fl cm 7p diésel crdi**.
    10. Suzuki Swift 2006 con 80.000 kilómetros, grupo: **glx ga gl**.
    11. Kia Sorento 2011 con 128.000 kilómetros, grupo: **EX gsl diésel 7 plazas**.
    12. Chevrolet Corsa 2009 con 167.000 kilómetros, grupo: **HB DH 3p ac 5p ltd**.
    13. Mazda 3 2011 con 100.000 kilómetros, grupo: **sedan s ac v sport te**.
    14. Peugeot 208 2015 con 56.000 kilómetros, grupo: **active pack puretech e6 diésel allure 68hp**.
    15. Chevrolet D-Max 2017 con 70.000 kilómetros, grupo: **E4 CC D DAB ABS DIÉSEL**.
    16. Mitsubishi Montero 2012 con 162.000 kilómetros, grupo: **SPORT G2 diésel glx**.
    17. Suzuki Celerio 2012 con 74.000 kilómetros, grupo: **GLX AC ll**.

18. Kia Rio 5 2016 con 51.000 kilómetros, grupo: **EX 6mt ac dab abs euro copa 4at sport**.

19. Peugeot 206 2009 con 165.000 kilómetros, grupo: **xn 3p 5p xr aa 16v x-line contact mec**.

20. Kia Cerato 2017 con 60.000 kilómetros, grupo: **ex 6mt special pack dh ac dab nvo mt full abs sx eurov 6at**.

21. Chevrolet Aveo 2007 con 140.000 kilómetros, grupo: **LT NB AC 16v**.

22. Toyota Rav-4 2012 con 78.000 kilómetros, grupo: **ADVANTAGE super lujo**.

23. Ford Ranger 2011 con 148.000 kilómetros, grupo: **DIÉSEL xl plus hirider xlt**.

24. Suzuki Alto 2008 con 90.000 kilómetros, grupo: **GL 800**.

25. Mazda CX-5 2012 con 60.000 kilómetros, grupo: **R**.

26. Jeep Compass 2011 con 101.000 kilómetros, grupo: **SPORT lx**.

27. Nissan Terrano 2010 con 300.000 kilómetros, grupo: **dx diésel pick-up dxs turbo sin l4858c aa ax**.

28. Samsung SM3 2006 con 127.000 kilómetros, grupo: **pe entry se mec llantas apertura distancia aut le abs**.

29. Kia Rio 4 2014 con 91.000 kilómetros, grupo: **UB EX 5mt 6mt**.

30. Nissan Navara 2015 con 90.000 kilómetros, grupo: **DIÉSEL se xe-1 lux le se-2**.

b. Se construye una regresión utilizando autos que pertenezcan al mismo Clan del vehículo de muestra pero que hayan sido ingresados a la base de datos en los años 2017 y 2019 (año 2018 no tiene autos). De esta forma **la predicción se hará sin muestras actuales del auto**.

c. Se calcula el precio de un auto con las mismas características del vehículo de muestra (*ground truth*) en la regresión creada en el paso anterior.

d. Se compara el precio obtenido en el punto anterior con un precio promediado del vehículo de muestra, el cual se calcula obteniendo autos idénticos en características a éste último en la base de datos (que también hayan sido ingresados a ésta en 2020) y con un kilometraje parecido (más o menos 1000 km). Con ellos se hace un promedio de los precios con el fin de evitar que el precio a comparar sea un *outlier*.

e. De la comparación anterior se obtiene la diferencia en valor absoluto de los precios y también el porcentaje de error entre la diferencia y el precio esperado. Se espera que el error en el precio no sea mayor a $500.000 en términos absolutos y a 8 % en términos porcentuales. Estos valores fueron establecidos en conversaciones con la empresa.

Como se mencionó anteriormente, la intención es comparar dos métodos de construcción de una regresión polinomial para escoger la que se adecua mejor al problema que se quiere abordar. Estos métodos son regresión polinomial utilizando mínimos cuadrados y por método de Ridge.

Por lo tanto, las pruebas se realizaron primero con uno y luego con el otro. Los resultados de estas se pueden apreciar en la Tabla 5.4 para mínimos cuadrados y en la Tabla 5.5 para Ridge. Con estos resultados, se calcula el promedio de diferencia de precios, la desviación estándar de este y el porcentaje de error entre la diferencia de precio y el precio real, también con su desviación estándar. Estos datos pueden ser encontrados en la Tabla 5.6.

Prueba	Precio real (promedio)	Precio (calculado)	Diferencia	Error (%)
1	$3.260.000	$2.794.032	$465.968	14,29
2	$3.895.000	$4.327.597	$432.597	11,11
3	$4.470.000	$4.196.666	$273.334	6,11
4	$6.500.000	$6.112.793	$387.207	5,96
5	$7.646.666	$7.075.647	$571.019	7,47
6	$2.450.000	$2.318.605	$131.395	5,36
7	$7.500.000	$7.613.567	$113.567	1,51
8	$4.231.666	$4.052.198	$179.468	4,24
9	$7.036.000	$7.535.363	$499.363	7,10
10	$2.950.000	$2.534.535	$415.465	14,08
11	$6.300.000	$6.160.067	$139.933	2,22
12	$2.300.000	$2.083.771	$216.229	9,40
13	$4.800.000	$4.944.200	$144.200	3,00
14	$6.390.000	$5.586.179	$803.821	12,58
15	$10.797.857	$10.038.605	$759.252	7,03
16	$8.800.000	$8.783.842	$16.158	0,18
17	$3.201.666	$3.120.540	$81.126	2,53
18	$6.266.000	$6.052.694	$213.306	3,40
19	$2.790.000	2.696.773	$93.227	3,34
20	$7.868.000	$7.754.154	$113.846	1,45
21	$2.912.000	$2.505.891	$406.109	13,95
22	$6.600.000	$6.212.142	$387.858	5,88
23	$7.200.000	$6.461.629	$738.371	10,26
24	$1.558.000	$1.778.843	$220.843	14,17
25	$7.600.000	$6.678.945	$921.055	12,12
26	$5.272.500	$4.843.793	$428.707	8,13
27	$6.000.000	$5.828.237	$171.763	2,86
28	$3.000.000	$1.153.302	$1.846.698	61,56
29	$5.433.333	$5.043.841	$389.492	7,17
30	$8.447.500	$8.004.632	$442.868	5,24

Tabla 5.4: Resultados utilizando mínimos cuadrados.

Prueba	Precio real (promedio)	Precio calculado	Diferencia	Error (%)
1	$3.260.000	$2.991.530	$268.470	8,24
2	$3.895.000	$4.070.441	$175.441	4,50
3	$4.470.000	$4.128.344	$341.656	7,64
4	$6.500.000	$6.839.425	$339.425	5,22
5	$7.646.666	$7.163.348	$483.318	6,32
6	$2.450.000	$2.517.040	$67.040	2,74
7	$7.500.000	$7.541.361	$41.361	0,55
8	$4.231.666	$4.030.248	$201.418	4,76
9	$7.036.000	$7.390.648	$354.648	5,04
10	$2.950.000	$2.549.017	$400.983	13,59
11	$6.300.000	$6.519.393	$219.393	3,48
12	$2.300.000	$2.051.868	$248.132	10,79
13	$4.800.000	$4.866.540	$66.540	1,39
14	$6.390.000	$5.912.404	$477.596	7,47
15	$10.797.857	$10.380.554	$417.303	3,86
16	$8.800.000	$8.340.906	$459.094	5,22
17	$3.201.666	$3.157.246	$44.420	1,39
18	$6.266.000	$6.126.670	$139.330	2,22
19	$2.790.000	$2.696.758	$93.242	3,34
20	$7.868.000	$7.581.043	$286.957	3,65
21	$2.912.000	$2.577.351	$334.649	11,49
22	$6.600.000	$6.351.620	$248.380	3,76
23	$7.200.000	$6.868.066	$331.934	4,61
24	$1.558.000	$1.882.779	$324.779	20,85
25	$7.600.000	$8.246.589	$646.589	8,51
26	$5.272.500	$4.982.065	$290.435	5,51
27	$6.000.000	$5.795.468	$204.532	3,41
28	$3.000.000	$2.869.956	$130.044	4,33
29	$5.433.333	$5.032.856	$400.477	7,37
30	$8.447.500	$8.226.274	$221.226	2,62

Tabla 5.5: Resultados utilizando Ridge.

	Mínimos cuadrados	Ridge
Promedio diferencia de precio	$400.142	$275.294
Desviación estándar de la diferencia en precio	$359.536	$147.102
Porcentaje de error diferencia de precio	8,79 %	5,80 %
Desviación estándar de error en diferencia de precio	10,83 %	4,16 %

Tabla 5.6: Estadísticas de resultados.

Como se puede observar, ambos métodos logran predecir de manera bastante acertada, sin embargo Ridge es muy superior en desempeño a mínimos cuadrados. Un error de aproximadamente 6 % y una diferencia promedio de aproximadamente $275.000 son resultados satisfactorios teniendo en cuenta lo esperado inicialmente, por lo tanto, se da por validado el método planteado, en base a grupos y clanes de versiones, como un enfoque correcto para la solución del problema.

5.2. Interfaz gráfica

Para la validación de la usabilidad de la interfaz gráfica se barajaron tres opciones: cuestionario SUMI [15], cuestionario USE [16] y cuestionario SUS [5]. Sin embargo, los dos primeros fueron rápidamente descartados por su gran cantidad de preguntas (50 y 30 respectivamente) y la inconsistencia de algunas preguntas con el contexto donde se desarrollará la prueba. Por lo tanto el cuestionario escogido es el System Usability Scale (SUS).

Este cuestionario tiene 10 preguntas, las cuales se responden en una escala de 1 a 5 puntos, siendo uno "muy en desacuerdo" y cinco "muy de acuerdo". Al usuario se le solicita que use el sistema e inmediatamente después se le pide que conteste el cuestionario. El puntaje interesante es el total más allá del puntaje de las preguntas por separado, las cuales son:

1. Si trabajara en la industria de compra/venta de autos, creo que usaría esta aplicación frecuentemente.

2. Encuentro esta aplicación innecesariamente compleja (encuentro que esta aplicación es más compleja de lo que debería ser).

3. Creo que la aplicación fue fácil de usar.

4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación.

5. Las funciones de esta aplicación están bien integradas (las funciones que tiene la aplicación son justo las necesarias).

6. Creo que la aplicación es muy inconsistente.

7. Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida.

8. Encuentro que la aplicación es muy difícil de usar.

9. Me siento confiado al usar esta aplicación.

10. Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación.

Como se puede apreciar, hay preguntas donde el ideal de respuesta es un uno y en otros donde el ideal es un cinco. Es por esto que la forma de suma de los puntajes se hace según las siguientes dos reglas:

- Las preguntas 1, 3, 5, 7 y 9 se suman como el puntaje individual menos 1.
- Las preguntas 2, 4, 6, 8 y 10 se suman como 5 menos el puntaje individual.

Entonces la suma de los puntajes entrega un valor entre 0 y 40. A esta suma se le multiplica por 2.5 para llevarla a un rango de 0 a 100. Este valor es el puntaje SUS otorgado por un participante del cuestionario a la plataforma y se considera positivo si es mayor a 68. En esta ocasión se contó con un universo de 15 personas que probaron la plataforma y luego contestaron la encuesta. Este grupo estaba compuesto por 5 mujeres y 10 hombres, con edades de entre 20 años hasta 55 años. Ninguno de ellos era experto en el área automotriz, pero si la mayoría tenía auto propio. Además, para todos fue su primera interacción con la plataforma. Los puntajes SUS de los participantes ordenados de menor a mayor se pueden observar en la Figura 5.1 a continuación.

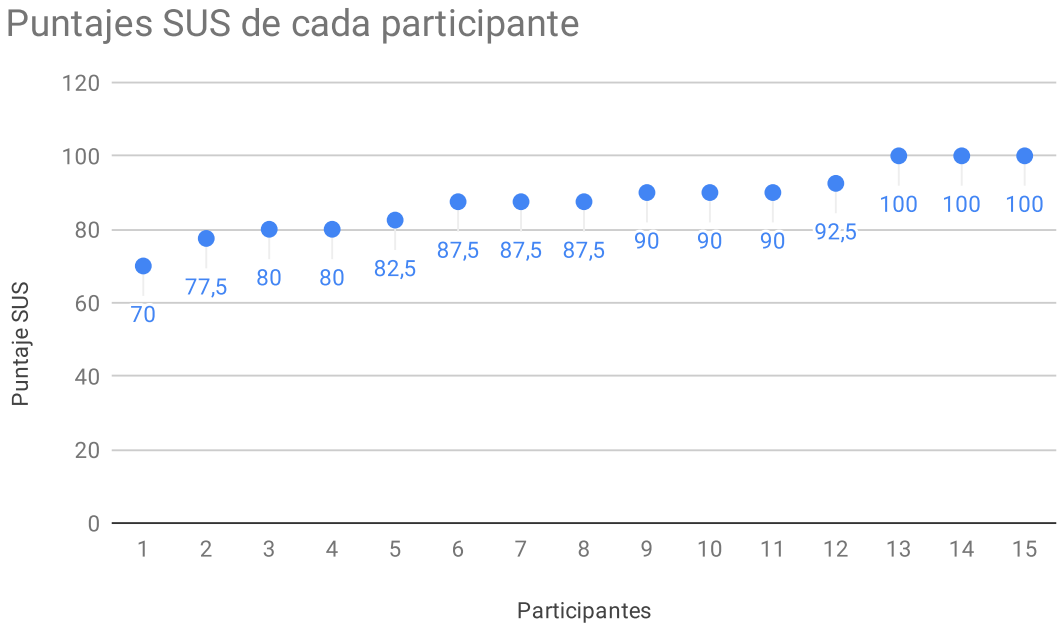

Figura 5.1: Puntaje SUS de los participantes.

Como se puede observar, todos los participantes entregaron un puntaje mayor a 68, por lo que se considera la evaluación como positiva. Además, el promedio es de 87.67 lo cual es satisfactorio. Por otro lado, desde la Figura 5.2 hasta la Figura 5.11 se presentan los resultados globales para cada pregunta en donde se puede apreciar que hay tres que tienen mayor dispersión de respuestas: la cuatro, la ocho y la diez. Esta información ayuda a identificar que se puede mejorar en una próxima iteración de la interfaz gráfica, lo cual será discutido como trabajo futuro en el capítulo próximo.

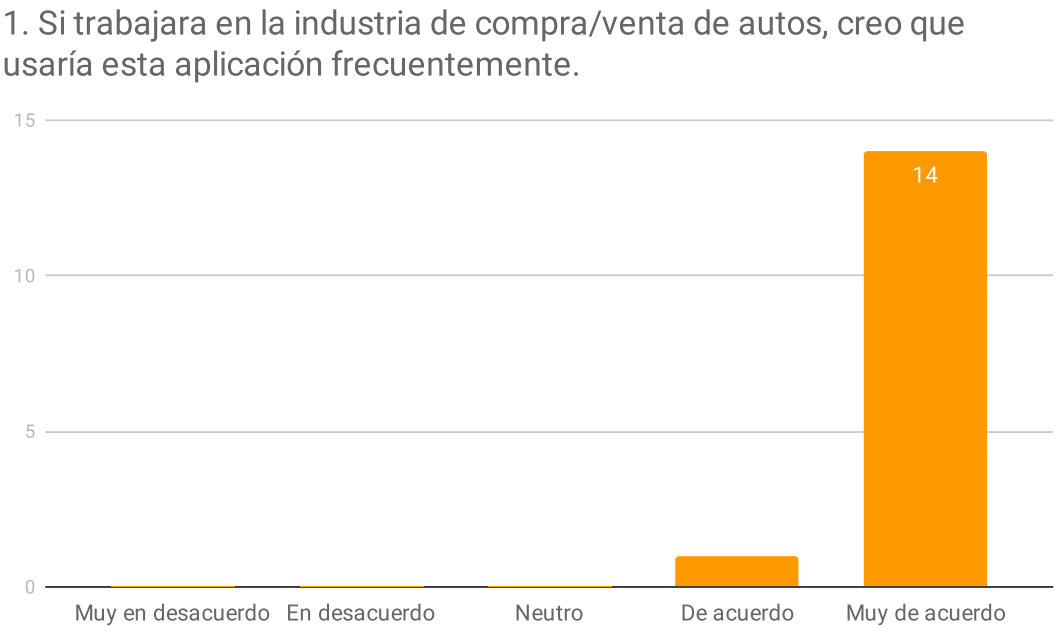

Figura 5.2: Resultados pregunta uno del cuestionario SUS.

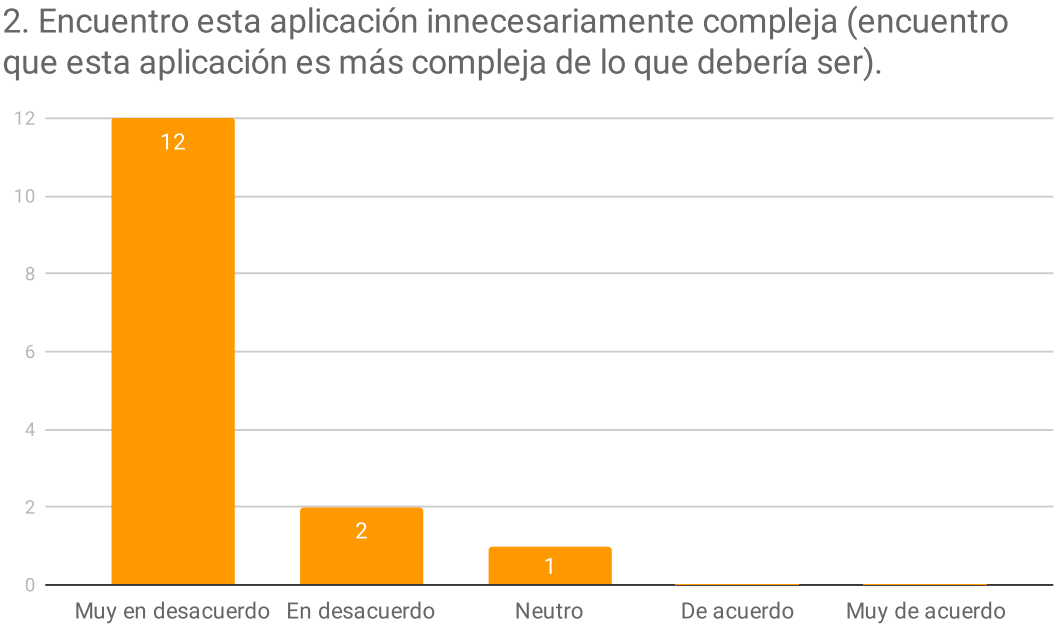

Figura 5.3: Resultados pregunta dos del cuestionario SUS.

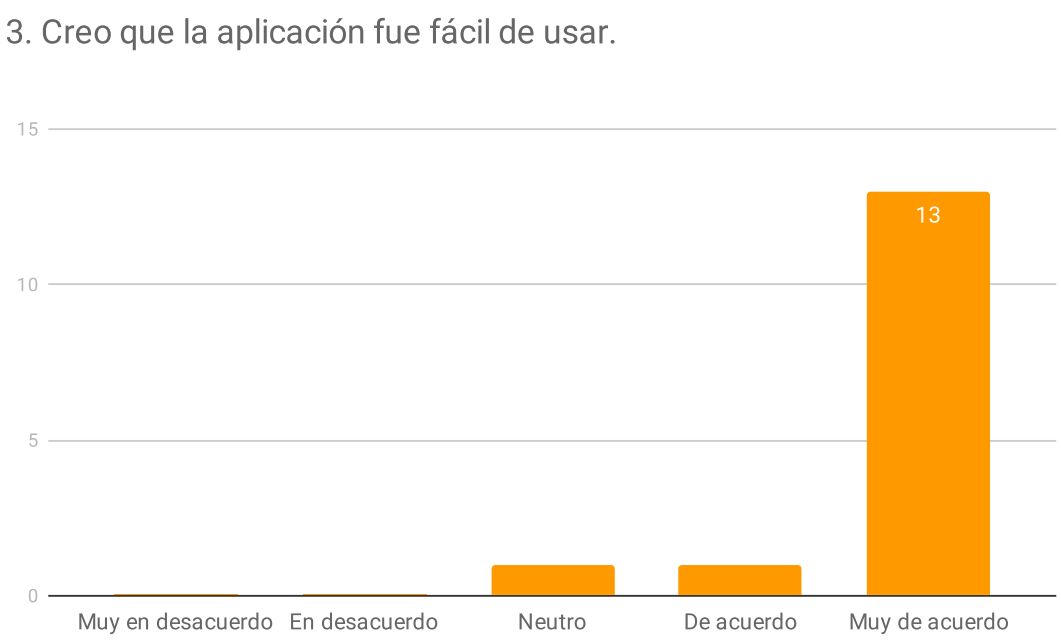

Figura 5.4: Resultados pregunta tres del cuestionario SUS.

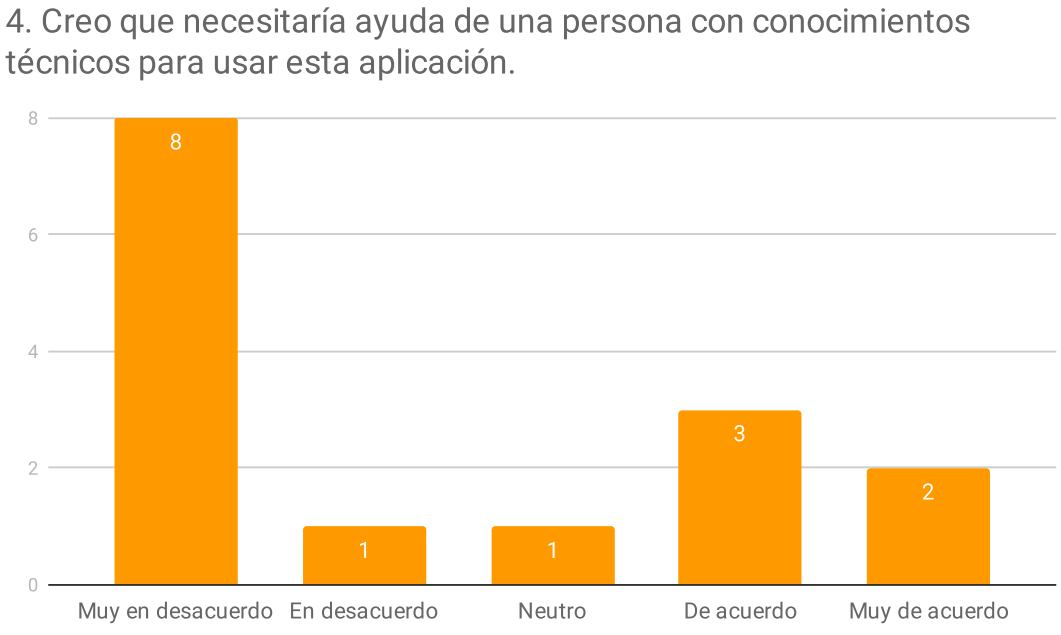

Figura 5.5: Resultados pregunta cuatro del cuestionario SUS.

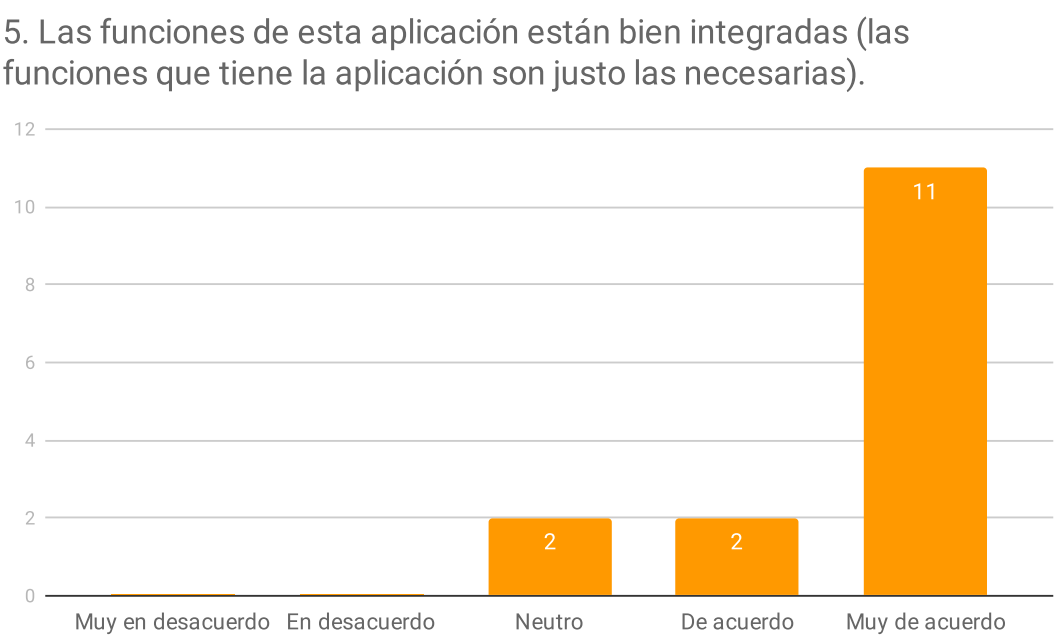

Figura 5.6: Resultados pregunta cinco del cuestionario SUS.

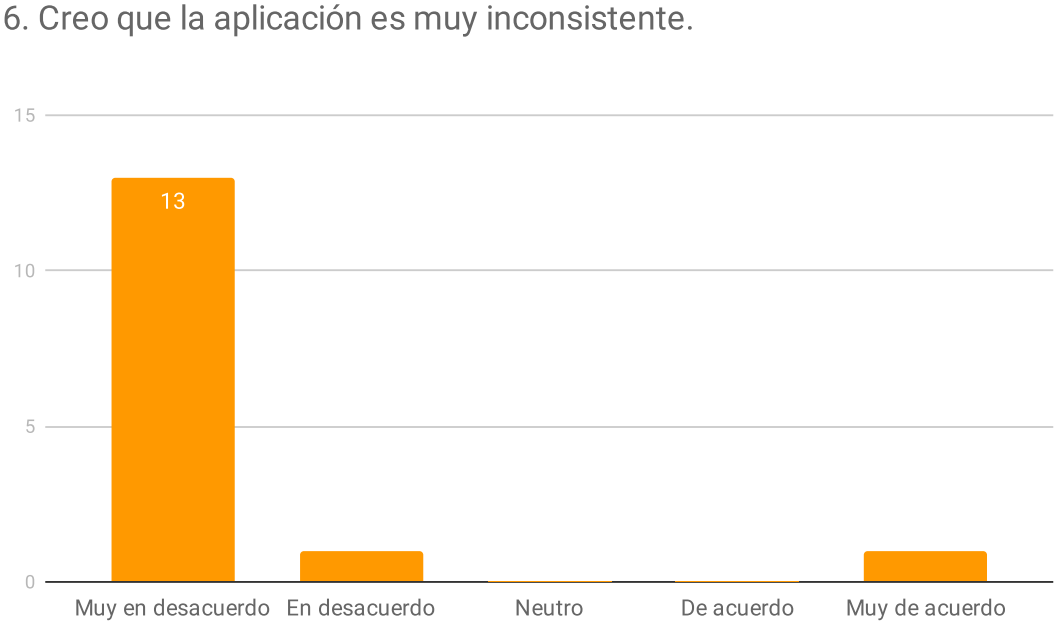

Figura 5.7: Resultados pregunta seis del cuestionario SUS.

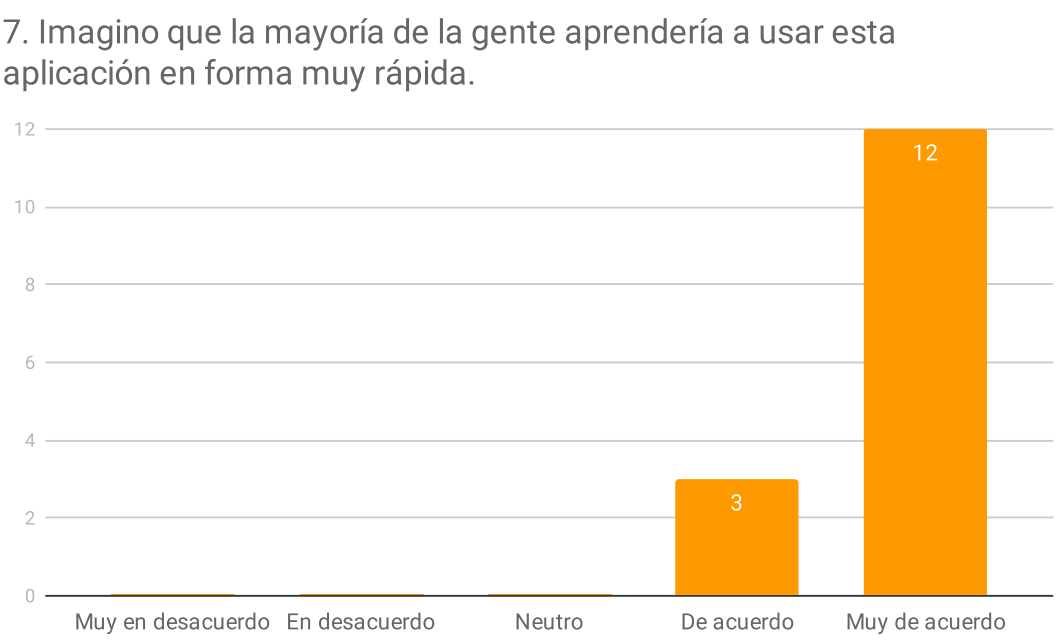

Figura 5.8: Resultados pregunta siete del cuestionario SUS.

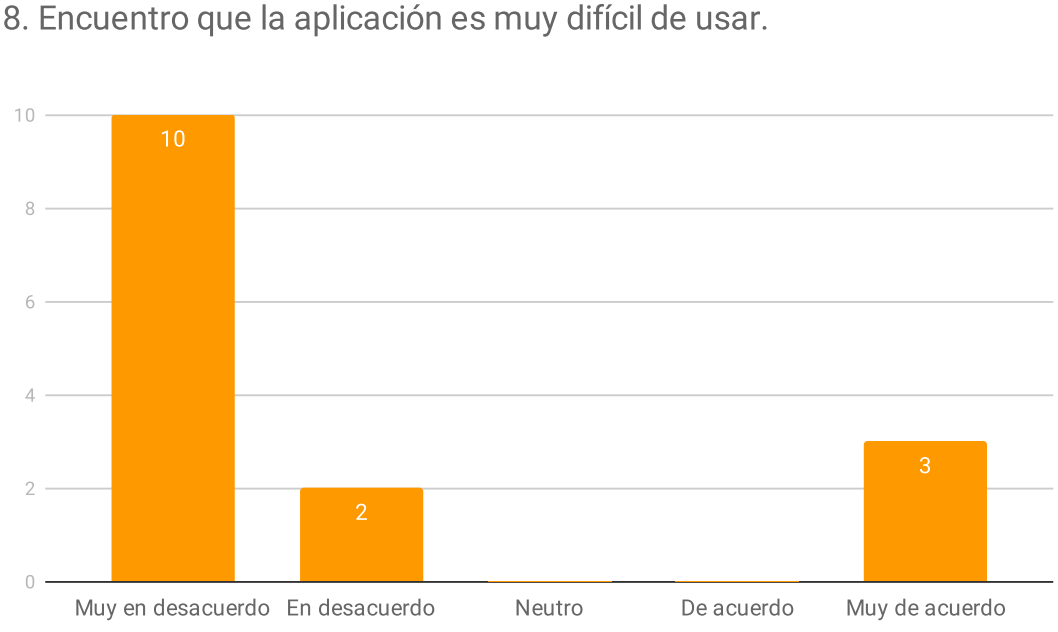

Figura 5.9: Resultados pregunta ocho del cuestionario SUS.

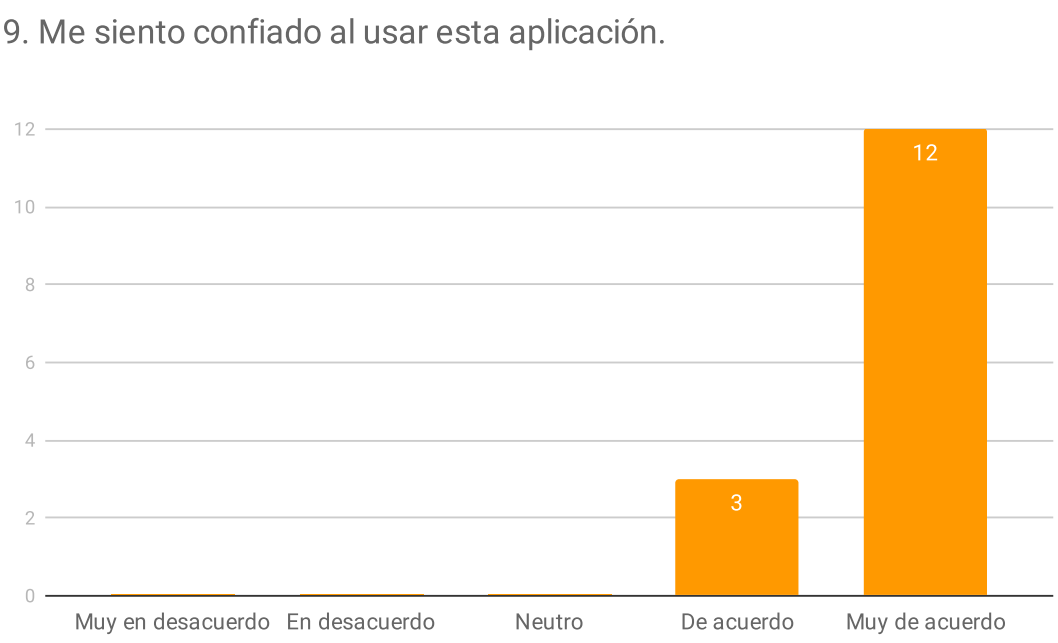

Figura 5.10: Resultados pregunta nueve del cuestionario SUS.

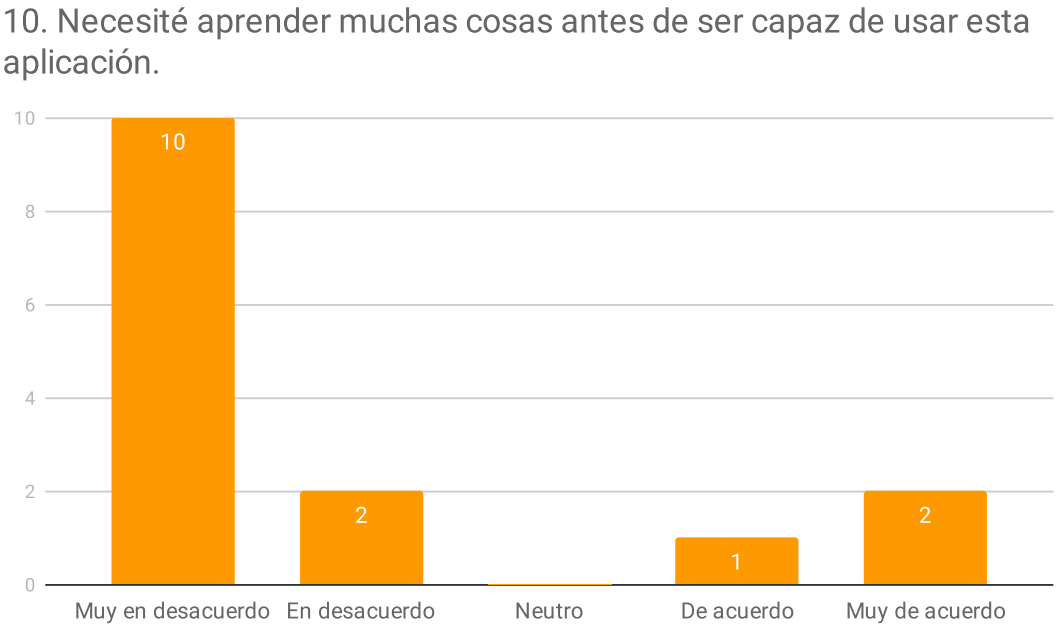

Figura 5.11: Resultados pregunta diez del cuestionario SUS.

Capítulo 6

Conclusión y Trabajo Futuro

A lo largo del desarrollo descrito en este documento, es posible notar que la tarea de tasar y predecir la devaluación de un vehículo usado no es directa y se deben considerar diversos factores para llegar a una respuesta. En primera instancia se tiende a pensar que basta con recopilar autos que calcen con la marca, modelo y año del vehículo consultado pero se demostró que el dato clave para la predicción es la versión exacta del modelo. Sin embargo esto genera dos problemas los cuales sus soluciones son consideradas por el autor como la mayor contribución de este trabajo: la distinción por versión específica de los autos y la correcta indexación de las entradas provenientes del scraper.

Para el primer problema se propuso una solución basada en la agrupación de versiones con el fin de tratarlas como una sola, lo cual se denominó Grupo de Versiones (sección 4.2.1). Como fue explicado, su cálculo se basa en el precio al cual las versiones salieron al mercado y se demostró que el método genera grupos consistentes. Luego, para mapear los Grupos de Versiones a través de los años se generó el sistema de Clanes de Versiones, el cuál también demostró resultados positivos. Parte fundamental de este paso fue la unificación de las palabras claves, la cual fue posible gracias a la base de datos de palabras de adaptación, que fue generada a través de una revisión manual minuciosa de todas las versiones sin modificar (35.000 entradas aproximadamente).

Para el segundo problema se propuso una solución de formateo de entradas, basado en palabras de adaptación, para su posterior indexación basada en calces de palabras fuertes y palabras débiles con los nombres de los grupos de versiones (secciones 4.2.2 y 4.2.3). Estos métodos demostraron ser efectivos y su resultado se considera satisfactorio.

Si bien la tasación y predicción del valor de un vehículo es el eje principal del trabajo descrito en esta memoria, esto no sería posible si no existiesen los datos para generar los cálculos. En este trabajo también se construyó satisfactoriamente un scraper de páginas de compra y venta de autos en Chile, abarcando la gran mayoría de estas (sección 4.1). El scraper está basado en colas de mensajes lo que lo hace robusto a errores: si un mensaje no se puede procesar solamente se deja pasar y no detiene todo el proceso. Sin embargo, como se discutió anteriormente, es un software con poca capacidad de abstracción lo que significa que si una página web cambia la disposición de sus elementos, el scraper de ésta dejará de

funcionar. Como se puede inferir, este es un problema que no se puede prever, ya que no se puede saber cómo ni cuándo cambiará una página. No obstante, se podría programar un detector de cambio de página (por ejemplo, si de un día para otro esa página se quedo sin autos en la base de datos) para que así se pueda hacer el arreglo de forma oportuna, lo cual se deja propuesto como un trabajo futuro.

El otro desarrollo incluido en el proyecto fue una aplicación web para permitir al usuario tasar el auto que desee. Se opto por una interfaz simple solo con la información necesaria, como fue expuesto en la sección 4.5. Esta aplicación se comunica con el backend a través de consultas REST, que hacen posible mostrar datos inteligentemente como solo modelos de una marca en específico o años disponibles según combinación marca-modelo.

En cuanto a resultados y validación de los trabajos efectuados, se realizaron pruebas tanto para medir la precisión del predictor como para medir la usabilidad de la interfaz gráfica. Para el primero se utilizaron datos reales como *ground truth* y se compararon contra valores generados por un predictor construido solo con datos de años anteriores (2017 y 2019), como fue descrito en la sección 5.1. Para la construcción del predictor se utilizaron dos métodos diferentes, regresión polinomial por mínimos cuadrados y regresión por método Ridge, de los cuales el segundo logro mejores resultados obteniendo un error promedio del 5,8 % entre el precio calculado y el precio observado. En un comienzo la expectativa del error era de 8,0 %, por lo que se concluye que la forma propuesta para abordar el problema, o sea, modelación a través de grupos y clanes de versiones, es válida. Por su parte, la validación de la interfaz gráfica se realizó a través de una encuesta de usabilidad estándar SUS. Como se expuso en la sección 5.2, el puntaje promedio entregado por cada una de las quince personas encuestadas fue de 87.67, donde además cada uno de los puntajes fue superior a al limite considerado como positivo, es decir 68. Se concluye que la implementación escogida satisface los requisitos esperados.

A partir de los resultados obtenidos es posible encontrar distintos elementos que pueden ser utilizados para la realización de trabajos futuros. En primer lugar el ya mencionado "notificador de cambio de página en el scraper" sería una extensión útil, ya que la forma actual de detectar el cambio es revisando manualmente, lo que genera respuestas tardías al cambio y eso provoca que exista información que se pierda al no poder ser scrapeada.

En segundo lugar existe un problema, hasta ahora sin solución aparente, que son las entradas a las cuales no se les puede asignar un grupo de versiones. Esto puede ocurrir por principalmente dos factores: la entrada tiene versión vacía desde su página de origen o su versión no coincide con ninguna de las ingresadas en el sistema. Para el primer caso se podría utilizar la comparación entre otras características como tracción, cilindrada o cantidad de puertas, si es que esa información existe. Para el segundo caso no se tienen sugerencias al momento de escribir este documento.

Por último, en la validación de usabilidad de la interfaz gráfica existieron tres preguntas con respuestas distribuidas, las cuales fueron: "Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación", "Encuentro que la aplicación es muy difícil de usar" y "Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación". Para entender mejor estos resultados, se generó una conversación posterior con los encuestados donde la mayoría hizo hincapié en que no tenían conocimiento sobre su versión especifica y que la vista de elección de ésta (Figura 4.10) era un poco "abrumante" por la cantidad de opciones mostradas. De esta forma, se cuenta con información suficiente para hacer los cambios que se consideren pertinentes en una siguiente iteración de la aplicación.

Finalmente, se concluye que el trabajo descrito en esta memoria cumple de buena forma con los objetivos planteados al inicio y entrega cimientos para su mejora en una nueva iteración del mismo.

Bibliografía

[1] Pekka Abrahamsson, Outi Salo, Jussi Ronkainen, and Juhani Warsta. Agile software development methods: Review and analysis. In *VTT Publications 478*, pages 1–107, 2017.

[2] Adel Alshamrani and Abdullah Bahattab. A comparison between three sdlc models waterfall model, spiral model, and incremental/iterative model. In *IJCSI International Journal of Computer Science Issues, Volume 12, Issue 1, No 1*, pages 106–111, 2015.

[3] Autobook. Informe autobook. `https://www.autobook.cl/informe-autobook/detalle`, 2016. [Online; accedido 26-Enero-2020].

[4] Barry Boehm. A spiral model of software development and enhancement. In *Computer Vol. 21*, pages 61–72, 1988.

[5] John Brooke. Sus: A "quick and dirty" usability scale. In *Usability evaluation in industry*, page 189–194, 1996.

[6] CheerioJS. Cheerio. `https://github.com/cheeriojs/cheerio`, 2011. [Online; accedido 26-Enero-2020].

[7] CNN. Medio millón de vehículos se sumaron en dos años a la región metropolitana. `https://www.cnnchile.com/pais/medio-millon-de-vehiculos-se-sumaron-en-dos-anos-a-la-region-metropolitana_20190114/`, 2019. [Online; accedido 22-Febrero-2020].

[8] David Cournapeau. Sklearn. `https://scikit-learn.org/`, 2007. [Online; accedido 26-Enero-2020].

[9] Software Foundation Django. Django. `https://www.djangoproject.com/`, 2005. [Online; accedido 22-Febrero-2020].

[10] Facebook. React. `https://es.reactjs.org/`, 2013. [Online; accedido 22-Febrero-2020].

[11] Gerhard Fischer. The software technology of the 21st century: From software reuse to collaborative software design. In *Proceedings of ISFST'2001: International Symposium on Future Software Technology*, pages 1–8, 2001.

[12] Tomás Flores. La explosión del parque automotriz. `https://ellibero.cl/opinion/`

tomas-flores-la-explosion-del-parque-automotriz/, 2019. [Online; accedido 22-Febrero-2020].

[13] Google. Angular. https://angular.io/, 2010. [Online; accedido 22-Febrero-2020].

[14] José Gutiérrez. 1 por cada 47 habitantes: Chile lidera la región en venta de autos nuevos. https://www.latercera.com/mtonline/noticia/chile-autos-nuevos-region/845481/, 2019. [Online; accedido 22-Febrero-2020].

[15] Jurek Kirakowski. The use of questionnaire methods for usability assessment. http://sumi.uxp.ie/about/sumipapp.html, 1994. [Online; accedido 28-Agosto-2020].

[16] Arnold Lund. Measuring usability with the use questionnaire. In *Usability and User Experience Newsletter of the STC Usability SIG*, volume 08, 01 2001.

[17] Pitoval. Rabbitmq. https://www.rabbitmq.com/, 2007. [Online; accedido 26-Enero-2020].

[18] Laukik Raut, Rajat Wakode, and Pravin Talmale. Overview on kanban methodology and its implementation. In *International Journal for Scientific Research Development*, volume 03, pages 2518–2521, 07 2015.

[19] Winston Royce. Managing the development of large software systems. In *Proceedings of IEEE WESCON 26*, pages 1–9, 2001.

[20] SensioLabs. Symfony. https://symfony.com/, 2005. [Online; accedido 22-Febrero-2020].

[21] SII. Patentes vehículos. https://www.portaltransparencia.cl/PortalPdT/pdtta/-/ta/AK002/OA/OANT/39238731, 2020. [Online; accedido 10-Diciembre-2019].

[22] SII. Tasación de vehículos. http://www.sii.cl/servicios_online/1049-2612.html, 2020. [Online; accedido 22-Febrero-2020].

[23] VendenosTuAuto. Vendenos tu auto. https://www.vendenostuauto.com/, 2017. [Online; accedido 26-Enero-2020].

Apéndices

Palabras de adaptación

Palabra original	Palabra adaptada
ALFA␣ROMEO	ALFA-ROMEO
ASTON␣MARTIN	ASTON-MARTIN
␣CABINA␣SIMPLE␣	␣S/C␣
␣C/D␣	␣D/C␣
␣CD␣	␣D/C␣
␣C/S	␣S/C␣
␣CS␣	␣S/C␣
␣DC␣	␣D/C␣
␣DCAB␣	␣D/C␣
␣DOBLE␣CABINA␣	␣D/C␣
DS␣AUTOMOBILES	CITROEN
GAC␣GONOW	GAC-GONOW
GRAND␣	GRAND-
GREAT␣WALL	GREAT-WALL
HAVAL␣	␣GREAT-WALL␣HAVAL␣
KIA␣MOTORS	KIA
LAND␣ROVER	LAND-ROVER
MERCEDES␣BENZ	MERCEDES-BENZ
PICK␣UP␣	PICK-UP␣
ROLLS␣ROYCE	ROLLS-ROYCE
␣SC␣	␣S/C␣
SIN␣VERSION	SIN-VERSION
ZX␣	ZX-Auto␣

Tabla 6.1: Palabras de adaptación sin marca asociada.

Palabra original	Palabra adaptadaa
IA␣	␣IA␣
I␣	␣I␣
D␣	␣D␣
XDRIVE	XDRIVE␣
SDRIVE	SDRIVE␣
C␣I	CI
X␣I	XI
0E␣	0␣E
114␣I	114I
␣114␣	␣114I␣

Tabla 6.2: Palabras de adaptación de la marca BMW.

Palabra original	Palabra adaptada
␣S␣	␣S-

Tabla 6.3: Palabras de adaptación de la marca Changan.

Palabra original	Palabra adaptada
ARRIZO␣	ARRIZO-
TIGGO␣2	TIGGO-2
TIGGO␣3	TIGGO-3
TIGGO␣4	TIGGO-4
TIGGO␣7	TIGGO-7

Tabla 6.4: Palabras de adaptación de la marca Chery.

Palabra original	Palabra adaptada
CAPTIVA␣II	CAPTIVA-II
CAPTIVA␣III	CAPTIVA-III
CAPTIVA␣IV	CAPTIVA-IV
CAPTIVA␣5	CAPTIVA-5
CAPTIVA␣6	CAPTIVA-6
CORSA␣II	CORSA-II
CORSA␣III	CORSA-III
CRUZE␣II	CRUZE-II
CRUZE␣III	CRUZE-III
OPTRA␣II	OPTRA-II
S␣10	S10
S-10	S10
ASTRO␣VAN	ASTRO-VAN
AVEO␣II	AVEO-II
AVEO␣III	AVEO-III

Tabla 6.5: Palabras de adaptación de la marca Chevrolet.

Palabra original	Palabra adaptada
TOWN␣COUNTRY	TOWNCOUNTRY

Tabla 6.6: Palabras de adaptación de la marca Chrysler.

Palabra original	Palabra adaptada
␣PICASSO	-PICASSO
␣CACTUS	-CACTUS

Tabla 6.7: Palabras de adaptación de la marca Citroën.

Palabra original	Palabra adaptada
CARGO␣	CARGO-

Tabla 6.8: Palabras de adaptación de la marca DFSK.

Palabra original	Palabra adaptada
DF␣	DF-
REFRI␣	REFRI

Tabla 6.9: Palabras de adaptación de la marca Dongfeng.

Palabra original	Palabra adaptada
GRANDE␣	GRANDE-
500C	500␣C␣
500L	500␣L␣
500␣X	500X
RAM␣700␣	RAM700␣

Tabla 6.10: Palabras de adaptación de la marca Fiat.

Palabra original	Palabra adaptada
NEW-FIESTA	FIESTA

Tabla 6.11: Palabras de adaptación de la marca Ford.

Palabra original	Palabra adaptada
HAVAL␣	HAVAL-
VOLEEX␣	VOLEEX-
WINGLE␣5	WINGLE-5
WINGLE␣6	WINGLE-6
WINGLE5	WINGLE-5
WINGLE6	WINGLE-6

Tabla 6.12: Palabras de adaptación de la marca Great Wall.

Palabra original	Palabra adaptada
SANTA␣FE	SANTA-FE
PORTER␣HR	PORTER-HR

Tabla 6.13: Palabras de adaptación de la marca Hyundai.

Palabra original	Palabra adaptada
A␣137	A137
AO	A0

Tabla 6.14: Palabras de adaptación de la marca Jac.

Palabra original	Palabra adaptada
XK␣R	XKR
XJ␣R	XJR

Tabla 6.15: Palabras de adaptación de la marca Jaguar.

Palabra original	Palabra adaptada
RIO␣4	RIO-4
RIO␣3	RIO-3
RIO␣II	RIO-II
RIO␣5	RIO-5
CERATO␣5␣	CERATO-5␣
RIO␣JB	RIO-JB
RIO3	RIO-3
RIO4	RIO-4
RIO5	RIO-5
RIOII	RIO-II
GRAND-SPORTAGE	SPORTAGE␣GRAND

Tabla 6.16: Palabras de adaptación de la marca Kia.

Palabra original	Palabra adaptada
DISCOVERY␣3␣	DISCOVERY-3␣
DISCOVERY␣4␣	DISCOVERY-4␣
FREELANDER␣2␣	FREELANDER-2␣
RANGE␣ROVER	RANGE-ROVER
RANGE-ROVER␣EVOQUE	RANGE-ROVER-EVOQUE
FREE␣LANDER	FREELANDER

Tabla 6.17: Palabras de adaptación de la marca Land Rover.

Palabra original	Palabra adaptada
MG3	3
MG5	5
MG6	6

Tabla 6.18: Palabras de adaptación de la marca MG.

Palabra original	Palabra adaptada
XUV␣	XUV
PICK␣UP	PICKUP

Tabla 6.19: Palabras de adaptación de la marca Mahindra.

Palabra original	Palabra adaptada
MAZDA3	3
MAZDA2	2
CX␣	CX-
BT␣	BT-
MAZDA5	5
MAZDA6	6

Tabla 6.20: Palabras de adaptación de la marca Mazda.

Palabra original	Palabra adaptada
L␣200	L-200

Tabla 6.21: Palabras de adaptación de la marca Mitsubishi.

Palabra original	Palabra adaptada
X␣TRAIL	X-TRAIL
SENTRA␣II	SENTRA-II

Tabla 6.22: Palabras de adaptación de la marca Nissan.

Palabra original	Palabra adaptada
␣EHDI␣	␣E-HDI␣
␣GT␣LINE␣	␣GT-LINE␣
␣X␣LINE␣	␣X-LINE␣

Tabla 6.23: Palabras de adaptación de la marca Peugeot.

Palabra original	Palabra adaptada
A.SPORTS	ACTYON-SPORTS
ACTYON␣SPORT	ACTYON-SPORTS

Tabla 6.24: Palabras de adaptación de la marca Ssangyong.

Palabra original	Palabra adaptada
S␣CROSS	S-CROSS

Tabla 6.25: Palabras de adaptación de la marca Suzuki.

Palabra original	Palabra adaptada
␣CRUISER	-CRUISER
RAV␣4	RAV-4
4␣RUNNER	4-RUNNER

Tabla 6.26: Palabras de adaptación de la marca Toyota.

Palabra original	Palabra adaptada
␣XC␣	␣XC
␣S␣	␣S
␣V␣	␣V
␣C␣	␣C

Tabla 6.27: Palabras de adaptación de la marca Volvo.

Palabra original	Palabra adaptada
␣C␣	␣C
␣B␣	␣B
␣A␣	␣A
␣E␣	␣E
␣G␣	␣G
␣GLA␣	␣GLA
␣GLC␣	␣GLC
␣GLE␣	␣GLE
␣GLK␣	␣GLK
␣SLC␣	␣SLC
␣GL␣	␣GL
␣R␣	␣R
␣S␣	␣S
␣V␣CLASS␣	␣V-CLASS␣
BLUE␣EFFICIENCY	BLUEEFFICIENCY
4␣MATIC	4MATIC
␣CLA␣	␣CLA
␣CL␣	␣CL
␣CLK␣	␣CLK
␣CLS␣	␣CLS
␣CLC␣	␣CLC
␣G␣	␣G
␣GLC␣	␣GLC
␣GLA␣	␣GLA
␣GL␣	␣GL
␣GLE␣	␣GLE
␣GLK␣	␣GLK
␣GLS␣	␣GLS
␣MB␣	␣MB
␣ML␣	␣ML
BLUE␣TEC	BLUETEC
␣GLC␣	␣GLC
␣SLC␣	␣SLC
␣SL␣	␣SL
␣SLK␣	␣SLK
␣X␣	␣X

Tabla 6.28: Palabras de adaptación de la marca Mercedes Benz.

Sinónimos

Palabra	Significado	Campo asociado
BENCINA	BENCINA	Combustible
DIE	DIESEL	Combustible
DIESEL	DIESEL	Combustible
DSL	DIESEL	Combustible
GASOLINA	BENCINA	Combustible
HDI	DIESEL	Combustible
HRDI	DIESEL	Combustible
PETROLEO	DIESEL	Combustible
PETROLERA	DIESEL	Combustible
TD	DIESEL	Combustible
TDI	DIESEL	Combustible
2WD	4x2	Tracción
4MOTION	4X4	Tracción
4WD	4X4	Tracción
4X2	4X2	Tracción
4X4	4X4	Tracción
AWD	4X4	Tracción
FWD	4X2	Tracción
QUATTRO	4X4	Tracción
RWD	4X2	Tracción
AT	AUTOMÁTICA	Transmisión
AUT	AUTOMÁTICA	Transmisión
AUTO	AUTOMÁTICA	Transmisión
AUTOMATICO	AUTOMÁTICA	Transmisión
CVT	AUTOMÁTICA	Transmisión
DSG	AUTOMÁTICA	Transmisión
MANUAL	MANUAL	Transmisión
MEC	MANUAL	Transmisión
MECANICA	MANUAL	Transmisión
MECANICO	MANUAL	Transmisión
MT	MANUAL	Transmisión

Tabla 6.29: Sinónimos.