Modelo genérico de web scraping que facilite la integración, visualización y normalización de datos inmobiliarios en EE. UU.

Daniel Felipe Bravo Calvache
Luis Fernando Gómez Gómez

Director: PhD. Carlos Alberto Cobos Lozada

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Sistemas
Grupo de I+D en Tecnologías de la Información (GTI)
Área de interés en Gestión de la Información
Popayán, diciembre de 2021

RESUMEN

El sector inmobiliario requiere de un constante monitoreo de la información de inmuebles o propiedades que se encuentran localizados en distintas y extensas áreas geográficas, por ejemplo, identificar propiedades bien ubicadas que tienen deudas en impuestos o el pago de sus hipotecas y donde su compra, mejora y posterior venta deje un buen margen de utilidad. Este proceso es realizado actualmente por los agentes inmobiliarios de manera manual e implica una considerable inversión de tiempo y esfuerzo.

Una de las herramientas más útiles para abordar problemas que implican recolección continua de información es el web scraping, y es de especial interés en el sector inmobiliario de Estados Unidos, ya que allí muchos de los condados ofrecen la información de los inmuebles a través de páginas web y no ofrecen servicios o APIs de consulta.

En el presente trabajo se logró plantear un modelo genérico de información, que permite automatizar el proceso de recolección de información inmobiliaria en páginas de Real Estate por medio de un software que hace uso de web scraping para extraer la información de sitios web en diferentes ciudades de los Estados Unidos. También se desarrolló una aplicación web que implementa una arquitectura de microservicios que permite al usuario seleccionar propiedades inmobiliarias (parcelas, lotes, casas, apartamentos, etc.) a través de un mapa y luego ver la información general de dicha propiedad, junto con información de sus hipotecas, escrituras e impuestos. Además, permite que un usuario con conocimientos en HTML pueda agregar una nueva ciudad a la aplicación definiendo las fuentes de datos para esta nueva ciudad sin tener que programar o modificar la programación actual de la aplicación, en lugar de esto, configura la extracción de los datos haciendo uso de un nuevo lenguaje (propuesto en este trabajo) que le permite definir las instrucciones para llegar a las páginas de interés y extraer los datos requeridos.

La aplicación web se presentó a usuarios interesados en su uso y posible comercialización y además de evaluarla positivamente, dieron recomendaciones y sugerencias valiosas, algunas de las cuales se incluyeron en la aplicación y otras se dejaron como trabajo futuro. Los resultados obtenidos muestran el gran potencial de las técnicas de web scraping para buscar y extraer datos inmobiliarios, pero también de otras áreas de aplicación.

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO 1

1 INTRODUCCIÓN

1.1 PLANTEAMIENTO DEL PROBLEMA

El sector inmobiliario es uno de los sectores con mejor evolución en los precios y valorización del mercado desde el año 2012 [1]. Por eso es natural que en países con las economías más sólidas exista una considerable cantidad de compañías o empresas dedicadas enteramente a la negociación de bienes inmuebles como casas, apartamentos, tiempos compartidos o parcelas [2]. El incremento de personas interesadas en adquirir bienes habitables, así como el ascenso económico de los últimos años ha propiciado que se pueda generar una fuerte competencia entre compañías y compradores por adquirir uno o varios bienes habitables, como lo sugiere Blazheski en el Observatorio Económico ESTADOS UNIDOS. BBVA [3]. Razón por la cual, descubrir y localizar ofertas favorables es una tarea que se debe realizar rápidamente para lograr una adquisición antes que la competencia.

También cabe mencionar que en los últimos años durante los períodos de emergencias económicas que han afectado a diferentes países (por ejemplo, la crisis hipotecaria en Colombia de 1998-99, la crisis financiera mundial de 2008 y la crisis sanitaria por COVID 2020), se han reducido drásticamente los ingresos de muchas familias y con ello la capacidad de pagar sus hipotecas e impuestos, lo cual impulsa a la venta de sus inmuebles antes de llegar a ser embargados o desalojados [4]. Por otro lado, las empresas del sector inmobiliario tienen que estar pendientes continuamente de las ofertas de bienes inmuebles que aparecen a diario en el mercado para aprovechar las mejores y definir muy bien qué hacer con los bienes adquiridos, ya que en épocas de recesión se pueden congelar los inventarios y con ello tener graves problemas de liquidez que los pueden llevar a la quiebra [4]. En este sentido, la búsqueda de ofertas se hace más compleja, ya que se deben buscar continuamente bienes inmuebles en los cuales invertir, pero, además, mejorar los criterios de inversión buscando disminuir riesgos futuros.

Por otro lado, las técnicas de recolección de datos sobre las diferentes páginas, sitios y dominios web encontrados en internet, ha venido desarrollándose casi desde la popularización de la “Word Wide Web” y ha evolucionado a la par que las nuevas tecnologías que se incorporan en las páginas web modernas. Este conjunto de técnicas de recolección de datos en la web es conocido como Web Scraping [5]. A

pesar de que el web scraping no es un concepto nuevo, tampoco es un concepto muy conocido en el ámbito informático y empresarial, por lo que se encuentra una escasa cantidad de investigaciones y desarrollos teóricos formales entorno a esta técnica. Sin embargo, la falta de formalización en esta área no ha impedido que se hayan desarrollado tecnologías y herramientas que facilitan la creación de programas de Scraping. Estas herramientas incluyen desde frameworks y librerías para poder programar Scrapers hasta aplicaciones comerciales o APIs que vienen ya creadas y configuradas para su uso por parte de usuarios sin conocimientos técnicos, requiriendo poca intervención humana para comenzar el proceso de recolección de datos en dominios específicos de internet. Más adelante en este documento en la sección 2 se resumen las principales herramientas de scraping.

El sector inmobiliario tiene en el Web Scraping una herramienta muy importante para superar las limitaciones de tiempo y esfuerzo que implica el continuo monitoreo e identificación de bienes de su interés. Estos datos se relacionan con ofertas de bienes inmuebles publicados en innumerables sitios web de compra y venta de bienes raíces y portales gubernamentales de diferentes condados en EE. UU, en los cuales día a día se actualiza la información de su base de datos. Según la oficina de censo de Estados Unidos de 2014 a 2018 existían 119.730.128 hogares en todo el país [6], cifra dentro de la cual hay viviendas sujetas a conflictos hipotecarios, por lo que verificar si una vivienda pertenece o no al conjunto de bienes inmuebles en deuda se convierte en una cifra difícil de alcanzar con trabajo manual y poco personal.

Las compañías inmobiliarias en Estados Unidos están día a día en el proceso de búsqueda y selección manual de las mejores ofertas de bienes raíces, estas búsquedas se componen de múltiples datos contenidos en páginas gubernamentales de los diferentes condados del país. Las páginas de cada condado (similar a municipio en Colombia) cuentan con datos estructurados de diferente manera a los demás, como se puede evidenciar al analizar la estructura HTML de las páginas inmobiliarias de los condados de Miami-dade [7] y Dallas [8], lo que hace difícil la extracción de estos datos además de demandar mucho tiempo y esfuerzo. Algunas de estas compañías investigan casa por casa las mejores opciones de compra de propiedades en diferentes sitios, donde se tienen que identificar dentro de toda la estructura de la página web datos de relevancia como impuestos, escrituras e hipotecas que permitan tomar una decisión sobre distintas ofertas de adquisición para presentarla al propietario y posiblemente concretar una compra.

Uno de los problemas más grandes del Web Scraping es la heterogeneidad en las estructuras de las páginas web, debido a que cada página o dominio web está construido de manera diferente y cambian su estructura con el paso del tiempo. Razón por la cual un programa de scraping que fue programado para extraer un fragmento de datos en una región específica de una página web, muy seguramente

no encontrará esa información en la misma región de una página web distinta, debido a que ésta contiene una estructura HTML diferente.

Por esta razón, el mayor reto del web scraping en el sector inmobiliario es: ¿Qué características debe contemplar un modelo generalizado de información inmobiliaria, que le permita a un scraper automatizar la forma de extraer datos específicos de interés, en la mayor cantidad de sitios web útiles para las compañías de bienes raíces? Considerando que para dicho sector es conveniente estar revisando constantemente los datos de diversos sitios o portales web donde se puedan encontrar ofertas y bienes listos para la venta.

Además de esta dificultad importante, el Scraper también podría encontrarse limitado por las medidas de seguridad y restricción que implementan las páginas web para evitar la extracción y explotación de sus datos por parte de programas automatizados con propósitos maliciosos, como son los Bots. Entre estas limitaciones, las más conocidas son las relacionadas con la restricción o bloqueo de direcciones IP y la implementación de Captchas anti-bot [9] que se encuentran actualmente en algunos portales y páginas web. Afortunadamente para este tipo de inconvenientes también se han creado algunas herramientas que ayudan a superarlos [10], aunque la mayoría de estas herramientas son APIs de carácter comercial y pueden limitar el número de operaciones a realizar. El uso de este tipo de herramientas ha sido restringido solo en algunos países específicos como China, Turquía, Irak, Rusia, Bielorrusia, Corea del Norte y Turkmenistán.

Otro posible inconveniente en el proceso de recolección de datos por medio del web scraping, y muy alejado del aspecto técnico, es el aspecto legal, ya que muchas de las páginas web que contienen los datos de interés, también implementan una serie de políticas y reglas de uso de los datos que deben ser consideradas para no tener dificultades o implicaciones de tipo jurídico. Sin embargo, este aspecto se sale del alcance técnico del scraper y se debe acatar y respetar al momento de determinar los sitios web a los cuales se les pretende extraer los datos.

Buscando delimitar el alcance del trabajo desarrollado, esta propuesta se centró en la generación de un modelo de información que permita la extracción de datos e información inmobiliaria de distintas páginas web en ESTADOS UNIDOS. Además, el desarrollo de un primer prototipo que se base en dicho modelo, estructurado como un microservicio y evaluado por un grupo pequeño de usuarios.

1.2 APORTES DEL PROYECTO

Desde el ámbito investigativo la principal contribución es la generación de nuevo conocimiento para la integración de la tecnología de web scraping con el sector inmobiliario. Lo anterior, debido a que Scopus (buscador académico y científico que recopila los artículos más importantes de diferentes editoriales como ScienceDirect, IEEE, Springer y ACM) contiene una baja cantidad de artículos que relacionen estos dos temas y además en este buscador no se encontraron propuestas que permitan

solventar la problemática presentada (extraer automáticamente información relacionada con un inmueble desde diferentes fuentes y en diferentes ciudades). Este nuevo conocimiento se ve reflejado en la creación de un lenguaje genérico para lograr acceder y extraer información de las páginas web de bienes inmuebles basado en el framework de web scraping Selenium. Este lenguaje es soportado por dos procesadores de instrucciones: uno enfocado en la interacción con las páginas web, para acceder a las fuentes de la información, y otro enfocado en la extracción de los datos específicos que hay en los elementos de las páginas web. Este lenguaje es interpretado por 3 programas scrapers, cada uno diseñado para extraer un tipo de información específico, como lo son, la información general de las propiedades, sus impuestos y sus escrituras. Esto es la base de lo que podría llegar a ser un lenguaje mucho más robusto que permita y facilite la extracción de información de cualquier página web.

Otro aporte investigativo del trabajo consiste en la creación de un modelo genérico de datos para el sector inmobiliario que permita resumir los atributos más comunes y útiles que presentan los distintos sitios web sobre información de bienes inmuebles, modelo que está acoplado al funcionamiento de los tres scrapers anteriormente mencionados.

En el ámbito de innovación, se desarrolló un prototipo software que se espera tenga utilidad práctica para las tareas diarias de elección de propiedades a comprar por parte de las diferentes compañías inmobiliarias de Estados Unidos, gracias a la disminución de la dificultad en la tarea y la reducción del tiempo de esta en relación con el trabajo manual que se realiza actualmente. Se espera que el uso del software propuesto genere un beneficio directo para las empresas, sus empleados y motive una mayor dinamización del sector inmobiliario.

En el ámbito de desarrollo, el grupo de investigación empieza su experiencia en el desarrollo de aplicaciones orientadas a microservicios en la nube integrando el uso de diferentes microservicios de web scraping. El presente trabajo espera aportar experiencia a los miembros del grupo y de la Facultad de Ingeniería Electrónica y Telecomunicaciones en este tema en particular. Específicamente se desarrolló una aplicación web que implementa el sistema de mapas de Google Maps para poder localizar y seleccionar cualquier inmueble de los condados de ESTADOS UNIDOS. al que el sistema de web scraping le extrae la información presentada en los sitios web inmobiliarios configurados para una ciudad o condado específico. Esta aplicación se desarrolló con el framework Angular 10 para el Frontend, el framework .NET Core para el Backend y el framework Selenium de Python para los scrapers. Además, el motor de base de datos que implementa el modelo de datos genérico de información escogido fue Microsoft SQL Server.

1.3 OBJETIVOS

A continuación, se presentan los objetivos tal y como fueron aprobados por el consejo de facultad de la Facultad de Ingeniería Electrónica y Comunicaciones al inicio del proyecto y la aprobación de una modificación posterior al tercer objetivo específico.

1.3.1 Objetivo General

Proponer un modelo genérico de información para web scraping que facilite la integración, visualización y normalización de datos inmobiliarios en los diferentes condados de ESTADOS UNIDOS.

1.3.2 Objetivos Específicos

- Definir un modelo genérico de información para web scraping que facilite la integración, visualización y normalización continua de datos de bienes inmobiliarios relacionados con el estado de las hipotecas, los impuestos y la escrituración de la propiedad, disponible en sistemas web del gobierno condal o local de los estados de ESTADOS UNIDOS., usando el patrón de investigación iterativa de Pratt para facilitar el proceso de búsqueda, compra y venta que realizan los agentes inmobiliarios.
- Desarrollar una aplicación basada en el modelo previamente propuesto, una arquitectura orientada a microservicios sobre .NET Core y una interfaz web de un solo documento usando SCRUM como metodología de desarrollo para facilitar la integración, visualización y normalización continua de la información inmobiliaria de EE. UU.
- Definir el nivel de satisfacción de la aplicación web desarrollada basado en una muestra de usuarios de la empresa ATIX DIGITAL S.A.S., mediante la adaptación y uso de la encuesta de “Satisfacción del cliente (producto)” proporcionada por encuestafacil.com usando el índice CSAT [11], conforme se expresa en el apartado 9.1.2 de la norma ISO 9001:2015 [12].

1.4 RESULTADOS OBTENIDOS

A continuación, se resumen los principales resultados del presente trabajo de grado:

1. **Monografía de trabajo de grado**: Se refiere al presente documento en el cual se presenta el estado del arte de las técnicas y tecnologías utilizadas en el web scraping como Selenium, BeautifulSoup y Scrapy, las cuales se consideraron para la creación del software que permita la automatización de la recolección de información inmobiliaria de las páginas web de algunas ciudades de EE. UU. Este documento también contiene el modelo genérico de datos para soportar la búsqueda de información inmobiliaria en condados de Estados Unidos, y, por último, presenta la aplicación web desarrollada junto con los resultados de la evaluación de satisfacción de los usuarios de ATIX DIGITAL S.A.S.

2. **Modelo de datos genérico de información inmobiliaria en EE. UU:** Se refiere al modelo relacional diseñado a partir del análisis de los datos y la información que se puede obtener de las diferentes páginas web de algunos condados de Estados Unidos referente al estado de bienes inmuebles.
3. **Aplicación web para realización de web scraping inmobiliario**, de la cual se destacan dos productos principales, a saber:
   - ***Código fuente de aplicación web scraping desarrollada*:** Hace referencia al código fuente con el que se desarrollaron los componentes de la aplicación, entre los cuales se incluyen TypeScript para el desarrollo del Frontend con Angular, C# para el desarrollo del Backend con .NET Core, y Python para el desarrollo de los scrapers con el framework Selenium.
   - ***Documentación del código de la aplicación web y del web scraping desarrollado*:** Hace referencia a la documentación realizada sobre el código y los componentes de la aplicación desarrollada. Esta documentación fue generada con ayuda de la herramienta Doxygen para la generación automática de documentación.

1.5 ESTRUCTURA DE LA MONOGRAFÍA

A continuación, se describe de manera general el contenido y organización de la presente monografía:

CAPITULO 1: INTRODUCCIÓN: Hace referencia al presente capitulo que introduce el tema de investigación, presenta la pregunta de investigación que originó el trabajo, los aportes al problema, también los objetivos (general y específicos) definidos para el proyecto, un breve resumen de los resultados obtenidos y finalmente la organización de la monografía.

CAPITULO 2: CONTEXTO TEÓRICO Y ESTADO DEL ARTE: En este capítulo se presentan algunos enfoques y herramientas que permiten la realización de web scraping, entre los cuales se encuentran Frameworks, APIs y las dificultades o impedimentos potenciales de su uso.

CAPITULO 3: MODELO GENÉRICO DE INFORMACIÓN PARA WEB SCRAPING: En este capítulo se presenta el modelo genérico para la obtención de información inmobiliaria extraída de varias páginas web con la implementación de un software que hace uso de web scraping para extraerla.

CAPITULO 4: APLICACIÓN WEB: En este capítulo se presenta el diseño detallado del sistema desarrollado describiendo la arquitectura de cada componente junto a los aspectos más importantes de su implementación.

CAPITULO 5: RESULTADOS DE EVALUACIÓN DE SATISFACCIÓN: En este capítulo se presenta la evaluación de la satisfacción de los usuarios por la aplicación web desarrollada usando como fuente de datos algunas páginas web del estado de la Florida (EE. UU).

CAPITULO 6: CONCLUSIONES Y TRABAJOS FUTUROS: En este capítulo se presentan las conclusiones obtenidas al finalizar el trabajo de grado e ideas que el grupo de investigación espera realizar en un trabajo futuro.

CAPITULO 7: BIBLIOGRAFIA: Este último capítulo contiene las referencias bibliográficas de los artículos y libros consultados para la realización del proyecto.

CAPÍTULO 2

2 CONTEXTO TEÓRICO Y ESTADO DEL ARTE

A continuación, se presenta un resumen de información relacionada con las categorías de los métodos para realizar web scraping, sus enfoques y algunas de las herramientas disponibles actualmente que permitan o faciliten la programación de crawlers (Bots) y scrapers para la obtención de datos de páginas web dispuestas en la red. Luego se resumen los trabajos previos más representativos de web scraping en el sector inmobiliario y otros relacionados que aportaron algunas ideas para el desarrollo de la investigación.

2.1 CONTEXTO TEÓRICO

El web scraping también conocido como raspado web, recolección web o extracción de datos web es un conjunto de métodos o técnicas que permiten la extracción automática de datos de diferentes sitios web. En la literatura se ha tratado de identificar diferentes factores que permitan clasificar los métodos de extracción de datos en páginas web. En 2016, Schulz et al. [13] presentan una clasificación de los métodos de web scraping de acuerdo con 2 categorías:

1. **Wrapper types**: son métodos de web scraping basados en cadenas como las expresiones regulares o en técnicas más sofisticadas como el uso de XPath, selectores CSS, XQuery u OXPath, que permiten aprovechar la estructura de árbol de un documento HTML para obtener los datos deseados.

2. **Wrapper scope**: dividido en cuatro niveles de acuerdo con el alcance del proceso de extracción de datos:

    a) **A nivel de registro**: en él se pueden extraer registros de un documento o listas de registros de campos.

    b) **A nivel de página**: que permite extraer datos de una sola página.

    c) **A nivel de sitio**: que son capaces de extraer la estructura de la página web, incluidas las subpáginas y su estructura de enlace.

    d) **A nivel de dominio**: Estos métodos no necesariamente se basan en la estructura HTML del documento y pueden ser más generales en la extracción de datos. En este nivel se incluyen los métodos basados en ontologías y

entrenamiento. Un ejemplo de esto es ODE [14], que provee un método automático para obtener la ontología de una página web y de esta forma hacer la extracción sin mayor intervención del usuario.

También en 2016, Varlamov y Turdakov [15] proponen otra clasificación de los métodos de web scraping, pero basándose en cuán extensos son los datos que se van a extraer, este trabajo también propone 2 clases:

1. **Métodos aplicables a páginas web individuales**, la cual se divide en 3 subclases, a saber:
    a) **Desarrollo manual de wrappers**: requieren intervención intensiva humana y en otros casos la programación en un entorno de desarrollo del wrapper.
    b) **Métodos supervisados**: dado un conjunto de páginas etiquetadas el algoritmo se entrena para luego ser capaz de reconocer cuáles datos son los que el usuario requiere extraer de una página web. Acá se incluyen los métodos semi supervisados donde el conjunto de páginas de entrenamiento no está totalmente etiquetado.
    c) **Métodos no supervisados**: son métodos que no requieren de intervención humana y se basan en el análisis de las estructuras repetitivas de las páginas para extraer datos comunes que se encuentran en éstas.
2. **Métodos aplicables a un dominio de aplicación completo**, los cuales se dividen en 2 subclases:
    a) **Métodos de dominio específico**: son métodos que usan el desarrollo manual, los métodos supervisados o los métodos no supervisados, pero en un dominio de aplicación específico.
    b) **Métodos de dominio agnóstico**: Son métodos independientes del dominio de la aplicación. Estos se dividen en:
        i. **Métodos ontológicos**: que a través de ontologías internas almacenan datos sobre el dominio al cual se le quiere realizar scraping.
        ii. **Métodos estadísticos**: en donde se utiliza la elección aleatoria para identificar los datos relevantes en cada página web.

2.1.1 Enfoques de los métodos de Web Scraping

Diouf et al, en 2019 [5] proponen los siguientes enfoques de web scraping basados en la estructura de las páginas web:

1. **Enfoque Mimicry**: El scraper se configura con reglas específicas para encontrar los datos dentro de una estructura de documento web. Sin embargo, su funcionamiento está limitado a páginas o documentos web con una misma estructura (como en un mismo dominio web) por lo que es un enfoque poco adaptativo y con una alta propensión a la intervención humana.

2. **Enfoque de medición de peso**: Este enfoque toma como base de análisis la longitud de las palabras que se encuentran en cada una de las ramas del árbol del DOM (Document Object Model) de una página web. A partir de estas mediciones, el algoritmo deduce un punto principal, a partir del cual comienza a extraer los datos de los nodos hijos. Puede adaptarse automáticamente a los cambios estéticos en la presentación. Sin embargo, los datos obtenidos de este proceso pueden estar acompañados de una cantidad considerable de "ruido".
3. **Enfoque diferencial**: En este enfoque se asume que dos páginas de un mismo dominio web tienen una estructura similar al tener elementos muy comunes entre ellas, como lo son barras de navegación, columnas o pies de página (footers), y se diferenciarán únicamente por el contenido propio de cada una de las páginas. El algoritmo entonces superpone las dos páginas haciendo que queden resaltados únicamente los datos que son diferentes entre ellas.
4. **Enfoque de aprendizaje de máquina**: Estos algoritmos corresponden a los previamente mencionados como métodos supervisados. En el entrenamiento el algoritmo busca indicadores estadísticos o geográficos que le permitan definir dónde se encuentra el contenido buscado. Este enfoque no está excluido de ser propenso a un alto grado de intervención humana, al menos en el proceso de aprendizaje o reentrenamiento.

2.1.2 Herramientas para Web Scraping

Las herramientas se pueden clasificar en Frameworks, APIs, Aplicaciones comerciales y Add-ons, entre las que se destacan las siguientes:

1. Frameworks:
   a. **Scrapy:** es un framework de alto nivel para desarrollar rápidamente software para llevar a cabo web crawling y web scraping [10].
   b. **BeautifulSoup:** Es una librería de Python creada para extraer datos de documentos HTML y XML. Se diferencia de Scrapy principalmente en que el BeautifulSoup se enfoca más en la extracción de datos, mientras que Scrapy provee funciones especializadas para realizar crawling además de scraping, por lo que hacer crawling en BeautifulSoup requiere más intervención en la programación [16].
   c. **Jaunt:** Es una librería de Java para realizar web scraping, automatización web y consultas sobre datos JSON. Permite acceder al DOM de las páginas web y controlar peticiones HTTP. Desafortunadamente no cuenta con soporte para JavaScript [17].
   d. **Jauntium:** Al igual que Jaunt, es una librería de Java para realizar web scraping. Posee soporte para páginas web con tecnología JavaScript [18].
   e. **Html agility pack:** es un analizador de html perteneciente a la biblioteca de códigos .NET. Permite leer y escribir en los elementos que provee el DOM para una página web especificada y es compatible con XPATH y XSLT [19].

f. **MechanicalSoup:** Es una librería de Python para automatizar la interacción con las páginas web. Puede automáticamente almacenar y enviar Cookies, seguir links y redirecciones, e incluso enviar formularios [20].
g. **Selenium:** Selenium es un conjunto de herramientas desarrolladas para la automatización de la navegación en visores web (navegadores web). Es ampliamente usado para realizar pruebas automatizadas en aplicaciones basadas en la web. Una de las grandes ventajas de este framework es que, al contar con un lenguaje específico de dominio, las pruebas pueden ser escritas en una variedad de lenguajes de programación, entre los cuales se encuentran Java y Python. Además, es capaz de soportar JavaScript, por lo que se pueden tratar problemas como la carga dinámica de información y la interacción con elementos de las páginas web [21].

2. APIs:

a) **webhose.io:** Es una aplicación web que automatiza el proceso de web scraping ofreciendo a sus usuarios los datos limpios, estructurados y organizados de millones de sitios web. Esta plataforma proporciona APIs para diferentes segmentos de mercado, entre los cuales están, noticias, blogs, debates en línea, reseñas y dark web [22].

b) **Scrapestack API:** Es una interfaz REST API para la realización de web scraping sin tener la necesidad de lidiar programáticamente con problemas relacionados con la geolocalización, bloqueos de IP e incluso Captchas [23].

3. Aplicaciones comerciales:

a) **import.io:** es una plataforma web que proporciona miles de datos limpios de diversos sitios web, facilita el proceso de extracción de datos ofreciendo diversas utilidades, una de ellas es apuntar y hacer clic, la cual permite al usuario seleccionar los datos que necesite obtener de una página web. En su edición gratuita permite la extracción de datos mediante XPath, selectores CSS, RegEx y JS [24].

b) **Scrapinghub:** permite obtener datos a medida, le facilita al usuario el proceso de extracción. También proporciona herramientas cloud para clientes que necesiten realizar sus algoritmos de scraping en la nube [25].

c) **Parsehub:** es una herramienta de escritorio para realizar web scraping. Facilita la extracción de datos haciendo clic en los datos que se necesitan. Su versión es gratuita y ofrece funcionalidad de manera limitada [26].

d) **Agenty:** Además de permitir la extracción de datos con un clic, también permite extraer datos de forma ilimitada mediante scripts generados desde su API utilizando lenguajes de programación como C# y node.js [27].

4. Add-ons:

   a) WebScraper: Es una extensión para Google Chrome que permite hacer web scraping a diferentes páginas web de una manera simple. Soporta extracción de datos en páginas que cuenten con JavaScript y Ajax [28].

   b) DataScraper: Es una extensión para realizar web scraping mediante el navegador Firefox. Extrae datos de una página web mediante selectores jQuery que permiten configurar los datos que se desean extraer [29].

2.2 ESTADO DEL ARTE

En la literatura se encuentran diferentes trabajos que buscan resolver diferentes necesidades de recolección de datos que se presentan en el mercado inmobiliario. En 2017, Boeing y Waddell [30] utilizan web scraping para recolectar once millones de datos de la página web Craigslist para lograr explorar y comprender el mercado inmobiliario de alquileres en los Estados Unidos. En 2018, Kumkar et al. [31] muestran y comparan diferentes algoritmos de aprendizaje que se utilizan para el análisis de inversiones, seguros de propiedad, impuestos, entre otros, de acuerdo con una estimación de precios de los bienes con ciertas características. En 2019, Li et al. [32] presentan un proyecto de análisis de precios de viviendas de acuerdo con diferentes variables que se proponen, como lo son el transporte público, parques, hospitales, entre otras, mostrando cómo puede variar el valor de cada una de estas propiedades, por ejemplo, con respecto al número de edificios o distanciamiento en millas del hogar a éstas. También en 2019, Hong et al. [33] automatizan la extracción de datos de materiales ecológicos de diferentes páginas web y utilizan una ontología para su clasificación, con el fin de apoyar las certificaciones ecológicas en proyectos de construcción. Ahme. et al, en 2020 [34] recolectan datos de diferentes fuentes web y construyen un conjunto de reglas que ayudan a tomar una decisión de compra inmobiliaria basados en los precios recopilados de los bienes. También en 2020, Pasbola y Saxena [35] apoyan la decisión de adquisición de un hogar mediante un sistema de recomendación alimentado de diferentes datos obtenidos, algunos de éstos desde páginas web.

Además de los anteriores trabajos relacionados con el sector inmobiliario, también existen algunos otros en los que se ha utilizado la técnica del web scraping para extraer y analizar información útil para distintos sectores económicos y científicos. En 2018, Correa et al. [36] obtienen la valoración de los clientes sobre los restaurantes en Facebook por medio de web scraping para establecer el impacto que tiene el desempeño de los servicios de domicilios sobre estas valoraciones. También en 2018, Zhou et al. [37] proponen tres estrategias distintas de adquisición automatizada de datos del sector agrícola en China basadas en tecnologías de web crawler. En 2020, Pillai [38] plantea una recolección automatizada de información en portales de trabajos para poder determinar los perfiles y capacidades requeridas por el sector tecnológico en la India, y por otro lado, Sharma et al. [39] experimentan

con el desempeño de web crawlers de uno y múltiples hilos para indexar contenido web. También en 2020, Yang et al. [40] realizan recolección automatizada de propiedades de materiales en internet para el análisis del rendimiento energético de los edificios, y Lo et al. [41] recolectan datos mediante un web crawler para investigar si las ofertas de empleo de empresas taiwanesas que cotizan en la bolsa están asociadas a sus posteriores rendimientos bursátiles. En 2021, Zhang et al. [42] proponen un método para la adquisición viable y eficiente de datos sobre vertidos de aguas residuales haciendo uso de tecnologías de rastreo web, entre ellas web crawling y web scraping. Por otro lado, Stallone et al. [43] obtuvieron una muestra de información de 800 compañías identificadas en páginas de sitios web de ICO (Initial Coin Offerings) haciendo uso de un método de web scraping. También en 2021, Muehlethaler y Albert [44] utilizan el web scraping para extraer información de la página web de un distribuidor de ropa mayorista con la intención de facilitar los estudios de mercado sobre los tipos de fibras, su popularidad y tendencias.

Uno de los sectores donde se encontraron más trabajos relacionados para la extracción de datos mediante web scraping es el de la salud y la medicina. En 2018, Baskaran et al. [45] idean una técnica automatizada para extraer mensajes de varios sitios web de foros médicos con estructuras diferentes. En 2020, Santos et al. [46] presentan un conjunto de datos de producción académica relacionada con investigaciones sobre el virus del COVID-19 obtenido de diferentes bases de datos investigativas como Scopus y PubMed haciendo uso de técnicas de web scraping. También en 2020, Azibeiro Melchor et al. [47] implementan una técnica de web scraping para extraer continuamente información sobre historias clínicas, registros de enfermedades, exploración física, signos vitales, test microbiológicos, etc. de los pacientes contagiados con SARS-CoV2 que son registrados en la plataforma web de un hospital. En 2021, Schedlbauer et al. [48] logran extraer 544 anuncios de empleo de una portal alemán de empleos mediante un programa desarrollado en R para llevar a cabo web crawler y scraping, con la intención de identificar los requisitos más comunes que permitan estandarizar un perfil de trabajo para el área de la informática médica. En 2021, Catalani et al. [49] crean un software de web crawling y scraping para escanear continuamente de una lista de URLs de sitios web de psico-nautas y nuevas sustancias psicoactivas, para extraer datos como el nombre y la formula química de las nuevas sustancias.

Es preciso destacar que en la mayoría de los trabajos previos consultados hay dos factores comunes, el primero es que el web scraper se desarrolla específicamente para, y conociendo de antemano, la fuente de los datos en la cual se realizará el proceso de extracción, es decir, un mismo dominio web o un conjunto de páginas web muy específico. El segundo es que el enfoque investigativo de estos proyectos está más dirigido hacia el análisis de los datos recolectados que en el proceso de recolección en sí mismo, precisamente por saber de antemano que la fuente de los datos no representa una dificultad o un aspecto variable para sus aplicaciones.

Teniendo en cuenta el anterior contexto teórico y estado del arte, esta investigación se centró en el desarrollo de un scraper de dominio específico. Lo anterior debido a que el proceso de extracción se va a realizar sobre varios sitios web con datos relacionados con el sector inmobiliario, la definición del modelo de información no se realizó de forma automática y la instanciación de la solución para cada localidad requirió intervención humana.

CAPÍTULO 3

3 MODELO GENÉRICO DE INFORMACIÓN PARA WEB SCRAPING

La generación del modelo genérico implicó la realización de una serie de actividades que empezaron por la exploración y el análisis de un conjunto de páginas y portales web sobre información inmobiliaria en varias ciudades y condados de EE. UU, esto con la intención de recolectar la mayor cantidad de atributos útiles de la información que se puede obtener de esas páginas. Después de la recolección de estos atributos, y tras un proceso de análisis y categorización, se obtuvo la base fundamental sobre la cual se logró proponer el modelo genérico de información que abarca la gran mayoría de información relevante y común a la mayoría las páginas web de Real Estate (bienes raíces) y property appraiser (tasadores de propiedades) que se pudieron analizar. Este modelo, a su vez, es la base para la construcción de la aplicación web y el programa de web scraping que permitió la recolección automatizada de esta información.

3.1 RECOLECCIÓN DE ATRIBUTOS DE INFORMACIÓN

Para lograr dar forma a un modelo genérico de información que abarque la mayor cantidad de atributos de información comunes de bienes raíces en las ciudades y condados de Estados Unidos, se determinó realizar exploración sobre distintas páginas de property appraiser y Real Estate de diferentes condados del país. La elección de las ciudades, cuyas páginas web de Real Estate fueron inspeccionadas y analizadas, se basó en una investigación previa sobre cuáles eran las mejores ciudades en Estados Unidos para invertir en bienes inmuebles [50][51][52][53]. Desde luego, la selección de esta lista de ciudades factibles para invertir está influenciada por una serie de factores muy propensos a variar con el paso del tiempo, como los índices de crecimiento urbanos, la ubicación de la ciudad y las propiedades, y hasta la situación económica mundial y regional. Al momento de realizar la selección de estas ciudades se tomó como base la información encontrada en artículos y publicaciones de algunas páginas web de bienes inmuebles importantes en los Estados Unidos como lo son Zillow [54], investor junkie [55] y WalletHub [56].

Los estados de donde se seleccionaron las páginas web a ser analizadas se muestran en el mapa presentado en la **Figura 1**.

***Figura 1**. Estados seleccionados para analizar información inmobiliaria contenida en sus sitos web.*

El resultado de este primer proceso investigativo es una lista de URLs de páginas gubernamentales de Real Estate y de agentes inmobiliarios en donde se puede encontrar información referente a las propiedades y bienes inmuebles, sus características internas y externas, propietarios e información sobre impuestos y valorización.

La **Tabla 1** muestra algunas ciudades con sus respectivas páginas web de donde se puede extraer la información relacionada con Real Estate, impuestos y escrituras. La lista completa se puede encontrar en la Tabla 1 del Anexo A.

***Tabla 1**. Páginas de estudio y análisis de Web Scraping.*

CIUDAD	INFORMACIÓN GENERAL	IMPUESTOS	ESCRITURAS
Miami Dade, FL	https://www.miamidade.gov/Apps/PA/propertysearch/#/	https://miamidade.county-taxes.com	https://onlineservices.miami-dadeclerk.com/officialrecords/StandardSearch.aspx
Tampa, FL	https://gis.hcpafl.org/propertysearch/#/nav/Basic%20Search	https://hillsborough.county-taxes.com/public	https://pubrec6.hillsclerk.com/ORIPublicAccess
Fort Lauderdale, FL	https://web.bcpa.net/BcpaClient/#/Record-Search	https://broward.county-taxes.com/public	https://officialrecords.broward.org/AcclaimWeb

Una vez recolectadas las direcciones web de donde se puede extraer la información, se realizó un análisis visual de la información presentada en cada una de las páginas.

En la **Figura 2**, la **Figura 3**, la **Figura 4** y la **Figura 5** se pueden ver algunos ejemplos de la información presentada en las páginas web de property appraiser.

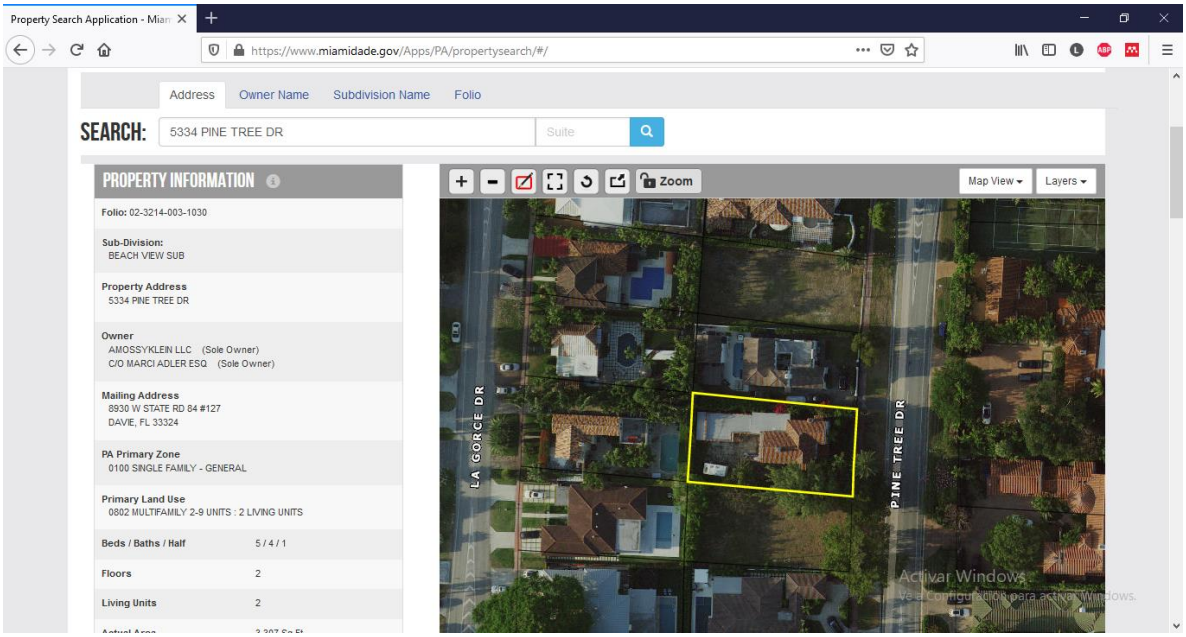

***Figura 2**. Miami Dade Property Search.*

***Figura 3**. Jacksonville Property Appraiser.*

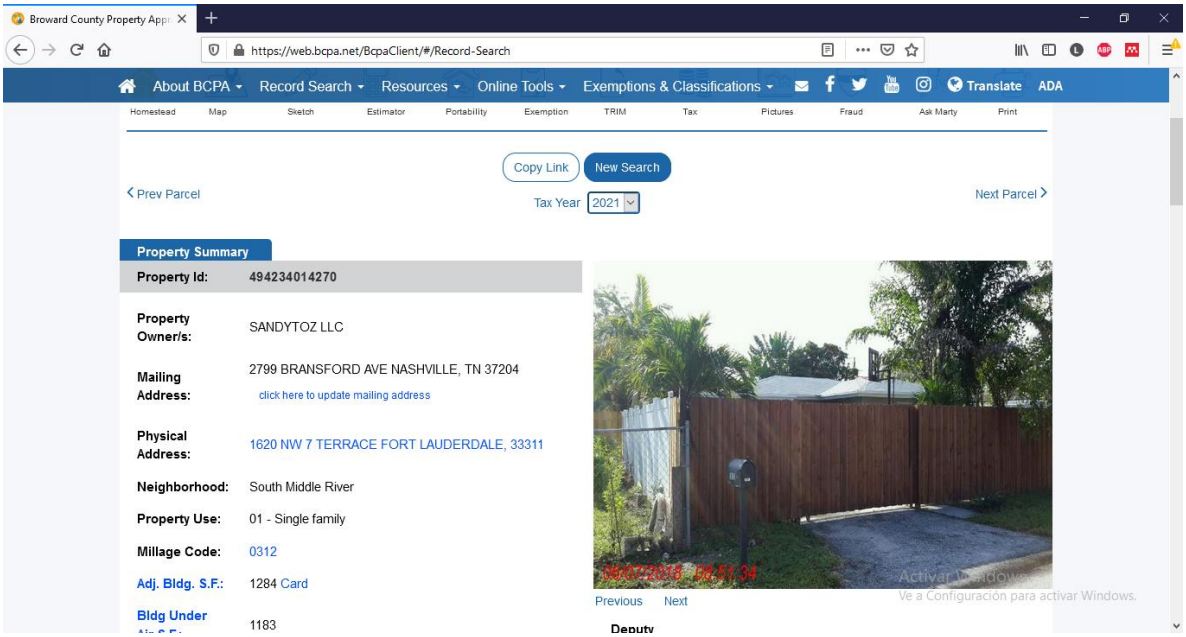

***Figura 4**. Fort Lauderdale.*

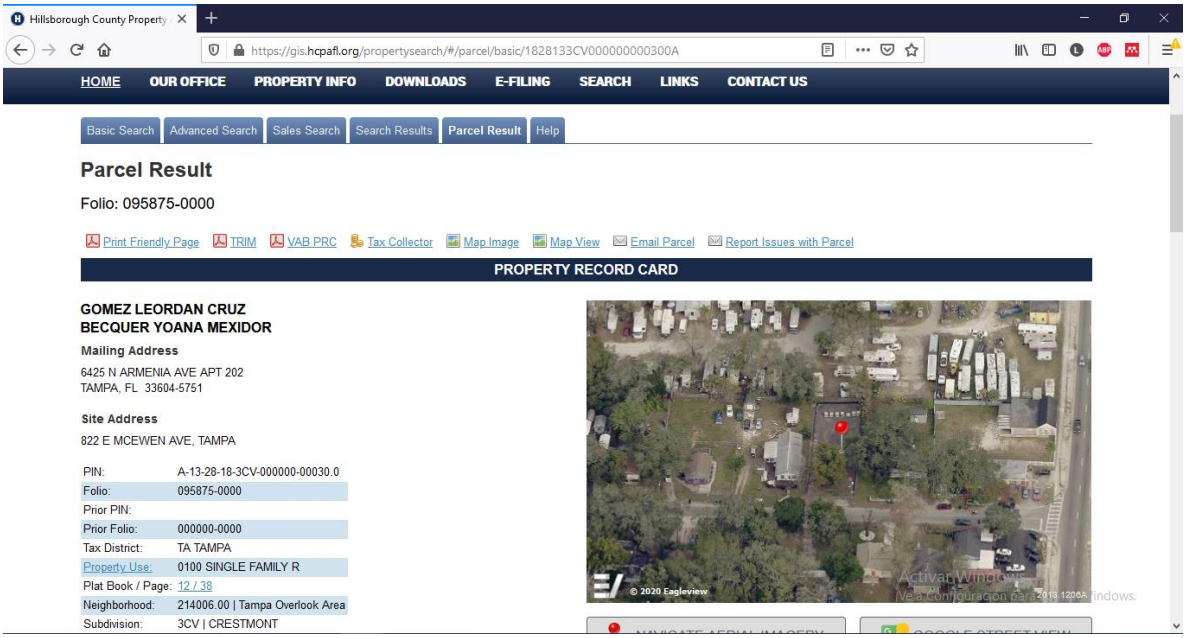

***Figura 5.** Tampa Property Appraiser.*

De las páginas revisadas, se observó que se muestra información referente a las características del terreno, las características de la estructura construida, datos del o los propietarios, y en algunos casos se muestra información relacionada con impuestos y la ubicación de la propiedad. Sin embargo, no en todas las páginas se muestran los datos agrupados de la misma forma, y en determinados casos, algunos de los atributos de información incluso pueden llegar a ser nombrados de manera diferente. También es preciso destacar que algunas de esas páginas presentan información más detallada y extensa, llegando a mostrar incluso el mapa

de la ubicación y los planos de la construcción o el terreno, mientras que en otras no se encuentra información tan detallada y específica.

Después de hacer un recorrido por distintos registros de propiedades, escogidos aleatoriamente, en cada una de las páginas web, se logró extraer los atributos más relevantes que pueden ser extraídos de cada página. La **Tabla 2** muestra el resultado de este proceso en tres ciudades. La lista completa se puede encontrar en la **Tabla 5** del **Anexo A**.

***Tabla 2**. Atributos extraídos en páginas web de información general.*

CIUDAD	ATRIBUTOS DE INFORMACIÓN GENERAL	
MIAMI-DADE	Folio Sub-divison Property address Owner Mailing address Pa primary zone (building type) Adjust area Lot size	Dormitorios Beds / baths / half Floors Living units Actual area Living area Year built
JACKSONVILLE	Owner name (sin label) Address (sin label) Primary site address Re # # Buildings Subdivision Rooms /units Heating type Air condition Gross area	Total area Year built Roof struct Roofing cover Bedrooms Baths Heating area Property use Zoning assessment Effective area
PALM BEACH	Location address Parcel control number Subdivision Owners Owner mailing address Total square feet Acres Number of units	Year built Bedrooms Full baths Half baths Roof structure Roof cover Property use Zoning

La información anteriormente obtenida fue fundamental para la consecución del objetivo de este trabajo, dado que supone la base de la información que se pretende representar con el modelo genérico de información, y que se obtendrá por medio de web scraping. No obstante, el hecho de que sea una parte importante de la información objetivo del proyecto implica superar una serie de situaciones y precondiciones para llegar a la información de cada registro específico para una

propiedad (casa, apartamento, parcela, lote o similar). La primera pregunta que debió ser respondida fue ¿qué datos o información específica se debe conocer de antemano para poder recuperar la información completa de una propiedad? Esta pregunta sólo pudo ser respondida haciendo uso de cada una de las páginas requeridas.

Cada una de las páginas de Real Estate y property appraiser cuenta con un buscador de folios o registros con diferentes filtros y campos de búsqueda que facilitan a los usuarios llegar a la información de las propiedades. La **Figura 6**, la **Figura 7**, la **Figura 8** y la **Figura 9** muestra los formularios de búsqueda de folios de algunas de las páginas a las que se les realizó web scraping. Es preciso destacar que los formularios tienen diferentes campos de búsqueda (unos generales y otros muy específicos) y su organización es diferente.

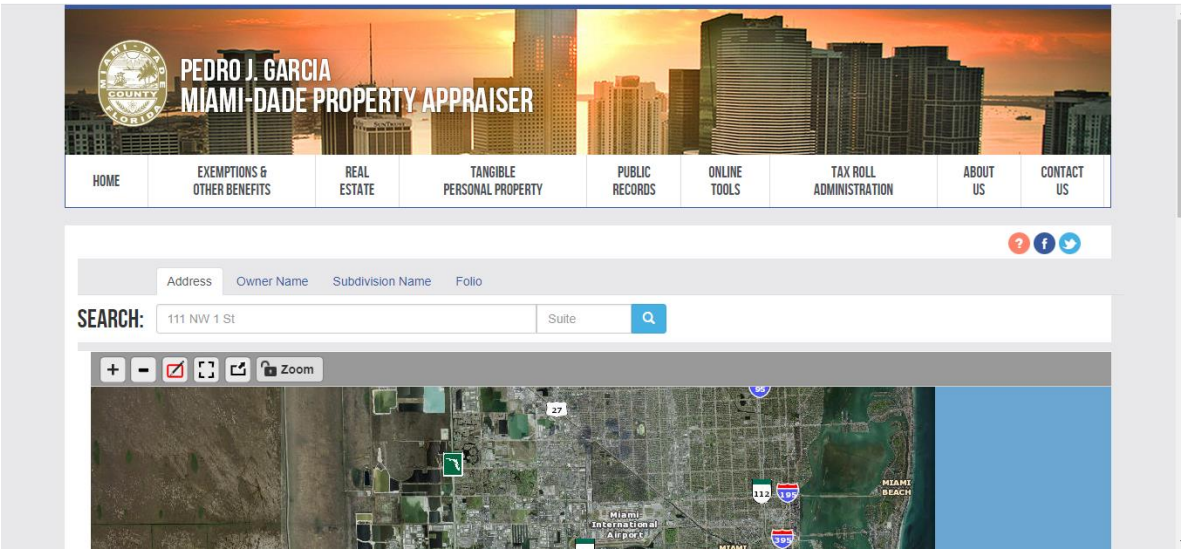

Figura 6. *Miami Dade Property Search.*

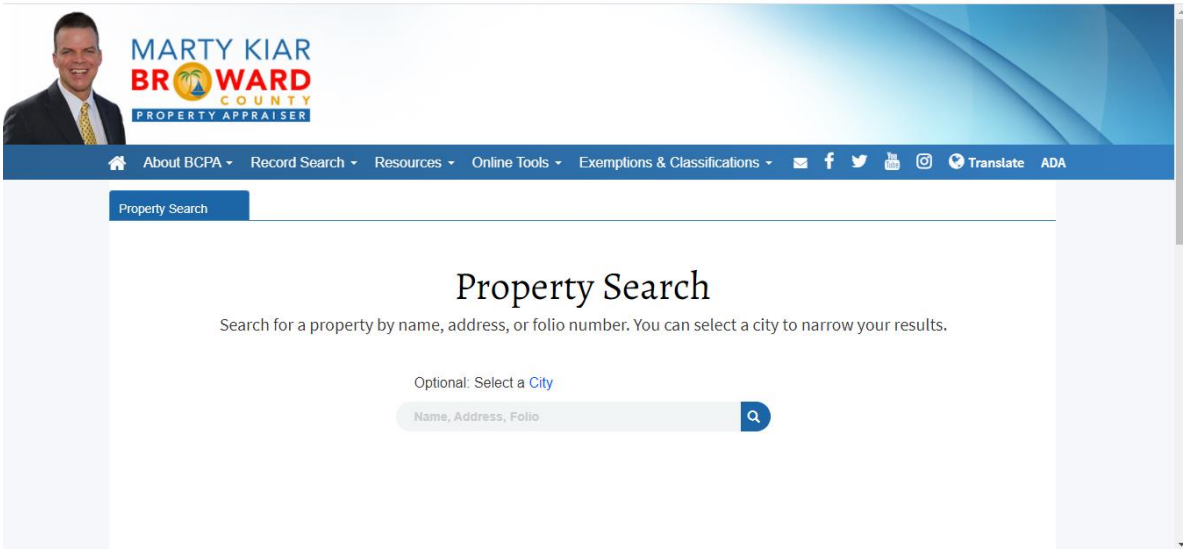

Figura 7. *Fort Lauderdale Property Search.*

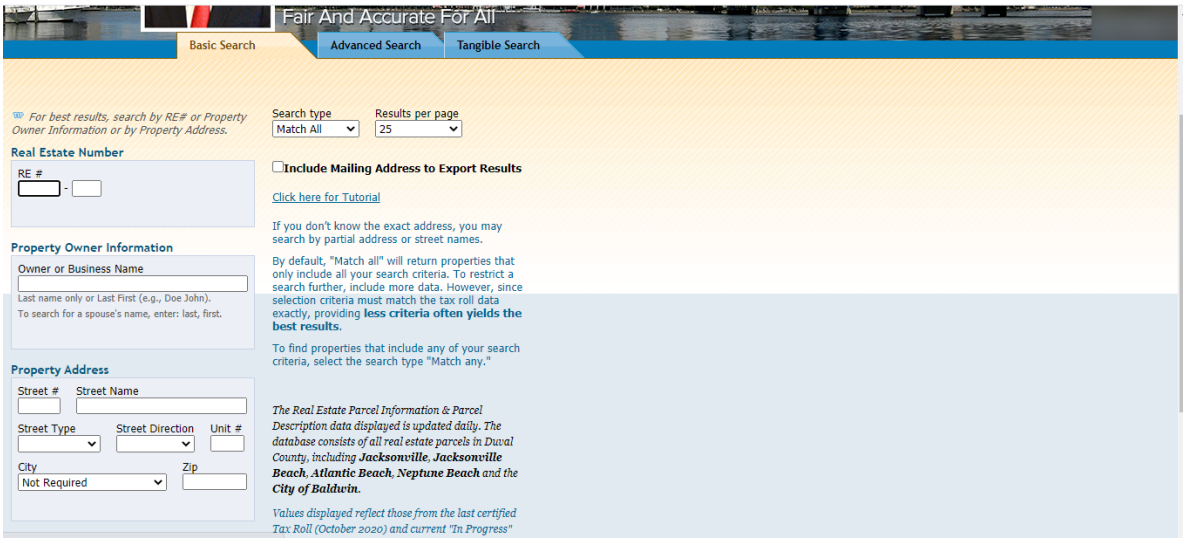

Figura 8. *Jacksonville Property Search.*

Figura 9. *Tampa Property Search.*

Al igual que en el paso anterior, se realizó la recolección de los atributos de búsqueda de cada una de las páginas y se organizaron en una tabla con la intención de facilitar la clasificación de la información y encontrar los atributos más comunes a todas (ver

Tabla 3). La lista completa se puede encontrar en la **Tabla 2** del **Anexo A**.

Con estos atributos como base, luego se completaron los dos grupos de información indispensables para la elaboración del modelo genérico de información relacionada en el primer caso con la información que se debe extraer y en el segundo caso con la información que se necesita conocer para llegar a la información que se debe extraer.

Tabla 3*. Atributos de búsqueda de páginas de información general.*

CIUDAD	CAMPOS DE BÚSQUEDA
MIAMI DADE	Owner name Folio Subdivision name Address
FORT LAUDERDALE	Name Address Folio Parcel ID
JACKSONVILLE	Real Estate number Property Owner Information Property Address / Steet # Street Name

3.2 CREACIÓN DEL MODELO GENÉRICO

El modelo genérico propuesto para este trabajo de grado está compuesto por 3 submodelos fundamentales, cada uno representa las características propias de cada tipo de información inmobiliaria ofrecida por las páginas web destinadas a este propósito. De tal forma que se cuenta con un submodelo relacionado con información general (GeneralInfo) de las propiedades inmobiliarias en los condados y ciudades de ESTADOS UNIDOS. Otro submodelo se encarga de representar la información propia de los impuestos (Taxes), y un último submodelo que se encarga de representar la información de escrituras e hipotecas (Documents) de esas propiedades. Estos 3 submodelos son la base para la creación tanto de los microservicios de la aplicación web que se usa para visualizar propiedades como de los programas de web scraping que se encargan de extraer la información de manera automatizada de cada uno de los portales web contenedores de esta información. Además, se plantea un submodelo auxiliar (submodelo que soporta el web scraping) que es común a los 3 submodelos principales (1 - submodelo de información general, 2 - submodelo de información relacionada con hipotecas y escrituras y 3 - submodelo de información relacionada con impuestos), cuyo propósito consiste en contener la información que se necesita para poder llegar o alcanzar los datos específicos que se espera obtener sobre los bienes inmuebles.

Para lograr plantear cada uno de los submodelos fundamentales se hizo necesaria la extracción de los atributos comunes que existen en cada dominio de información (información general, impuestos y documentos), para dar solución a esto se agruparon los datos que tenían distinto nombre pero que hacían referencia al mismo concepto. La **Tabla 4** muestra la agrupación de campos comunes para el submodelo de información general. Las tablas completas se encuentran en el **Anexo A**.

Tabla 4: *Campos comunes en submodelo de información general.*

ATRIBUTO	SINÓNIMOS OBTENIDOS DE TODOS LOS SITIOS WEB.
Owner	Property Owner/s, Owners, Owner name, Name
Parcel Number	Folio, Residential Account #, Property Id, Parcel control number, RE #, Parcel Id, Parcel Number, HCAD Account (HCAD variable), Parcel
Address	Address, Location Address, Physical Address, Site Address, Primary site address, "", Parcel address, Property Address
Bedrooms	# Bedrooms, Unit/Beds/Baths, Room: Bedroom, beds / Baths / Half, Bedrooms
Lot size	Parcel Area, Acres, Land Area, Total Land Units, Total area, Units, Area, Land SqFt
Building type	PA Primary Zone, Use type, Property type, Property use, State class code, State Code, Type of build
Building Area	Total square feet, Total area under air, Bldg Under Air S.f., Effective area, total living area, building square foot, Total square footage, Total Finished Area, Actual Area, Living area

3.2.1 Submodelo que soporta el web scraping

A continuación, se describen las entidades de información que se requieren para poder encontrar los registros específicos de cada propiedad dentro de cada una de las páginas que contienen la información que debe ser consultada.

Basándose en la información recolectada en la

Tabla 3 referente a los campos de búsqueda en los formularios de cada página web de consulta, se determina cuáles son los atributos de búsqueda más comunes y necesarios para así crear un modelo de datos que almacene la información de los procedimientos o instrucciones que faciliten alcanzar la información objetivo dentro de las páginas. Este modelo auxiliar es una parte importante de cada uno de los tres submodelos principales de datos. La **Figura 10** muestra una vista global de este submodelo auxiliar a través de un modelo físico (diagrama realizado en Power Designer 16.x de SAP) de la base de datos y a continuación se describen las tablas allí presentadas.

City Data Origins: El primer dato necesario y fundamental para poder encontrar la información que se busca, es el link de acceso a la página o formulario a partir de los cuales se realizará la búsqueda de los registros de información de las propiedades. Para esto se creó una tabla que almacena el link de acceso al formulario de búsqueda y la instrucción de iteración en caso de que el sitio web requiera interacciones repetitivas para cada objeto informativo que se busca. Esta tabla se relaciona con la tabla Locations, debido a que los anteriores datos de la

página están directamente relacionados con una localidad específica, por lo que la relación con esta tabla es de uno a muchos

Scraping Fields: Esta tabla permite identificar y diferenciar todos los elementos del DOM que contienen los atributos deseados de una parcela (bien raíz) y que serán obtenidos por el scraper. Aquí se almacenan datos útiles a la hora de realizar el scraping de cada atributo a obtener, como la tabla a la que pertenecen (en caso de que el dato se encuentre dentro de una tabla HTML), el nombre del campo o atributo en sí, su obligatoriedad o no y la fecha de creación y edición.

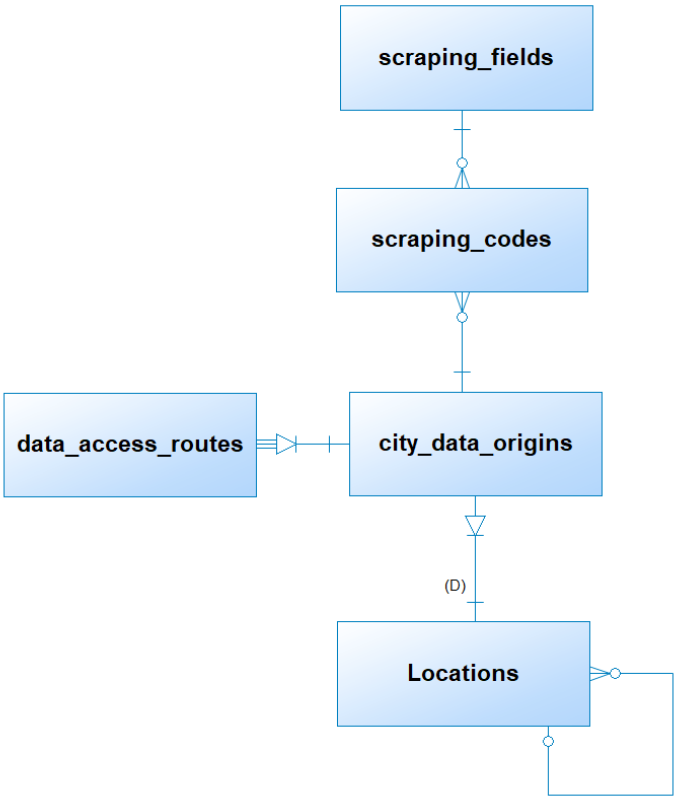

Figura 10. *Vista global del submodelo que soporta el web scraping.*

Scraping Codes: Para lograr acceder a cada dato o atributo de información que se encuentran dentro de los resultados encontrados de cada página, como el id de la parcela, el nombre del dueño, el año de construcción, el número de pisos, etc. es necesario recorrer la estructura HTML de la página para llegar a cada elemento particular. Cada dato tiene un camino o ruta diferente dentro del DOM para llegar a él, y cada página contiene un conjunto de caminos diferente para llegar a todos los atributos que contenga. Más adelante se habla sobre la creación de un lenguaje o conjunto de instrucciones que representan los caminos para alcanzar cada dato dentro de las páginas. Como cada página y cada atributo tendrá rutas diferentes representadas por este lenguaje, cada una de estas rutas se almacenan dentro de una tabla en la base de datos. La tabla scraping_codes está destinada a realizar

esta labor. Aquí se almacenan todas las rutas o conjunto de instrucciones para acceder al dato específico de cada atributo dentro de las páginas web que contienen la información objetivo. Esta tabla está relacionada con la tabla scraping_fields dado que en fields se almacenan los nombres de los atributos y en codes, la ruta para acceder a los datos de esos atributos. La **Figura 11** ilustra la distinción entre un dato almacenado en la tabla scraping fields y un dato almacenado en la tabla scraping_codes

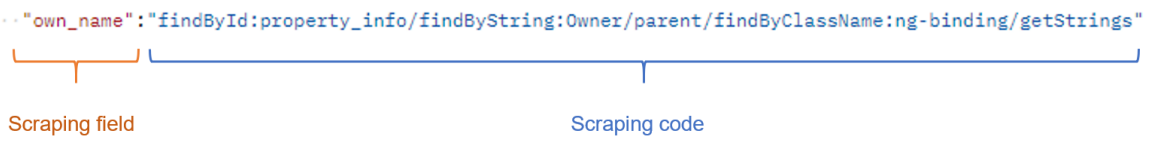

Figura 11. *Ejemplo de scraping field y scraping code.*

Data Access Routes: Las tablas descritas anteriormente son indispensables para almacenar el conjunto de instrucciones definido por un lenguaje de acceso a la información a través de la estructura HTML de un formulario en el que ya esté presentada la información objetivo. Sin embargo, previo a tener la información dada, es necesario determinar el procedimiento para llegar a esa información. En el caso de las páginas de property appraiser y Real Estate, el flujo regular para acceder a esa información es llenar un formulario de búsqueda ingresando datos como el nombre del dueño, la dirección del predio, el identificador de la parcela, entre otros. Posterior a ingresar los datos se debe activar (dar clic sobre) un botón de búsqueda, el cual da como resultado una lista de resultados que hacen match con los criterios de búsqueda. Esto implica la creación de un lenguaje adicional que describa el conjunto de instrucciones y acciones necesarias para realizar la búsqueda de la información, dicho lenguaje está especificado en el **Anexo B**.

La creación de este nuevo lenguaje de instrucciones es distinta al de obtención de los datos de las propiedades encontradas, e implicó la creación de una tabla que almacena el conjunto de instrucciones relacionadas con esta tarea. Esta tabla recibe el nombre de *data_access_routes*.

3.2.2 Submodelo de información general

Con la información de la **Tabla 1**, en donde se recolectan los nombres de las diferentes características presentadas para una propiedad en la página de cada ciudad, se puede notar que los atributos encontrados pueden ser clasificados en dos subgrupos principales: El primero, todos aquellos atributos que describen las características físicas propias de la parcela o propiedad, tales como el tamaño del lote, el año de construcción, el número de pisos, etc. Y en el segundo grupo se encuentran las características propias de la construcción y la estructura, en los que se pueden agrupar atributos como el tipo de construcción, la estructura del techo, número de baños, área de construcción, entre otros.

La **Figura 12** muestra las tablas que conforman este submodelo de información general y la relación de estas con las tablas del submodelo que soporta el web scraping. Este diagrama es la base para el desarrollo del microservicio que se encargará de obtener la información general de las parcelas en las distintas páginas web de las ciudades y condados de ESTADOS UNIDOS. A continuación, se resume el contenido de las tablas del modelo de información general.

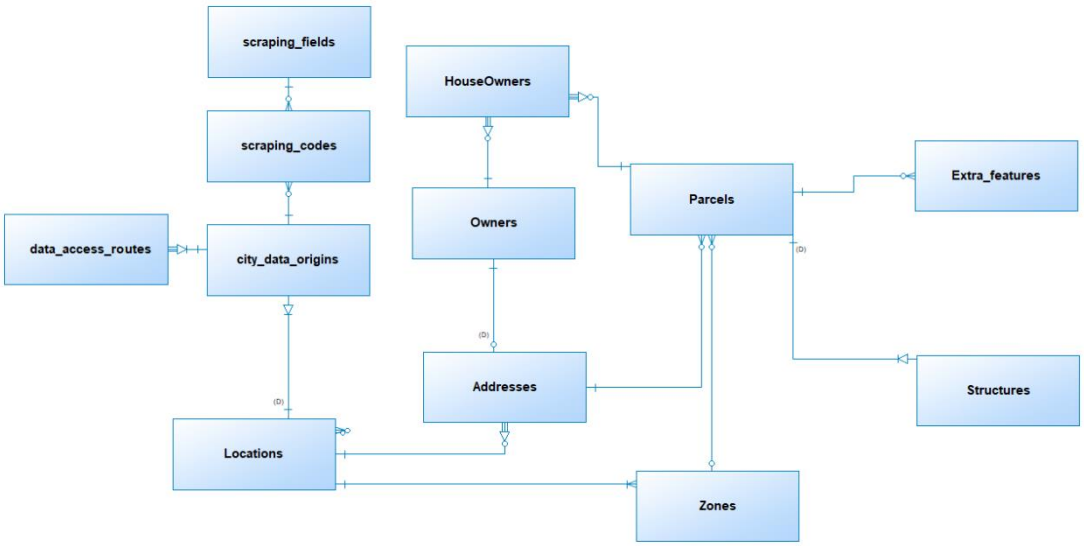

Figura 12. *Vista global del submodelo de información general.*

Parcels: Contiene información que corresponde únicamente a la parcela, como área, información histórica e información de conocimiento general como el año de construcción, la puntuación de la parcela y el precio del mercado aplicado sobre ella. Es la tabla principal de este modelo y relaciona una parcela con todas las demás tablas que almacenan información de una parcela.

Structures: Almacena la información estructural de una parcela, lo que significa que la tabla contiene únicamente atributos de tipos de materiales, número y tipo de cuartos, tipo de construcción y área construidas dentro de la parcela.

Extra Features: Esta tabla registra características adicionales de importancia para el proceso de comparación entre parcelas (casas, aptos, lotes y otros), pero estas características sólo están presentes en algunas páginas web de las que se estudiaron. Estas características extra (Extra Features) incluyen, por ejemplo, piscina, jacuzzi y sauna. Como puede haber muchos atributos extra para una parcela, la relación con la tabla Parcels es de uno a muchos.

Zones: Otros de los atributos informativos que suelen estar presentas en las páginas de información general de las parcelas se denomina zones, que en términos generales es una indicación de división geográfica en sectores que se agrupan y clasifican bajo ciertos criterios, como por ejemplo los tipos de construcciones

permitidas en el área, la capacidad productiva del sector, grados de riesgo, intensidad de una amenaza, etc. Esta información puede ser de considerable importancia para un comerciante de bienes inmuebles, ya que ésta también tiene una incidencia directa sobre el costo de las propiedades y la valorización futura.

Se pueden identificar también otros grupos de información entre todos los atributos encontrados en las páginas, que, si bien no son propios de las características de las parcelas o propiedades, si son de fundamental importancia para llevar a cabo el proceso de investigación y adquisición de un inmueble. Estos son todos aquellos atributos relacionados con la información de los propietarios de los inmuebles y su ubicación.

Owners: Dentro de la información sobre los propietarios de las parcelas que se encuentran en las páginas, se encuentran datos invariables como el o los nombres de los propietarios, su dirección, y en algunos casos, teléfonos de contacto. Dado que un propietario puede llegar a tener muchas propiedades, y una propiedad o parcela puede también pertenecer a más de un dueño, la relación entre la tabla Owners y Parcels se establece como muchos a muchos.

Addresses: En cuanto a los atributos relacionados con direcciones, no se incluyeron dentro de las tablas Parcels y Owners, que son los dos grupos que se relacionan con esta información, debido a que precisamente por ser atributos comunes entre dos entidades, y que además considerando que una dirección encontrada en una propiedad no tiene que corresponder necesariamente a la misma del propietario del inmueble, es más conveniente representar y almacenar esta información en una tabla a parte que esté relacionada tanto con las propiedades como con los dueños.

Locations: De entre todos los atributos agrupados y categorizados de la información obtenida en las páginas web de property appraiser, esta es la primera entidad que en el esquema relacional no está directamente relacionada con la tabla Parcels. Esto debido a que la tabla Locations almacenará la información propia de las ciudades, estados o condados de donde se está extrayendo la información. Tendrá una relación de uno a muchos con Addresses debido a que una ciudad puede tener muchas direcciones de propiedades registradas, pero una dirección no puede pertenecer a más de una ciudad.

Con esto, toda la información que se requiere extraer de las páginas de Real Estate y property appraiser está representada en un modelo relacional de información que permitirá su acaparamiento. La forma final de este modelo incluirá el modelo auxiliar para la realización de web scraping.

3.2.3 Submodelo de información relacionada con hipotecas y escrituras

Al igual que en el modelo de información general, para poder acceder a la información sobre escrituras e hipotecas que se encuentran en las páginas y

portales web destinadas a este tipo de información, se requiere de una serie de pasos previos para realizar las búsquedas de los documentos de cada parcela. A diferencia de las páginas de información general, las de documentos, en la mayoría de los casos, requieren de datos más específicos, como lo son el número de libro y de página relacionada con la propiedad. Para algunas ciudades, estos datos pueden ser encontrados dentro de las páginas de información general, por lo que obtener la información general de una parcela es el paso previo para poder encontrar la información de documentos como escrituras e hipotecas. Cabe aclara también que tanto las escrituras como las hipotecas pueden ser encontradas en la misma página web, dado que ambos son tipos de documentación y se distinguen principalmente por el atributo “type of document”. Como se puede suponer, estos atributos adicionales de búsqueda fueron tenidos en cuenta en el diseño del modelo de datos para los scrapers de los documentos de las parcelas.

Los atributos obtenidos también se organizaron dependiendo de su naturaleza para ser representados en diferentes tablas en la base de datos. Teniendo en cuenta que este será un microservicio distinto al de Información General, dispone de una base de datos propia, y en el diagrama (ver **Figura 13**) se muestran además las tablas del modelo que soporta el web scraping y otras del submodelo de información general, aunque se diferencian en algunos atributos. A continuación, se describen cada una de las tablas propias de este submodelo.

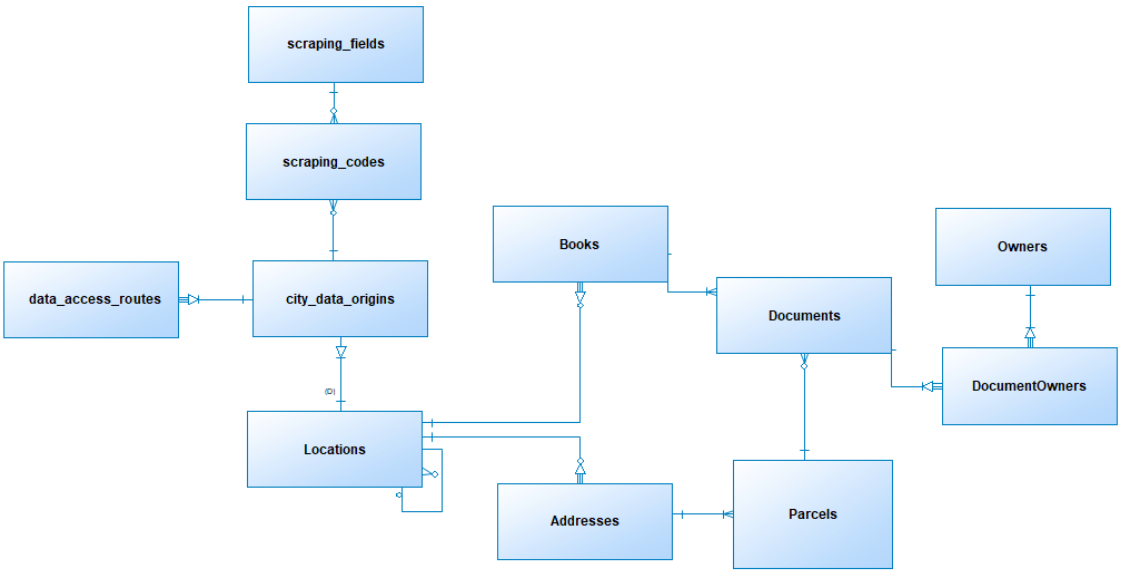

Figura 13. *Vista global del submodelo de hipotecas y escrituras.*

Documents: Es la tabla distintiva de este submodelo, en ella se almacenan los atributos encontrados para los documentos en las páginas web destinadas a la consulta de este tipo de información. Debido a que un documento de este tipo pertenece a una propiedad, tendrá una relación directa con la tabla Parcels. También está relacionada con la tabla Books debido a que un documento se

encuentra en un libro y número de página específico. La tabla Owners también está relacionada con esta tabla debido a que una propiedad puede tener varios propietarios, y un propietario puede tener más de una propiedad, por esto se estableció una relación de muchos a muchos entre la tabla Documents y la tabla Owners.

Parcels: En esta tabla se almacena la información relacionada con la propiedad encontrada en las páginas web de escrituras inmobiliarias. Esta tabla contiene un replica de la tabla Parcels del submodelo de información general de una manera menos detallada, incluyendo sólo los atributos útiles para que se realice con éxito el scraping de los Documentos. Se relaciona con la tabla Documents dado que una propiedad puede contar en su historial con varios documentos de escritura.

Books: Esta tabla contiene la información de libros en donde están los documentos de escrituras de las propiedades. En algunos casos esta información es útil y necesaria, por lo que podría ser indispensable al momento de realizar una búsqueda detallada de información.

Address: Almacena la dirección de las propiedades encontradas en las páginas. También tiene información replicada de la misma tabla del submodelo de información general que es útil para identificar las distintas parcelas y relacionar la información con los diferentes modelos. Esta table se relaciona únicamente con la tabla Parcels.

DocumentOwners: Esta es una tabla intermedia que se encarga de relacionar los múltiples posibles dueños de una propiedad con las múltiples posibles propiedades que puede poseer un propietario.

Owners: En esta tabla se guarda la información encontrada en las páginas sobre los propietarios de las parcelas. En el caso particular de las páginas de documentos de escrituras, la única información mostrada sobre el propietario es el nombre.

3.2.4 Submodelo de información relacionada con impuestos (taxes)

La forma de extraer la información de las páginas web para el modelo de impuestos es básicamente el mismo que para el modelo de documentos. Consta de un lenguaje de instrucciones para la interacción y obtención de información en las páginas web, por lo que además de las tablas destinadas a la información sobre impuestos, incluye las tablas que sirven al lenguaje procesador de instrucciones. Este modelo también requiere de información específica de una parcela para lograr realizar el scraping de manera exitosa, por lo cual antes de obtener la información de impuestos se necesita obtener el id de la parcela, información que se encuentra en el modelo de información general. La **Figura 14** muestra las tablas que conforman este submodelo de información relacionada con impuestos y las tablas relacionadas de los otros submodelos.

Taxes: Esta es la entidad central de este submodelo. Se encarga de representar los atributos propios de los impuestos como el número de cuenta, el total pagado, reducciones de monto, entre otros. Se relaciona directamente con la tabla Parcels por la relación de correspondencia entre estas dos entidades.

Assessments: Esta tabla recoge la información del avalúo del impuesto con mayor detalle, en caso de encontrar información más completa en alguna de las páginas web de impuestos.

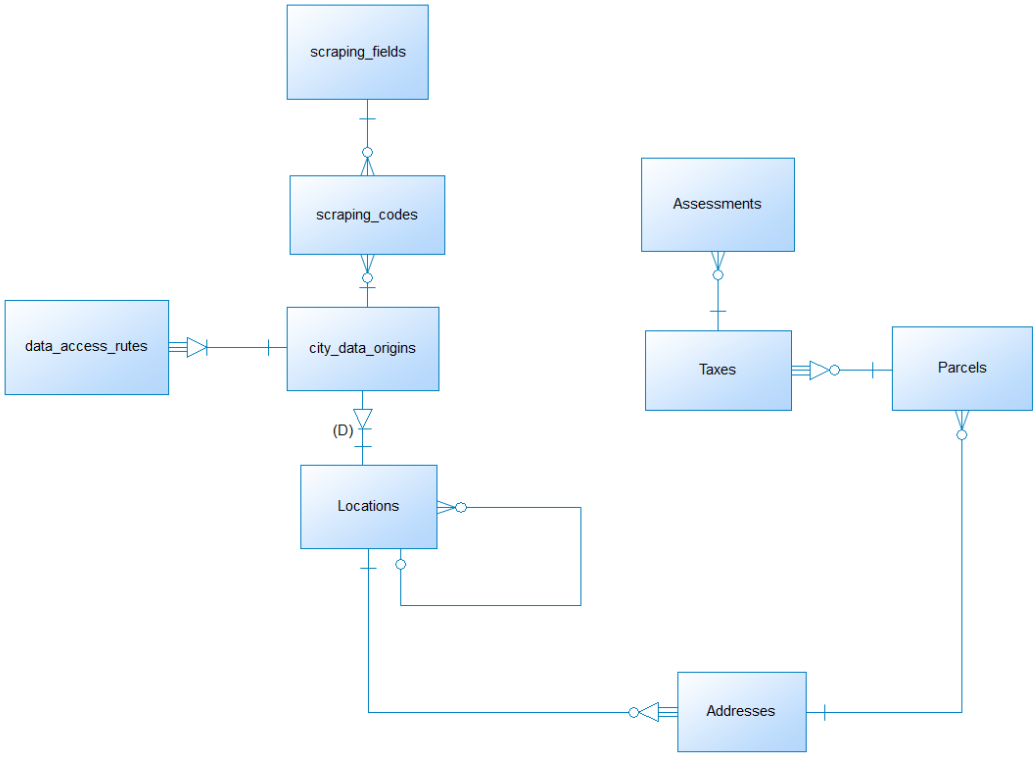

Figura 14. *Vista global del submodelo de impuestos (taxes).*

Parcels: Esta replica de la tabla Parcels del submodelo de información general, es una versión reducida, útil y necesaria para el almacenamiento de los impuestos.

Addresses: Esta tabla es una réplica de la tabla Addresses del submodelo de información general. Se usa aquí para relacionar la información sobre direcciones encontrada en las páginas de impuestos que es muy similar a la presentada en las páginas de información general.

En resumen: La creación del modelo de datos implicó un proceso que comenzó con la definición de las páginas web a ser analizadas, teniendo en cuenta la factibilidad de inversión de bienes raíces en cada ciudad. Una vez definidas las páginas web, se analizó el contenido de datos dentro de cada una de estas, extrayendo los atributos relevantes para definir el modelo de los datos. De esta

agrupación y clasificación de datos y atributos dentro de las páginas, se relacionaron términos semejantes y equivalentes con la intención de generalizar dichos atributos en todas las páginas. Esta clasificación de atributos fue la base para empezar a crear cada uno de los submodelos de datos para cada tipo de información de los bienes raíces. Al final se obtuvieron 3 submodelos principales: Uno que representa la información relacionada con la información general de la parcela, otro que representa la información de impuestos de la propiedad, y otro que representa la información contenida en la página de documentos (escrituras e hipotecas). Además de estos 3 submodelos, se propuso un submodelo auxiliar que es trasversal a los demás, cuyo propósito es almacenar la información requerida para llegar a cada uno de los registros de propiedades dentro de las páginas.

CAPÍTULO 4

4 APLICACIÓN WEB

Para materializar la idea de que un usuario logre visualizar la información inmobiliaria de cualquier parcela mediante la personalización del modelo genérico de información, se desarrolló una aplicación web que permite configurar los impuestos, los documentos y la información general para las parcelas de cada ciudad en Estados Unidos. En ella se visualiza de una manera amigable la información que el scraper logra extraer al realizar su correcta configuración.

La aplicación desarrollada fue denominada PiExtractor (Property Information Extractor). Las herramientas y entornos de desarrollo utilizados fueron: .NET Core para el desarrollo del Backend, Python para el desarrollo del Scraper, Angular para el desarrollo del Frontend y Docker en el servicio Cloud Run de Google Cloud para encapsular el entorno que necesita cada aplicación y hacer más sencilla la tarea de despliegue. Las bases de datos de Microsoft SQL Server (compatible de forma nativa con .NET Core) para cada microservicio y alojada en una máquina separada, para esto se utilizaron dos servicios diferentes proporcionados por Google Cloud, el primero, Cloud SQL, el cual permite crear una instancia de una base de datos SQL de manera rápida y sencilla, sin configuraciones de servidor adicionales, este se utilizó solamente para el microservicio Identity que no necesitó de configuraciones específicas de la base de datos, el segundo servicio utilizado fue Compute Engine, en el cual se realizaron las distintas configuraciones requeridas para el despliegue de una máquina en la nube y la configuración en ella de una base de datos en escucha. Mensajes JSON para la comunicación, Nginx para encapsular las aplicaciones en un servidor, PaperTrail para visualizar el comportamiento general de la aplicación y finalmente para el despliegue del Scraper se utilizó el servicio Compute Engine de Google Cloud.

4.1 ARQUITECTURA

Para lograr el desarrollo de una aplicación web basada en microservicios, se hizo necesaria una correcta documentación sobre los patrones de arquitectura que se utilizan en la elaboración de este tipo de software, para lograrlo se utilizó la documentación puesta a disposición por el autor de diversos patrones de microservicios disponible en https://microservices.io.

La arquitectura implementada se dividió en dos módulos, Backend y Frontend. El Backend de la aplicación fue dividido en cuatro microservicios, uno para identificación (Identity API), uno para información general (General Information API), uno para Impuestos (Taxes API) y uno para hipotecas y escrituras (Documents API). También lo componen una API para web scraper y una API Gateway que coordina cada una de las peticione hacia los microservicios. El Frontend se compone de una aplicación web que proporciona la interfaz mediante la cual los clientes interactúan y desde la cual se logra configurar el scraper. La arquitectura que se planteó para la aplicación se muestra en la **Figura 15**.

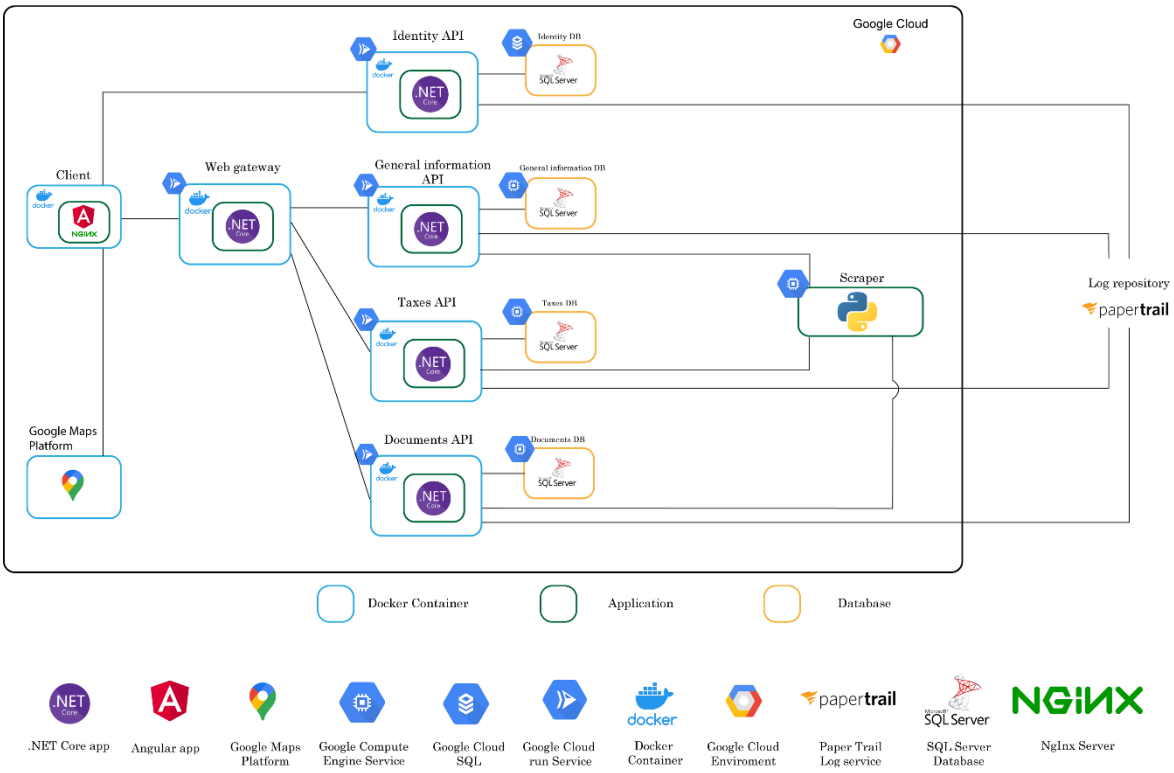

Figura 15. *Arquitectura de la aplicación web.*

En el Backend, los microservicios de Identity, GeneralInfo, Taxes y Documents cuentan con su propia base de datos desplegadas en una maquina diferente y se conectan a un repositorio de logs compartido entre todos, un Health-check propio y seguridad basada en Json Web Tokens. Estos son los encargados de mantener la información recolectada por el Scraper a través del tiempo y de enviarla al Frontend cuando se requiere. La API Gateway reúne las características mencionadas de los otros microservicios, exceptuando la base de datos. Este Gateway gestiona las peticiones que llegan desde un cliente. Además, en este módulo también existe un API encargada de realizar el web scraping a cada uno de los sitios web configurados

y enviar la información a cada uno de los microservicios correspondientes cuando estos hagan el llamado necesario.

Los clientes que deseen conectarse con la aplicación utilizarán en sus dispositivos el aplicativo creado para el Frontend, el cuál envía las diferentes peticiones requeridas por los clientes al API Gateway, el cual es el único encargado de comunicarse con los otros microservicios. El Frontend nunca tiene contacto directo con los microservicios a excepción del microservicio Identity, el cual se encarga de manejar las sesiones de usuario y de enviar el Json Web Token al cliente para que el Gateway lo autorice cuando le solicite una petición. Por lo anterior, el cliente solamente tiene contacto directo con el Identity API, acto que es necesario para este antes de solicitar información al Gateway o no se le concederán los permisos necesarios para su solicitud. A continuación, se describen con más detalle cada uno de los componentes de toda la aplicación web.

4.1.1 Frontend

El Frontend está conformado por una única aplicación, la cual consume los servicios de Geolocation, Maps JavaScript API y Places API de Google Maps. Aquí el usuario logra interactuar con la aplicación según el rol que le corresponde, permitiéndole obtener la información de las parcelas o editar la configuración del scraper según la fuente de información que requiera adaptar para cada ciudad. La aplicación se describe a continuación.

Aplicación Frontend: A ella se conectan los clientes que utilicen un navegador web en sus dispositivos. Su funcionamiento se define como sigue, la aplicación contiene un módulo de inicio de sesión el cual se conecta al microservicio Identity y solicita un Json Web Token, el cual se envía de regreso al Frontend en caso de que el usuario digite bien sus credenciales, este Token se envía en cada una de las peticiones que el cliente realice al API Gateway. En caso de que se genere un Token correcto el usuario podrá acceder a todas las funciones que su rol le permita (ver **Figura 16**).

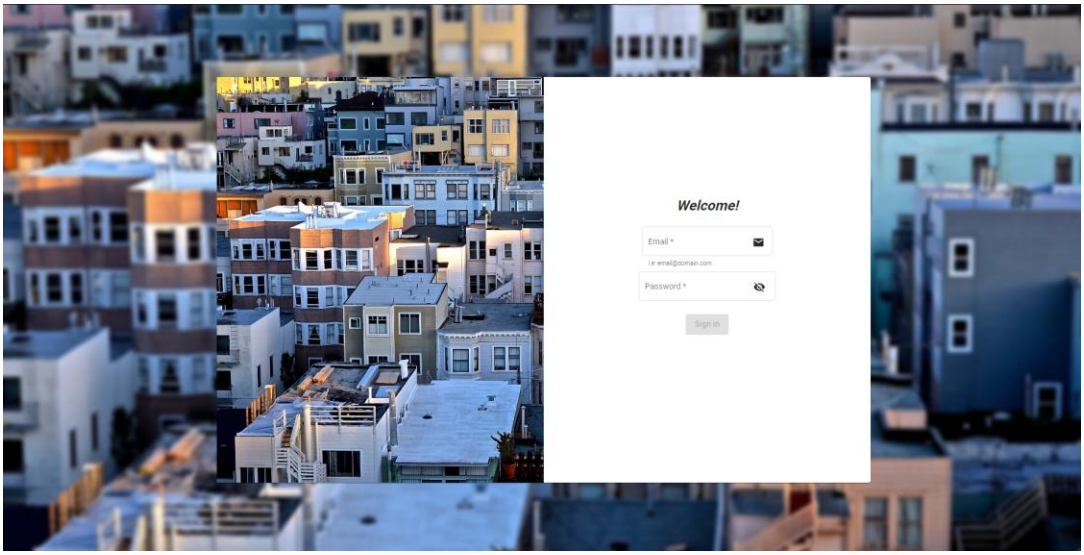

Figura 16. *Formulario de autenticación de la aplicación.*

Los tipos de usuarios que pueden acceder a la aplicación son dos. Usuario Admin, el cual puede configurar las instrucciones para que el scraper obtenga la información de las parcelas de cada ciudad y usuario Agent que sólo tiene la opción de solicitar a la aplicación datos existentes o nuevos de cada una de las parcelas mostradas por Google Maps.

Antes de empezar el desarrollo del Frontend se hizo necesario saber cómo un usuario podía seleccionar una parcela y de qué manera se obtendría la dirección de la parcela seleccionada. Para lograr esto se realizó la búsqueda de diferentes servicios que brindaran una posible solución. Se tuvieron en cuenta diferentes opciones para lograr la captura de direcciones de las parcelas, una de ellas OpenStreetMap [57], servicio de licencia abierta el cual está dotado de servicios bastante útiles para el proyecto, en un comienzo se exploraron algunas de las APIs que provee este servicio, el principal fue Nominatim; utilizada para Geocoding. Se realizaron pruebas escogiendo parcelas al azar en Google Maps y en OpenStreetMap buscando similitud entre las direcciones obtenidas, luego la misma dirección se buscaba en Zillow, obteniendo como resultado poca precisión al obtener las direcciones correctas del servicio de licencia abierta. Teniendo en cuenta que para que el scraper fuera exitoso necesitaba buena exactitud en las direcciones que provea el servicio, entonces se decidió no utilizar este servicio.

Otra alternativa fue poblar la base de datos de la aplicación con datos abiertos existentes en la World Wide Web, el primero, openAddresses [58], del cual se obtuvieron una gran cantidad de datos, los cuales lograron poblar la base de datos de distintas direcciones y coordenadas geográficas, permitiendo así la independencia de la aplicación con servicios de terceros. Esta alternativa no fue elegida gracias a que el servicio no era completo, no se encontraban todas las direcciones existentes, había una gran cantidad de estas que faltaban en los archivos y algunas estaban en formatos diferentes al establecido por el servicio ya

que las direcciones recolectadas provienen desde distintos colaboradores, lo cual convertía el algoritmo de normalización de direcciones en una tarea bastante lenta por la cantidad de datos que el algoritmo trataba. Un problema adicional se presentaba en la carga de direcciones a la aplicación, ya que al subir las direcciones de una ciudad al Backend (general information, taxes y documents), las APIs debían quedar en un estado de mantenimiento por una, dos y hasta más horas mientras se cargan todas las direcciones encontradas. Otra opción que se evaluó en paralelo con la anterior fue la de obtener los registros públicos de las distintas direcciones de Estados Unidos desde https://www.transportation.gov/gis/national-address-database/national-address-database-nad-disclaimer, pero se descartó por las mismas razones descritas en la segunda alternativa.

Teniendo en cuenta que las anteriores alternativas no fueron apropiadas, se escogió como mejor opción a Google Maps. Tomando los servicios de esta API se logra obtener la información de manera sencilla, exacta y a la vez actualizada, sin necesidad de que un administrador de sistema tenga que bajar los servicios en un día y hora específicos para subir las diferentes direcciones que se encuentren en una ciudad, permitiendo una mayor disponibilidad de la aplicación para los usuarios. Además, se aprovechó el programa de Google Cloud que ofrece una prueba de 300 dólares consumibles durante 3 meses, lo cual beneficiaba el desarrollo de la aplicación con las diferentes cuentas que se podían vincular.

Soportados en lo anterior, a continuación, se describe el funcionamiento principal de la aplicación, éste se divide en dos: el primero, consiste en mostrar a los agentes inmobiliarios (usuarios Agent) la información que se obtiene de una parcela cualquiera ubicada en Estados Unidos, esto se logra mediante la API de Google Maps y los servicios desarrollados. Los agentes inmobiliarios interactúan a través de un mapa proporcionado por Google Maps API, ellos pueden seleccionar la parcela de la cual desean obtener información, luego, la API Geolocation envía la dirección devuelta al Frontend, se empaquetan los datos y se envía la petición al Backend, el cual se conecta al Scraper y espera su respuesta con la información general, de impuestos y de documentos que logra obtener para la parcela seleccionada, guarda dicha información y luego la envía de vuelta al Frontend (ver la **Figura 17**).

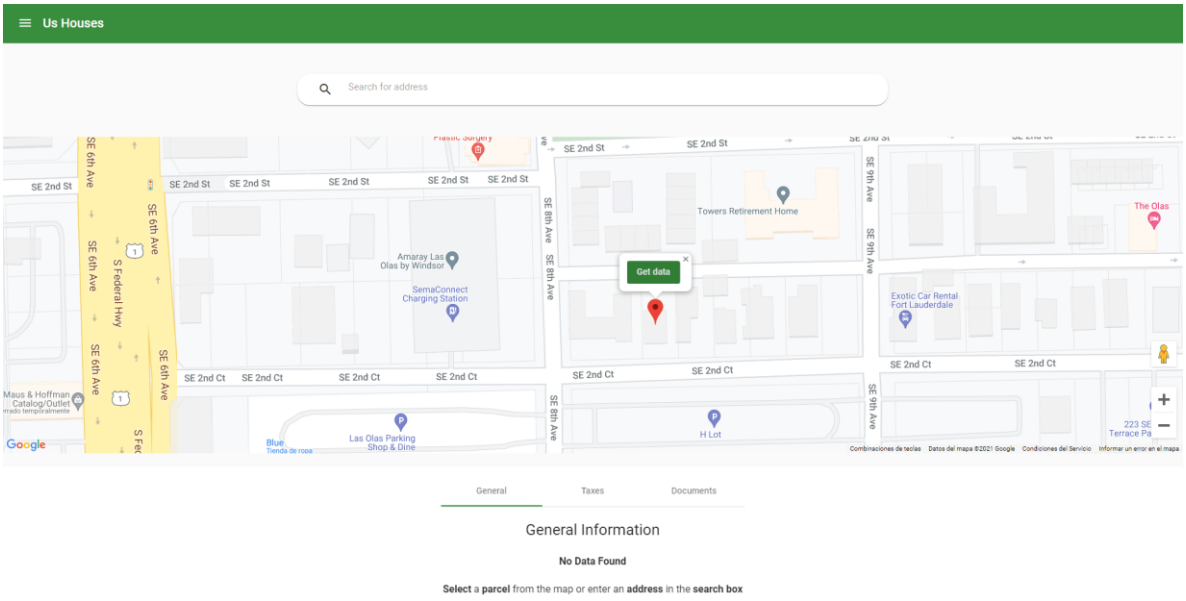

Figura 17. *Selección de una parcela para solicitar su información.*

Por otro lado, el usuario Admin se encarga de configurar cada una de las instrucciones necesarias para obtener la información de cada ciudad o condado de Estados Unidos. Para lograrlo el administrador ingresa la URL de cada página de donde es necesario obtener la información y si es necesario indica la instrucción requerida para que el scraper realice una iteración para obtener la información de cada resultado (ver **Figura 18**).

Posteriormente ingresa los pasos necesarios para llegar desde la página ingresada hasta la página donde se encuentra la información requerida para cada parcela (ver **Figura 19**).

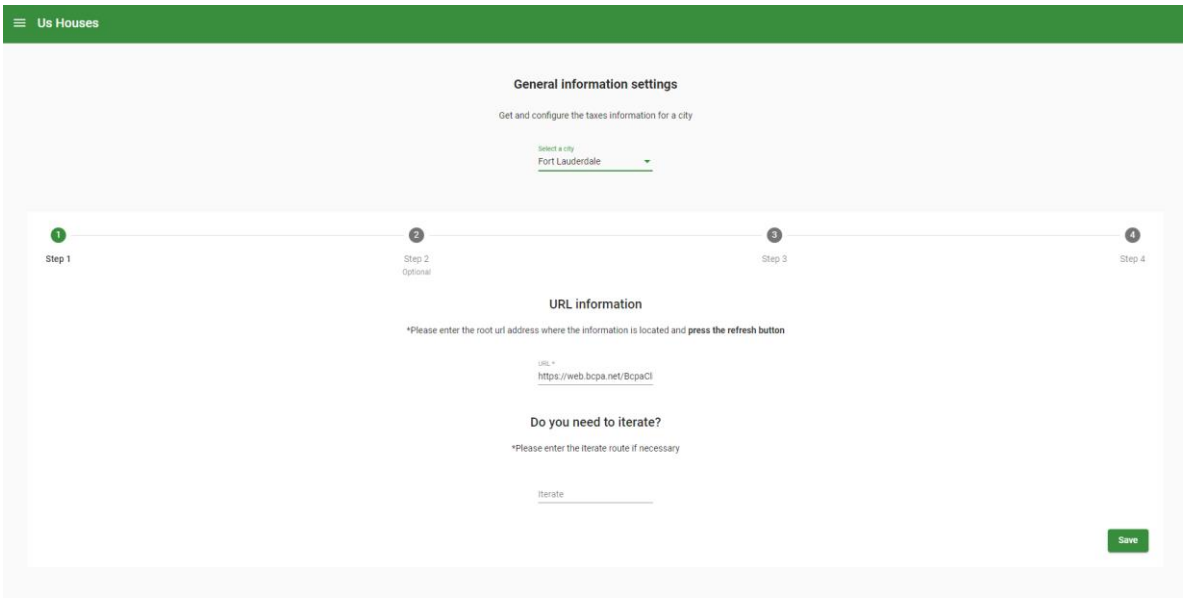

Figura 18. *Definir la ruta de obtención de datos para la ciudad.*

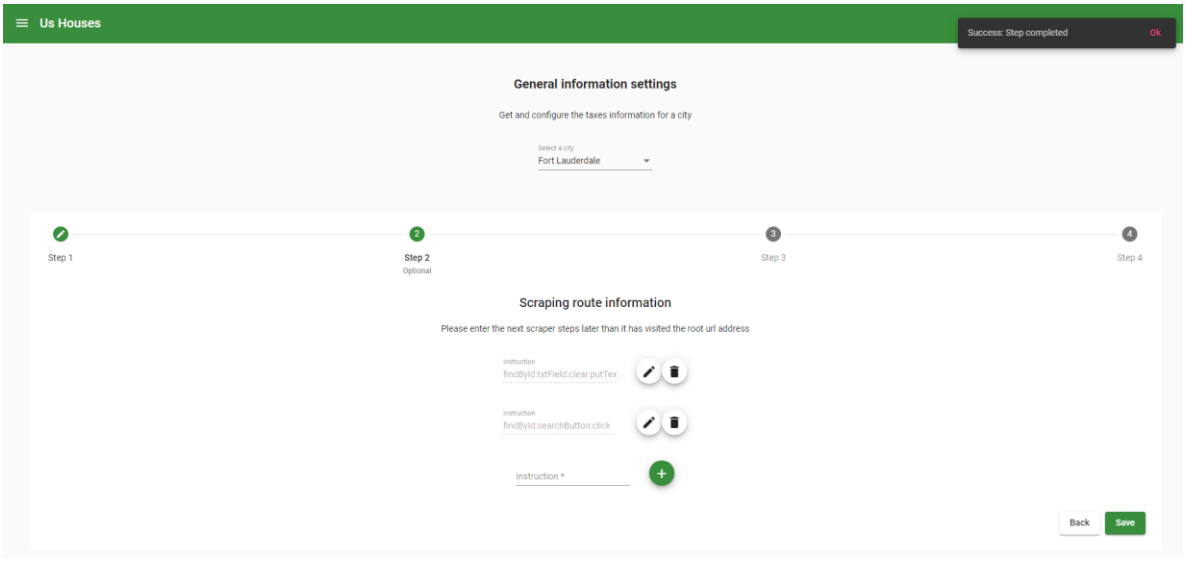

Figura 19. *Definir las instrucciones para llegar a los datos requeridos.*

Finalmente, el administrador ingresa una ruta por cada uno los campos (atributos) que desee obtener. Se elige el campo y se ingresa la instrucción que se necesite para lograr extraer cada dato (ver **Figura 20**). El administrador realiza esta configuración para cada uno de los diferentes dominios de información (Información general, impuestos y documentos) según se requiera (ver **Figura 21**).

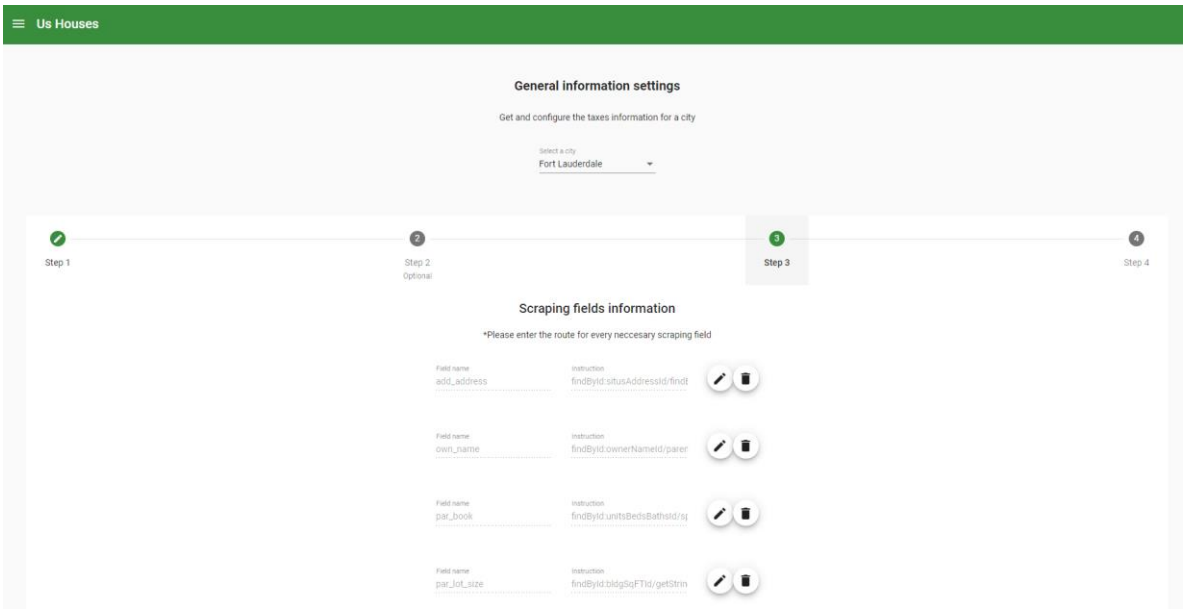

Figura 20. *Definir como extraer los atributos (campos de información).*

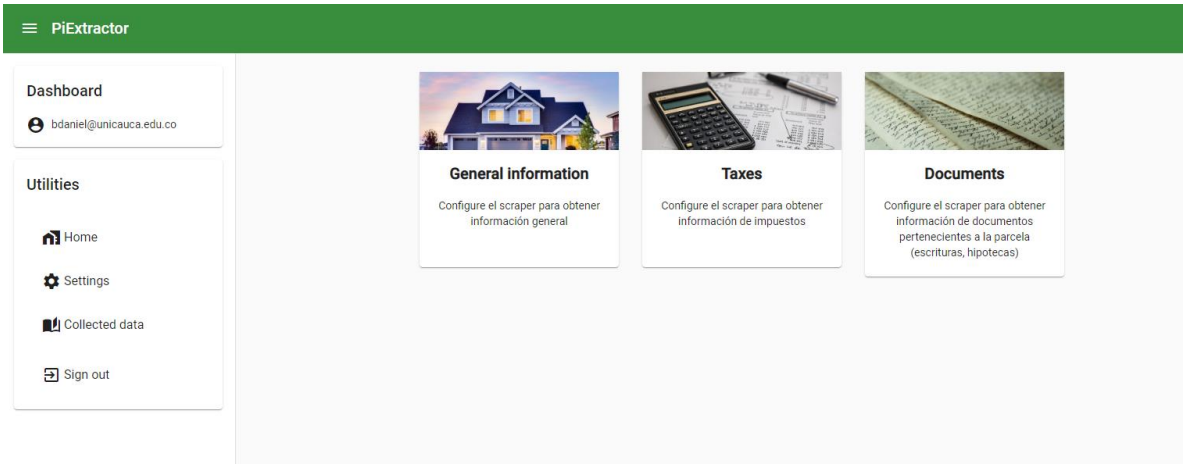

Figura 21. *Configurar los campos de cada grupo de información.*

Es preciso destacar que un usuario Agent solo puede obtener información de una parcela en una localidad específica cuando el Administrador previamente ha configurado la localidad para que el scraper logre obtener los datos correctamente, en caso contrario no se mostrará información al Agente.

Además de lo mencionado, los usuarios Agent y Admin también puede visualizar, buscar y filtrar cada una de las parcelas de las cuales se ha obtenido información, pueden observar información general, documentos e impuestos asociados a las parcelas existentes en la base de datos. La búsqueda de la información se puede realizar consultando cualquiera de los atributos principales de la propiedad. Esto con el fin de abrir la posibilidad de realizar diferentes análisis que requieran los usuarios (ver **Figura 22** y **Figura 23**).

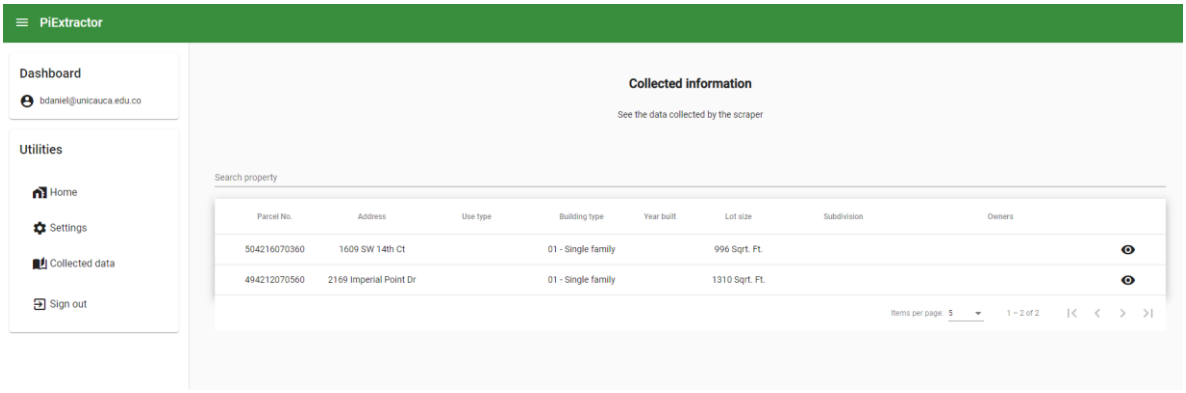

Figura 22. *Análisis de la información inmobiliaria recolectada.*

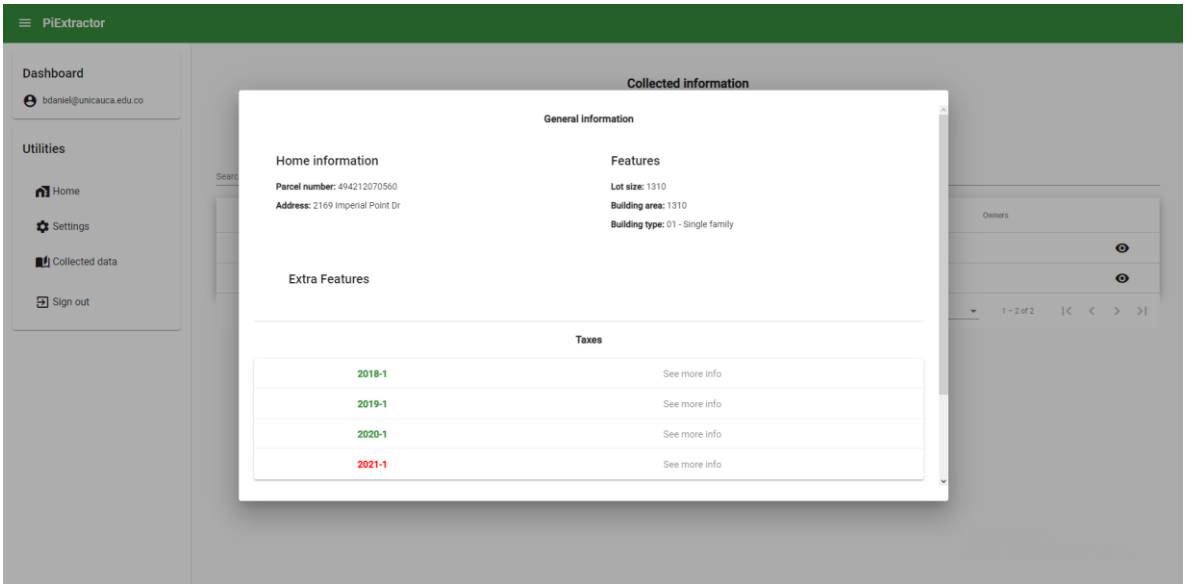

Figura 23. *Visualizar información detallada de una parcela listada en el informe*

La aplicación logró ser desplegada gracias al apoyo de la empresa ATIX DIGITAL S.A.S. que facilitó 5.000 créditos de una cuenta de Google Cloud para usarlos a conveniencia y lograr con esto poner en producción el software desarrollado. La aplicación fue desplegada en un contenedor Docker en el servicio Cloud Run de Google Cloud y escucha gracias a que se ejecuta en conjunto con un servidor Nginx. Cada contenedor se encuentra en el servicio Container Registry y está asociado a su respectiva instancia del servicio Cloud Run de la misma nube, facilitando nuevos despliegues cuando se realicen cambios dentro de la aplicación. En caso de que se necesite implementar una nueva versión de la aplicación, se construye el contenedor, se sube al servicio de Container Registry y se asocia el contenedor a la aplicación de Cloud Run con un par de clics. Todo lo anterior se pensó para lograr un despliegue sencillo, sin necesidad de conectarse para configurar una máquina específica, abrir puertos de red en el firewall y/o configurar distintas funcionalidades de la máquina que tendrían que modificarse para poner en línea la aplicación.

La aplicación Frontend está construida bajo el entorno de desarrollo Angular, las bibliotecas de diseño Bootstrap y Angular material gracias a su fácil usabilidad y documentación, los servicios de Geolocation, Maps JavaScript API y Places API de Google Maps, los cuales son útiles para el manejo del mapa y obtener las direcciones de las diferentes parcelas.

4.1.2 Backend

En este módulo se encuentran todas las aplicaciones con las cuales un cliente web no interactúa directamente, en ellas se encuentra la lógica de la aplicación en general. Contiene los microservicios; Identity, GeneralInfo, Taxes y Documents, una API Gateway y un Scraper.

Antes de entrar en detalle sobre cada uno de los servicios implementados, se comentan algunos aspectos generales de los microservicios, en especial la forma como se organizaron y cada uno de los componentes comunes que tienen estos. Luego se describe cada uno de los servicios y finalmente se detalla la API Gateway y el Scraper.

Descomposición: Los microservicios se organizaron (dividieron) en cuatro de acuerdo a la descomposición por subdominio [59], lo anterior gracias a que se logró identificar el desacoplamiento que logra tener cada uno de los objetivos informativos que quieren gestionarse en la aplicación, un ejemplo de esto, es cuando se habla de información hipotecaria y de escrituras de una parcela, la cual se busca en distintos sitios web y está organizada de manera distinta a la información que proveen los sitios web que guardan la información de impuestos, en otras palabras, el dominio de cada microservicio fue definido según el enfoque de la información que requiere gestionar. El microservicio Taxes se encarga de gestionar solo la información de tributos, el microservicio Documents gestiona solamente la información de actas (escrituras e hipotecas), el microservicio de General Information (GeneralInfo) se encarga solamente de la información común de una parcela como estructuración y tamaños de éstas, por último, el microservicio Identity gestiona la información de las sesiones para los usuarios de la aplicación.

Health-check: De acuerdo con los patrones de microservicios, cada microservicio posee un Health-check propio, éste es útil cuando se requiere conocer el estado de salud de este. Lo que quiere decir, que hay una ruta específica que muestra los resultados de las pruebas que se realizan automáticamente, para verificar si un microservicio se encuentra en línea o se encuentra caído, además mostrando el componente que causa la caída del microservicio. A éste Health-check se accede añadiendo a la URL de cada microservicio, la ruta “/healthchecks-ui#/healthchecks”. Por ejemplo, el microservicio General Information tiene como URL https://rescrape-generalinfo-hmdzwaukuq-ue.a.run.app, si le agregamos la ruta ““/healthchecks-ui#/healthchecks” al final, muestra la interfaz que se presenta en la **Figura 24**. Aquí se muestra en la primera línea el estado general del servicio, que en este caso es el de “General Information”. Este estado puede tener como resultado Healthy o Unhealthy, la primera indica si el microservicio está funcionando correctamente y la segunda aparecerá en caso de que se genere al menos un error en las verificaciones. Al desplegar la fila “General Information”, se pueden apreciar las verificaciones que se están realizando, General Status y Database. La primera indica el estado general del microservicio, si éste no se encuentra funcionando (Unhealthy) es porque ocurrió una excepción en la aplicación o el servidor donde está alojado está teniendo problemas, en caso contrario, Healthy indica que el sistema está disponible y funcionando sin ningún error. La segunda fila indica el estado de la base de datos, en caso de mostrar Unhealthy es porque el microservicio no se ha logrado conectar con la base de datos y se debe verificar la disponibilidad de ésta, en caso contrario se mostrará en estado Healthy.

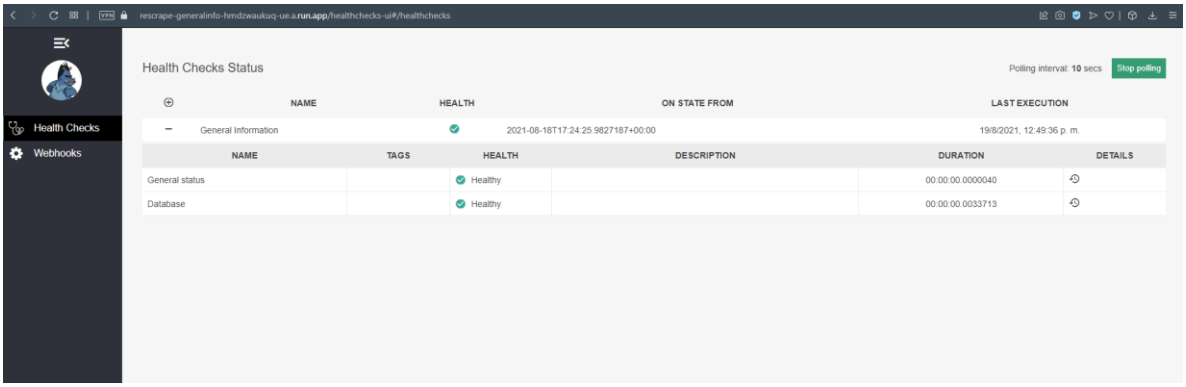

Figura 24. *Estado de salud del microservicio de información general.*

También si se requiere consumir el estado del microservicio desde una aplicación diferente se agregó un Health-check en formato JSON, el cual es accesible agregando la dirección "/hc" sin las comillas, a la URL del microservicio requerido. Por ejemplo, en la **Figura 25** se muestra la ruta del servicio General Information.

Estrategia de replicación: Para que el Scraper logre obtener la información de impuestos y documentos de una parcela es necesario que primero obtenga la información general, conforme se explicó en el modelo de datos para información de documentos (ver **Capitulo 3**). Para resolver lo anterior se hizo necesario definir diferentes estrategias.

La primera de las estrategias planteadas consistió en comunicar los microservicios Taxes y Documents con el microservicio General Information mediante proxies HTTP, lo que quiere decir, que antes de enviar la petición al Scraper para obtener los impuestos o los documentos de una parcela, se realiza una llamada HTTP al microservicio General Information para solicitarle los datos de búsqueda requeridos para lograr obtener la información. Esta manera de comunicación requiere abrir más conexiones entre los microservicios, quedando una conexión de Taxes a General Information y otra conexión de Documents a General Information cada vez que se requiera extraer la información de una parcela.

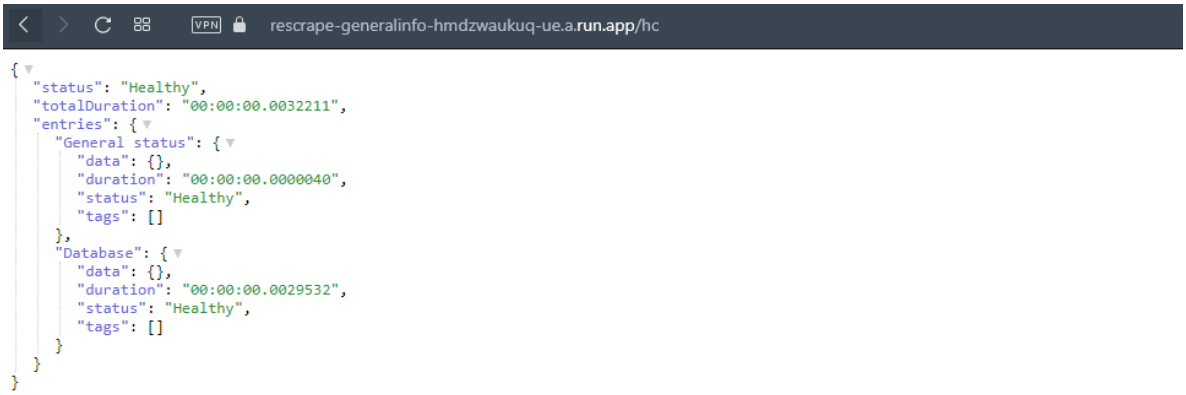

Figura 25. *Consulta REST y respuesta JSON del estado de salud de un microservicio.*

La segunda estrategia que se planteó fue la implementación de un bus de comunicaciones, en donde de manera asíncrona los microservicios Taxes y Documents envían la petición a General Information para obtener los campos requeridos y lograr obtener la información. Esta estrategia, así como la anterior no se tomó en cuenta debido a que si en algún momento no llega a estar en funcionamiento el microservicio General Information no se lograría que los demás microservicios funcionaran, debido a que para realizar una búsqueda de información en los microservicios Taxes y Documents se necesita primero hacer un llamado o enviar un mensaje a General Information.

Finalmente se optó por tomar la estrategia de replicación de la base de datos del microservicio General Information hacia la base de datos de Taxes y Documents, gracias a que, si en algún momento el microservicio de General Information no funciona, los datos obtenidos con anterioridad de las parcelas y que son necesarios para realizar una búsqueda de impuestos y/o documentos estarán replicados en la base de datos de cada microservicio y se podrán obtener los datos de impuestos y documentos de parcelas existentes sin problemas. De esta forma se logra que cada microservicio sea lo más independiente posible de los otros y en caso de no encontrarse en funcionamiento alguno de ellos, se puedan ejecutar la mayor cantidad de funciones posibles sin que necesiten comunicarse los microservicios entre sí para realizar scraping de la información. Cabe resaltar que si el servicio General Information no logra estar online no significa que su base de datos no esté online, debido a que la base de datos de cada microservicio está desplegada en una máquina diferente a la del microservicio.

Repositorio de Logs: Siguiendo con los patrones de microservicios, los microservicios propuestos en el proyecto se conectan a un repositorio de logs. El servicio de logs utilizado fue el proporcionado por Papertrail [60], debido a la posibilidad de tomar un plan gratuito mensual con buena capacidad de almacenamiento. El repositorio ayuda a rastrear cada una de las funcionalidades realizadas por los cuatro microservicios, cada vez que se procese una funcionalidad en cada uno de estos, se mandará un mensaje de Log indicando el inicio, el fin o si ocurrió un error mientras se realizaba el algoritmo de la función que se encuentre en ejecución. Esto ayuda a tener una visión general sobre el funcionamiento de la aplicación, en caso de que ocurra un error o no se logre un buen funcionamiento, se revisa el repositorio y se identifica de manera más eficiente el microservicio que está generando el problema.

Microservicio Identity: Encargado de manejar el subdominio de los usuarios de la aplicación. Este microservicio permite crear usuarios, manejar sus sesiones y crear el JSON Web Token necesario para cada una de las peticiones que se realizan desde el cliente hacia la API Gateway y desde la API Gateway a cada uno de los microservicios.

Este microservicio se encuentra disponible para cualquier cliente web que desee visitar la aplicación y realizar una solicitud de inicio de sesión. Si un cliente realiza una solicitud hacia este microservicio, la solicitud no necesitará pasar por la API Gateway, esta última está pensada para ser protegida por JSON Web Token y es la razón por la cual no se implementa como intermediadora entre un cliente y este microservicio.

Para que un cliente web obtenga una respuesta autorizada para cada petición que realice, el microservicio Identity tiene que abrirle una sesión para el usuario, esto lo realiza creando y enviándole un token válido para que el cliente lo agregue en cada una de sus peticiones, así los microservicios le devolverán una respuesta. Si un cliente obtiene un token, la aplicación Frontend automáticamente sabrá que inició una sesión para un usuario, guarda el token y lo añade en cada una de sus peticiones.

El microservicio Identity maneja su propia base de datos desplegada en una maquina distinta gracias al servicio Cloud SQL. El modelo de base de datos tiene como base el modelo planteado por la librería de Microsoft Entity Framework (microsoft.aspnetcore.identity.entityframeworkcore), la cual contiene la información necesaria para gestionar Json Web Tokens, roles de usuario, sesiones e información general del usuario como teléfono, nombre, etc. El modelo se puede visualizar en la **Figura 26**. A este modelo sólo se le modificó la tabla AspNetUsers, añadiendo los campos FirstName y LastName.

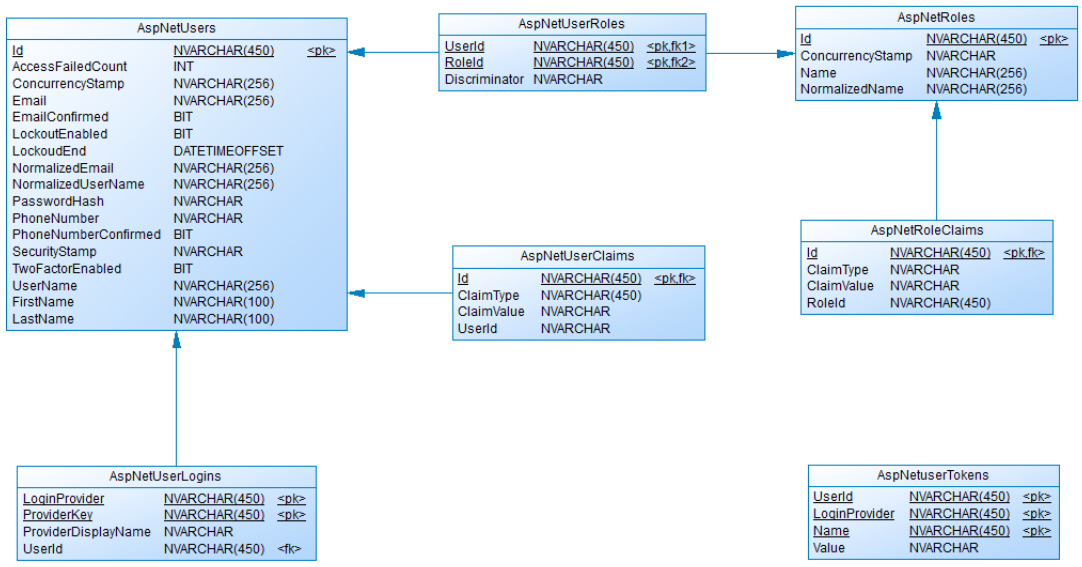

Figura 26. *Modelo de la base de datos del microservicio de Identity.*

Microservicio "General Information": Este microservicio fue desarrollado para gestionar el subdominio de información general, aquí se guarda y se procesa la información general de una parcela: estructura, características extra, precios,

ubicaciones y propietarios. El microservicio recibe peticiones provenientes del API Gateway, las procesa y si es necesario llama al programa de scraping para que obtenga información nueva de una parcela. Aquí se guardan también las diferentes instrucciones para que el scraper logre extraer información general de una parcela. Cada microservicio gestiona la información correspondiente a las instrucciones necesarias para que el scraper logre extraer la información que le compete en cada petición. En la **Figura 27** se aprecia el modelo de base de datos de este servicio. Es preciso notar que el modelo corresponde con las tablas del submodelo de información general presentado anteriormente en la **Figura 12**.

En esta base de datos se encuentra descrita la información general que se logra capturar para cada parcela desde las distintas páginas web seleccionadas. Esta base de datos además usa el patrón de publicador y distribuidor de las replicaciones que existen dentro de las bases de datos de toda la aplicación. Como se explicó anteriormente se aplicó una estrategia de replicación de datos para lograr el éxito de la aplicación, en esta base de datos se tienen dos replicaciones, una que envía los datos de las tablas Parcels, Addresses y Locations hacia el microservicio taxes, vea la **Figura 28** y otra que envía datos más detallados de la tabla Parcels y replica toda la información de las tablas Addresses y Locations hacia el microservicio Documents, vea la **Figura 29**.

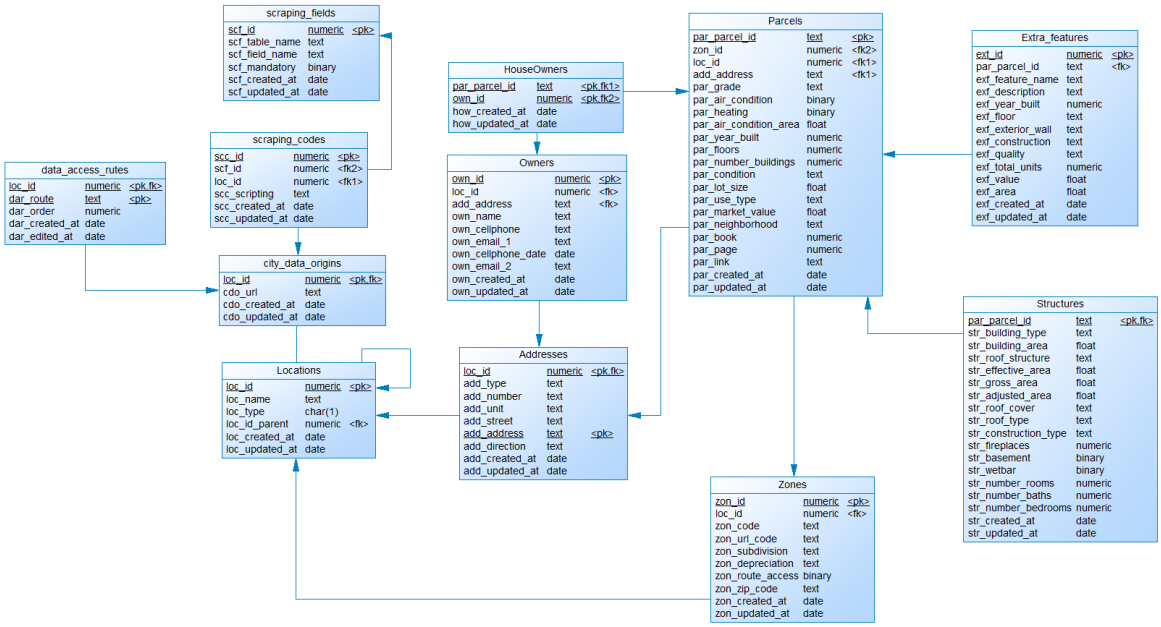

Figura 27. *Modelo de la base de Datos del microservicio de General Information.*

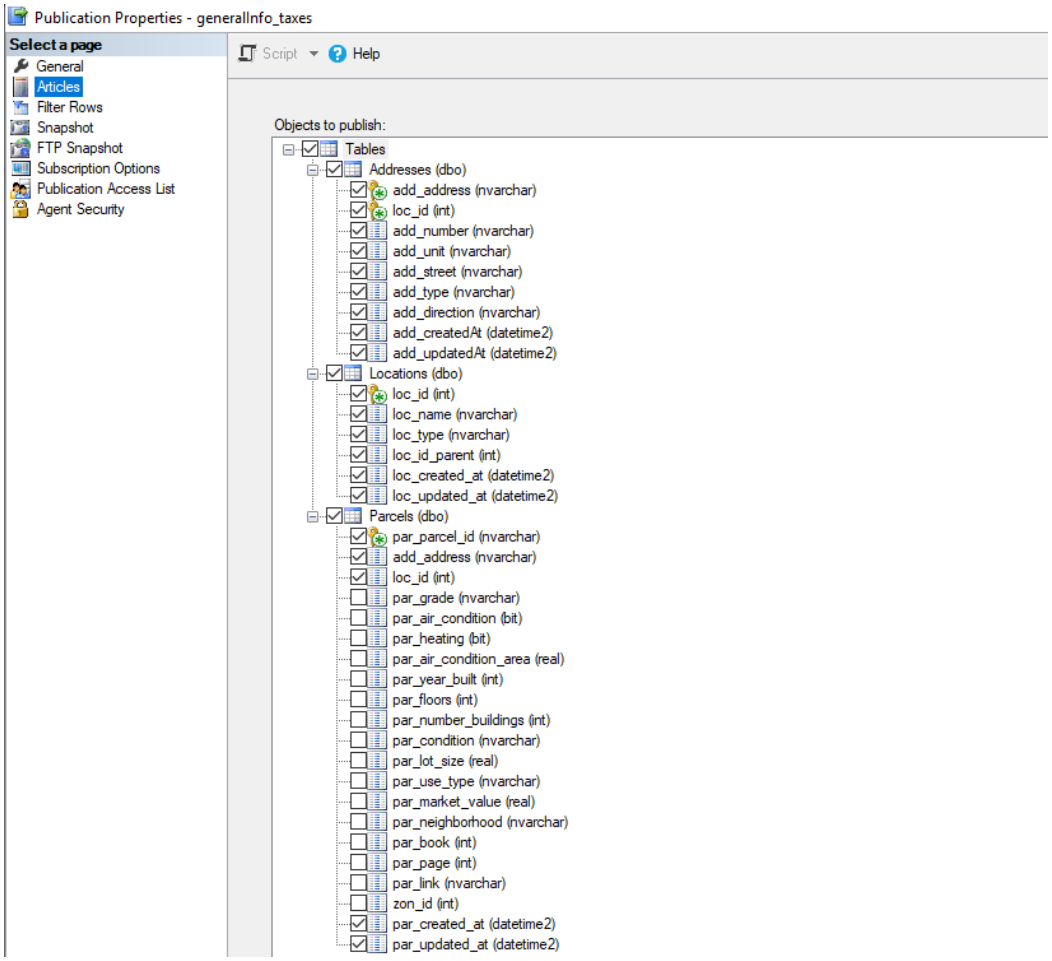

Figura 28. *Tablas y sus columnas replicadas desde GeneralInfo hacia Taxes.*

Para obtener la información de los impuestos de las parcelas se hace necesario solamente el campo par_parcel_id por lo cual se hace obligatorio replicarlo desde el microservicio GeneralInfo hacia Taxes, junto con este campo se envían algunos campos adicionales útiles para relacionar la tabla Parcels con las demás tablas del microservicio receptor.

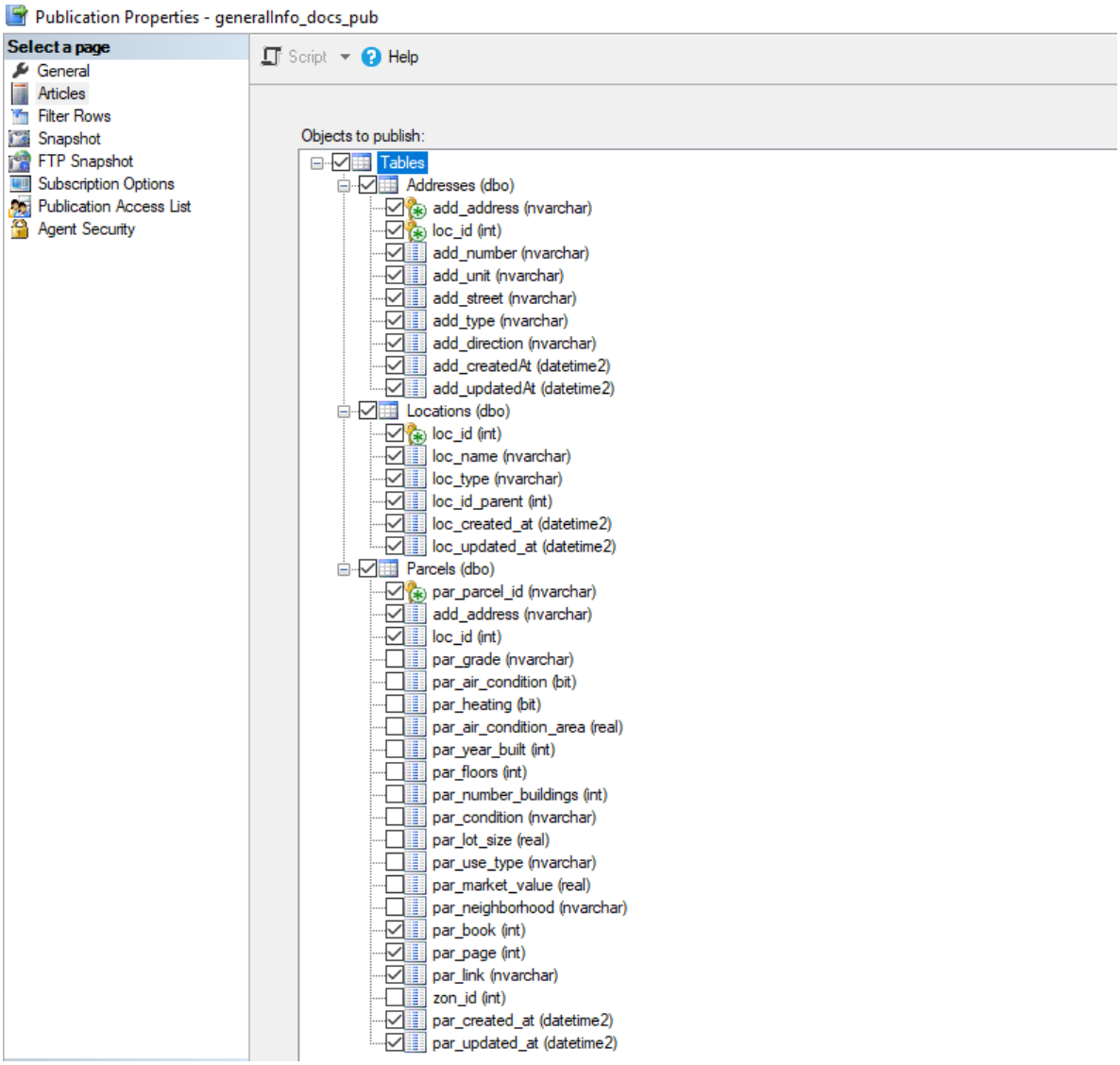

Figura 29. *Tablas y sus columnas replicadas desde GeneralInfo hacia Documents.*

Para lograr con éxito la tarea de extracción de información de documentos de una parcela se hacen necesarios los campos par_book y par_page; y en ciertos sitios web el campo par_link, por lo cual además de la llave primaria de la tabla Parcels, se replicaron estos campos mencionados hacia la base de datos del microservicio Documents, asimismo se replicaron los campos necesarios para relacionar la información básica de la parcela con las tablas del microservicio de destino.

La aplicación fue construida bajo el entorno de desarrollo .Net Core. Está desplegada en un contenedor Docker en el servicio Cloud Run de Google Cloud. La base de datos de esta API esta alojada en una maquina aparte, y a diferencia del microservicio Identity, se utilizó el servicio Compute Engine de Google Cloud, obligándonos a configurar una máquina para poner en línea la base de datos, la razón de esto es la imposibilidad de replicación de manera sencilla mediante el servicio Cloud SQL de Google Cloud, donde se ofrece una posibilidad de replicar

bases de datos completas, mas no tablas y datos específicos. Además, en el servicio Cloud SQL no se permite asignar a un usuario permisos de sys_admin, los cuales son necesarios para replicar la base de datos, por lo cual se tomó la tarea de configurar nuestra una máquina virtual propia para la base de datos. La base de datos que se configuró también es Microsoft SQL Server.

Microservicio Taxes: Este microservicio gestiona los datos de impuestos para las parcelas, se guarda año por año la información de pagos y deudas que tiene una parcela, permitiendo tener un histórico de la propiedad. Estos datos son útiles para un agente inmobiliario ya que puede saber cuánto dinero debe la parcela, cuánto dinero se debe pagar por año, entre otros, para apoyar la toma de la decisión de una compra inmobiliaria.

Cada vez que se realiza scraping de información general, el microservicio GeneralInfo guarda los datos obtenidos en su base de datos, luego, los datos necesarios son replicados a este microservicio para lograr hacer scraping de impuestos. Una vez la información se replica entonces el microservicio estará en la capacidad de enviar al scraper los datos necesarios para que se obtenga la información correctamente.

Al igual que el microservicio GeneralInfo, este microservicio contiene el submodelo correspondiente a la configuración del scraper, lográndose comunicar con el scraper y con la API Gateway.

La base de datos guarda la información correspondiente a este microservicio, y está configurada para suscribirse a la publicación correspondiente que el microservicio GeneralInfo le brinda (Ver figura **Figura 30**).

La aplicación fue construida bajo el entorno de desarrollo .Net Core. Está desplegada en un contenedor Docker en el servicio Cloud Run de Google Cloud. La base de datos de esta API esta alojada en una maquina aparte, al igual que la base de datos de GeneralInfo, se prefiere crear un entorno propio para configurar la base de datos de manera menos limitada a la ofrecida por Google Cloud.

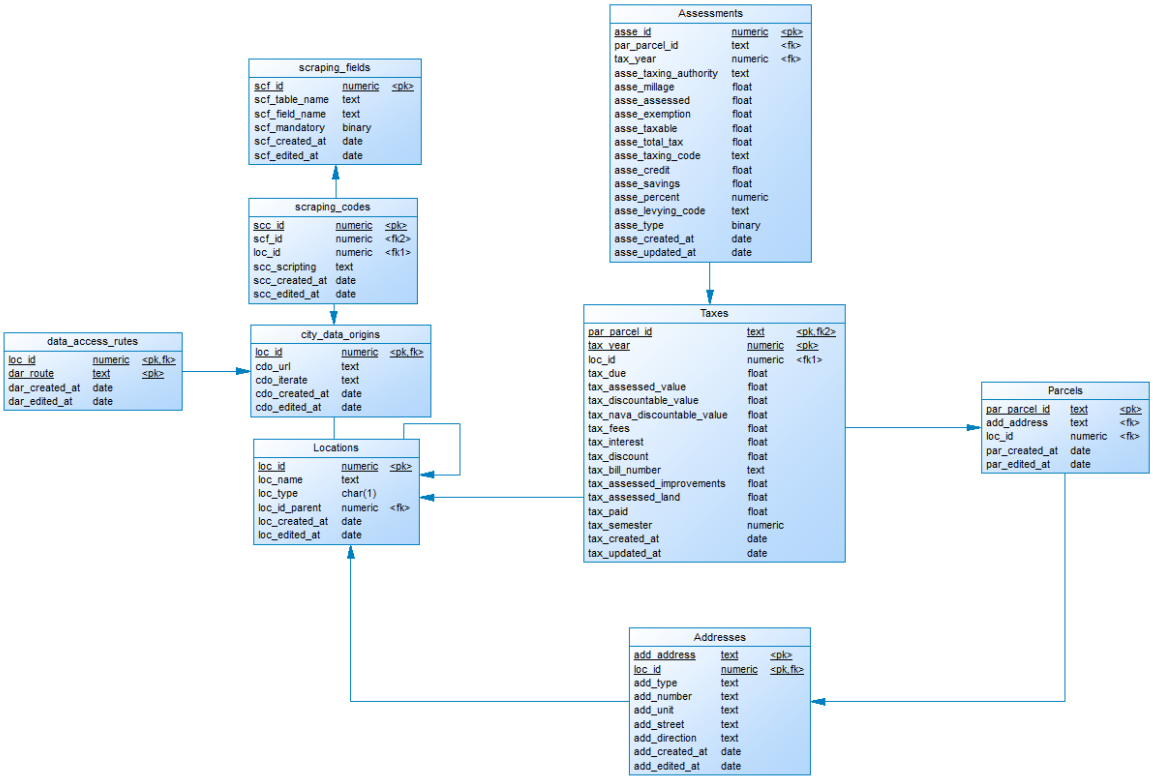

Figura 30. *Modelo de la base de datos del microservicio de impuestos (Taxes).*

Microservicio Documents: A este corresponde gestionar los datos de documentos de una parcela, de acuerdo con la página desde donde se extrae la información existen diferentes tipos de documentos que se logran extraer, entre los más usuales se tienen las escrituras y las hipotecas. La información que contienen estos documentos es útil para conocer la real pertenencia de una parcela a una persona, también es útil para conocer si una propiedad tiene deudas hipotecarias, lo cual es uno de los factores más buscados por los agentes inmobiliarios. Una vez realizado el scraping de información general, el microservicio GeneralInfo guarda los datos obtenidos en su base de datos y los replica a este microservicio para que permanezca listo cuando se requiera extraer la información por medio del scraper. Una vez la información se replica y se envíe la petición para actualizar o extraer información de documentos, el microservicio enviará otra petición al scraper junto con los datos necesarios para que se obtenga la información satisfactoriamente.

Al igual que el microservicio GeneralInfo y Taxes, este microservicio contiene el modelo correspondiente a la configuración del scraper, permitiendo una configuración propia de las instrucciones que se envían a este.

El microservicio Documents tiene una base de datos encargada de guardar los datos correspondientes. Al igual que Taxes, la base de datos está suscrita a la publicación de replicación que viene desde la base de datos del microservicio

GeneralInfo y que corresponde a este microservicio. La información que se guarda en la base de datos se muestra en la figura **Figura 31**.

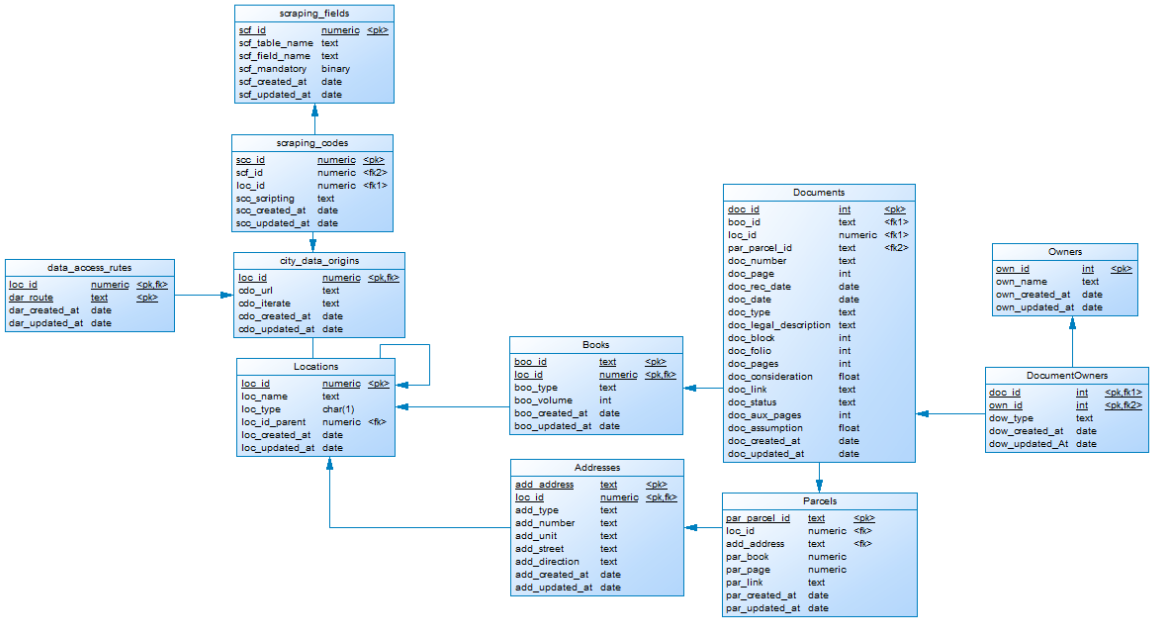

Figura 31. *Modelo de la base de datos del microservicio de documentos (Documents).*

La aplicación fue construida bajo las mismas tecnologías y condiciones del microservicio de Taxes. El entorno de desarrollo es .Net Core, está desplegada en un contenedor Docker en el servicio Cloud Run de Google Cloud. La base de datos de esta API esta alojada en una maquina aparte, al igual que la base de datos de GeneralInfo y Taxes, cuenta con un entorno propio para configurar la base de datos de manera menos limitada a la ofrecida por Google Cloud. La base de datos que se configuró también es Microsoft SQL Server.

API Gateway: Esta API se creó de acuerdo con los patrones de arquitectura de microservicios mencionados, donde un cliente Frontend solamente se comunica con una aplicación en el dominio Backend, lo que hace más sencillo y seguro el proceso de conexión a los diferentes microservicios. La aplicación Frontend solamente interactúa con una API Gateway que contiene las rutas de los microservicios, así no se hace necesario que el cliente Frontend conozca todas las rutas para llegar a los microservicios, ni tiene acceso directo con cada uno de estos. Para la aplicación, se desarrolló una API Gateway capaz de recibir las peticiones que lleguen desde un cliente web o móvil, pero la aplicación está construida para fácilmente adaptarse si se requiere implementar un Gateway diferente para cada cliente.

Cada vez que se realice una petición desde un cliente, esta será recibida por la API Gateway, la cual será la única que conoce las direcciones de cada uno de los microservicios y estará encargada de redireccionar al correspondiente, devolviendo la respuesta de dicha petición al cliente.

Esta API no posee una base de datos, su única función es redireccionar al cliente, no posee información almacenada adicional a la necesaria para su labor.

Al igual que las aplicaciones anteriormente explicadas del dominio Backend, esta fue construida bajo el entorno de desarrollo .Net Core y desplegada en un contenedor Docker en el servicio Cloud Run de Google Cloud.

4.1.3 Scraper

El scraper es el programa encargado de obtener los datos especificados y configurados desde la interfaz de la aplicación web, esto lo realiza para las diferentes páginas web de Real Estate de las diferentes ciudades y condados de Estados Unidos. Lo anterior se logra “simulando” el comportamiento humano para acceder, navegar e interactuar con las páginas fuente de información. Se desarrolló haciendo uso del framework Selenium en Python, y está constituido por tres componentes fundamentales. El primero son los mismos scrapers, representados por clases específicas para cada microservicio con características diferentes, es decir que existe un scraper propio para GeneralInfo, otro para Taxes y otro para Documents. Estos scrapers hacen uso del segundo componente fundamental que es el procesador de lenguaje, esto debido a que fue necesario la creación de un lenguaje que permitiera simplificar y facilitar la creación y configuración de instrucciones de un usuario de la aplicación, para indicarle al scraper cada una de las acciones que debe realizar sobre las páginas web y así lograr conseguir la información. Este procesador de lenguaje está a su vez divido en dos sub procesadores enfocados en una tarea en específico: uno encargado de las acciones de interacción para realizar búsquedas en las páginas web, y otro procesador enfocado en buscar los elementos en pantalla y extraer su información. El tercer elemento fundamental es el denominado Pool de drivers, que es el componente encargado de administrar el uso de los diferentes drivers del navegador que los scrapers necesitan para poder realizar la ejecución del proceso de web scraping.

La forma en que el Backend hace uso de las funciones del scraper es a través de un servicio REST. Para esto, en el programa scraper se implementó un servicio HTTP con el microframework Flask para Python. Esto a razón de que cada programa scraper requiere instancias del navegador Google Chrome, el cual consume una parte muy importante de los recursos de la maquina servidor, por lo que se optó por hacer uso de un framework muy liviano para la comunicación con los scrapers.

Scraper Core: Como se mencionó anteriormente, el núcleo del scraper está constituido por tres programas scraper distintos, cada uno enfocado y especializado en su propio propósito de datos, estos son: GeneralInfo Scraper, Taxes Scraper y Documents Scraper. Lógicamente están definidos de la siguiente manera: una super clase denominada Scraper, que es donde que se definen los métodos y propiedades básicas y comunes de un scraper, como lo son la configuración de los

tiempos de espera para el renderizado y carga de información de las páginas web, configuraciones propias del navegador como el modo de visualización, e incluso configurar diferentes user-agents y proxies para cada driver en cada scraper. Las subclases que heredan y especifican las funcionalidades particulares para cada tipo de información son GeneralInfoScraper, TaxesScraper y DocumentsScraper.

GeneralInfo Scraper: Este es el scraper principal y a partir del cual, gracias a la información que logra obtener, se puede empezar a extraer la información con los otros dos scrapers. Además de ser el más importante, también es el más básico en su implementación, dado que, a diferencia de las páginas de impuestos y documentos, la información presentada en las páginas de información general suele ser desplegada en una sola pantalla después de la búsqueda, es decir, este scraper no requiere interactuar con nada más en dicha pantalla.

Los datos que este scraper es capaz de extraer son los mismos atributos especificados en el modelo y la base de datos en la tabla GeneralInfo. Entre los cuales se encuentran datos de los propietarios como el nombre y dirección, información sobre las características del terreno y la estructura de la propiedad, entre otros.

Taxes Scraper: Este scraper toma como base importante la implementación lógica con la que se realizó el scraper de información general. Sin embargo, tiene una diferencia importante en la serie de acciones que se requieren para llegar a los datos finales a partir de un simple formulario de búsqueda. Esto debido a que en todas las páginas y portales de impuestos, la información de una única propiedad se presenta en forma de un historial de varios años, lo que resulta en una lista extensa en la mayoría de los casos que a simple vista mostraban información demasiado básica, como simplemente el año del impuesto, y era necesario hacer clic sobre cada uno de esos elementos de la lista para poder desplegar y visualizar otra ventana en donde se mostraba toda la información detallada de los impuestos de esa propiedad en un año en particular. Entre los atributos más comunes que se pueden extraer con este scraper está el año del impuesto, su costo, el total pagado y la deuda total.

En muchas ocasiones, para poder empezar a realizar web scraping con este scraper se dependía completamente de los resultados obtenidos con el scraper de GeneralInfo, debido a que en los formularios de búsqueda de las páginas de impuestos era necesario contar con el número de libro y página de los registros de una propiedad para lograr obtener resultados precisos. Estos datos en la mayoría de los casos sólo podían ser conseguidos en las páginas de información general.

Documents Scraper: Este programa scraper es el encargado de extraer los datos de los documentos de hipotecas y escrituras de las parcelas. La implementación de este scraper es más parecida a la del scraper de Taxes que a la de GeneralInfo, debido a que la información de los documentos es igualmente presentada como una lista larga de elementos que son el histórico de documentos que se han registrado para una propiedad en particular.

De nuevo, en la mayoría de los casos, el correcto funcionamiento y éxito en la obtención de la información de este scraper tiene como precondición la presencia de algunos datos y atributos que se obtienen con el scraper de GeneralInfo.

A diferencia de las respuestas obtenidas con el scraper de GeneralInfo, que se presentan únicamente en forma de un objeto JSON y sus propiedades específicas, las respuestas obtenidas con este scraper son listas o vectores con objetos y sus propiedades o atributos.

Drivers Pool: El hecho de que el funcionamiento de un scraper con Selenium dependa enteramente de un driver, hace plantear una serie de consideraciones para tener en cuenta. Primero, es necesario aclarar que cuando se habla del driver, se hace referencia a una instancia de un navegador (Chrome o Firefox) modificado o especializado en llevar a cabo un servicio de web scraping, por lo que el consumo de recursos de un driver en la máquina es muy similar al de un navegador Chrome o Firefox común y normal. La implementación de tecnologías para soportar páginas web más dinámicas e interactivas como AJAX propician que los navegadores modernos suelan consumir una importante cantidad de recursos de memoria RAM y procesador, por lo que el factor de rendimiento y optimización fue determinante en la creación de este componente.

En la concepción inicial, se planteó que por cada petición HTTP recibida desde un servicio externo que quiera utilizar el servicio de web scraping, se crearía una instancia de scraper, la cual a su vez ejecutaría un nuevo driver para llevar a cabo la extracción de la información a través del navegador web. Una vez terminado el proceso de extracción, se eliminaría la instancia del scraper y a su vez se daría una señal de cierre al driver para liberar la memoria. Aunque esta estrategia inicial garantizaba que no se consumirían recursos innecesariamente mientras no se estuvieran usando, se tuvo un impacto negativo considerable en el rendimiento y tiempo de respuesta de cada petición, debido a que la creación de un nuevo driver para cada llamada implicaba abrir el navegador web como si fuera la primera vez, lo que a su vez tenía como consecuencia que el sistema operativo tuviera que cargar los archivos e información necesaria para ejecutar un navegador y dejarlo listo para su uso, comportamiento que puede notarse en cualquier computador personal cuando ejecuta un navegador web por primera vez después de su encendido.

Debido a lo anterior, se hizo necesario buscar una solución que garantizara que cada proceso de web scraping se inicie lo más rápido posible, evitando el tiempo de carga inicial del web driver por cada petición. A partir de esto surge la idea de crear un componente denominado pool de drivers, que es el encargado de almacenar y administrar una lista de web drivers activos, que solo se inician una única vez en el momento en que se ejecuta el servidor del scraper, y permanecen abiertos y en espera de ser usados en el momento en que se realice una petición de scraping. Esto a su vez es útil para gestionar varias peticiones que puedan ser ejecutadas

simultáneamente, en el caso de que, exista más de un usuario utilizando la aplicación.

Así pues, el pool de drivers ejecuta inicialmente una cantidad de web drivers que debe ser acorde a la capacidad de las características del servidor. Un servidor que, por ejemplo, cuente con 4 GB de memoria RAM podría ejecutar óptimamente entre 4 y máximo 6 drivers al mismo tiempo. El pool de drivers se encarga también de la gestión de sus estados, para poder identificar cuándo un driver está siendo utilizado por un scraper y cuándo acaba de ser liberado. El proceso se describe a continuación.

El servicio REST recibe una petición HTTP por parte de un microservicio para comenzar a realizar la extracción de datos con los parámetros suministrados. En ese momento se le pregunta al pool de drivers por algún navegador web de su lista de drivers que esté disponible. El pool de drivers verifica en su estructura de datos cuáles drivers están disponibles y devuelve el primero que esté libre y lo marca como ocupado. En ese momento se crea la instancia de un Scraper (de GeneralInfo, Taxes o Documents) que recibe como parámetro de creación la referencia a un driver. El scraper recién creado hace uso del driver para interactuar con las páginas web de Real Estate y extraer los datos que encuentre en ellas. Al finalizar el proceso de extracción, el scraper manda una señal al pool de drivers para que éste se encargue de marcar el driver que se acaba de usar como disponible otra vez y lo deje listo para ser usado en la siguiente petición que se realice.

En el momento en que llegue una nueva petición para el web scraper y no se encuentre ningún driver disponible, esta petición se pone en espera y con un proceso que cada determinado tiempo (cada dos segundos, por ejemplo), le pregunta al pool de drivers por la disponibilidad de alguno de estos. En cuanto uno de los scrapers termina su ejecución y queda disponible, la petición en espera procede a usar ese driver. De esta manera los drivers siempre estarán listos para su uso desde el comienzo de la aplicación y no tendrán que ser creados ni destruidos con cada petición de realización de web scraping.

Language Processor: Debido a que la construcción del scraper con el framework Selenium requiere de cierto nivel de conocimiento avanzado, tanto en programación, en el uso del lenguaje Python y en el uso de las mismas funciones y propiedades ofrecidas por el framework, se determinó como conveniente y oportuno la creación de un lenguaje de instrucciones que fuera mucho más entendible y accesible para un usuario conocedor de HTML, CSS y en algunos caos de JavaScript (aunque con una interfaz apropiada esto se puede realizar sin estos conocimientos, hecho que se considera como un trabajo futuro de la presente tesis). Está lenguaje está especificado en detalle en el **Anexo B**.

El propósito de este lenguaje es simplificar considerablemente el conocimiento y el uso exhaustivo de las funciones y herramientas de Selenium para hacer web

scraping, resumiendo procesos complejos de implementación con el framework en una sola instrucción, con una sintaxis más entendible.

Durante el proceso de desarrollo de este lenguaje se pudo notar un comportamiento diferente al realizar ciertas acciones con el framework. Estas básicamente se dividen en dos tipos de acción: aquellas que involucran interacción con algún elemento de la página web (por ejemplo, hacer clic sobre un elemento o insertar texto en campos de entrada) y aquellas cuyo único propósito eran encontrar un elemento y extraer el texto que contenía. Esto hizo que a nivel lógico la implementación del procesador del lenguaje sea en forma de una superclase LanguageProcessor, que contiene los métodos más genéricos del procesador de lenguaje, y dos subclases especializadas, una denominada FolioAccessProcessor y otra denominada PathProcessor.

FolioAccessProcessor: Es el componente del procesador de lenguaje especializado en llevar a cabo instrucciones que impliquen interacción con las páginas web, como por ejemplo hacer clic en un botón, introducir texto en un campo de entrada de algún formulario, entre otras. La razón de crear un componente especializado para este propósito es que el comportamiento de un objeto Selenium se altera y retoma otro valor de referencia después de haberse realizado un clic sobre este, estos objetos generalmente representan una referencia a alguna sección o elemento del documento HTML de la página web. Lo que impide navegar y acceder a otros elementos dentro de la estructura HTML a partir de este objeto utilizado. El proceso empleado para evitar perder las diferentes referencias a la información de los distintos objetos se explica a continuación.

Este procesador de lenguaje es más ampliamente usado en los scrapers de taxes y documents, debido al gran nivel de interacción que se requiere en las páginas web para poder navegar por la lista de los historiales de las propiedades.

En el siguiente ejemplo se muestran la serie de instrucciones necesarias para acceder a los registros de impuestos de una propiedad dentro de una página para la ciudad de Fort Lauderdale. La **Figura 32** muestra las instrucciones que en su orden debe seguir el Scraper una vez ingrese a la página y así lograr llegar a las distintas páginas que contienen la fuente de datos.

```
"data_access_route": [
    "findById:search_query:clear:putText:propertyId",
    "findByClassName:btn-primary:click",
    "findByClassName:results:findByTagName:a:click",
    "findByStringExact:Account history/findNextSibling:div/findSelector:table/findAllByClassName:description"
]
```

Figura 32. *Array de instrucciones de acceso al folio en página de taxes.*

La **Figura 33** muestra las instrucciones que en su orden son necesarias para interactuar con la página y realizar la búsqueda de la información. Más adelante se muestran y explican otras de estas instrucciones que se envían al scraper y que son útiles para su éxito.

```
"search_parameters": [
    {
        "propertyId": "494234014270",
        "owner_name": "STRINGER, STEPHANIE JO",
        "address": "1620 NW 7 TERRACE FORT LAUDERDALE, 33311"
    }
]
```


Figura 33. *Objeto JSON con los parámetros necesarios para realizar la búsqueda.*

A continuación, se explica paso a paso cada una de las instrucciones que se muestran en la **Figura 32** y la **Figura 33** y cómo gracias a estas y otras instrucciones, se logra llegar a cada una de las páginas que contienen toda la información que va a ser extraída.

- **findById:**search_query:clear:**putText:**propertyId
  En esta instrucción inicialmente se busca un elemento cuyo valor en la propiedad id sea “search_query”. A continuación, por medio de la instrucción de entrada “clear” se le ordena al scraper que limpie el contenido que pueda tener el elemento encontrado. El siguiente argumento de la instrucción es otra instrucción: **putText** por medio de la cual, a ese elemento encontrado y limpiado, ahora se le ingresará una cadena de caracteres con el valor especificado en el atributo “propertyId” del objeto de parámetros de búsqueda (“search_parameters”) que se envía en el cuerpo de la petición (ver **Figura 34**).

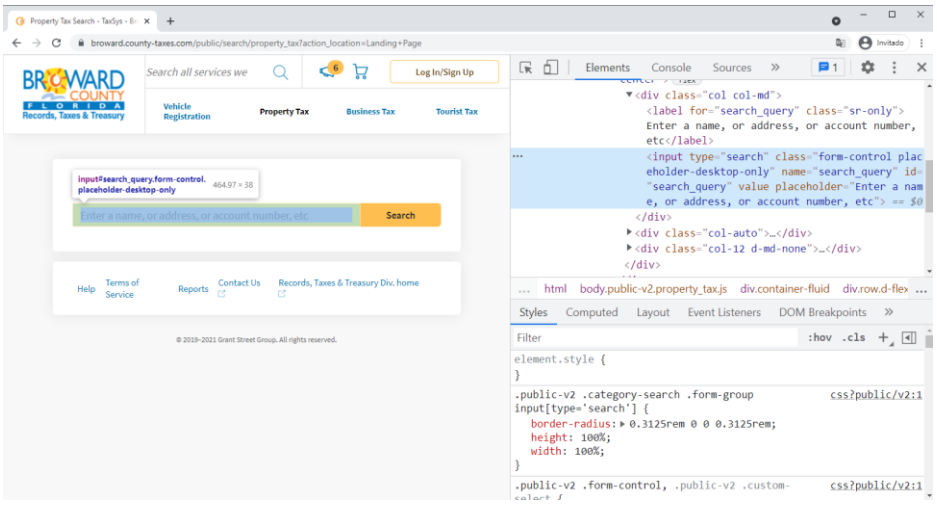

Figura 34. *Obtención del elemento “propertyId”.*

- **findByClassName:**btn-primary:click

Una vez ingresado el texto en el cuadro de búsqueda, esta instrucción se encarga de encontrar un elemento cuyo valor en su atributo “class” sea “btn-primary”. Acto seguido, por medio de la instrucción clic se ordena al scraper realizar una acción de clic sobre ese elemento encontrado (ver **Figura 35**).

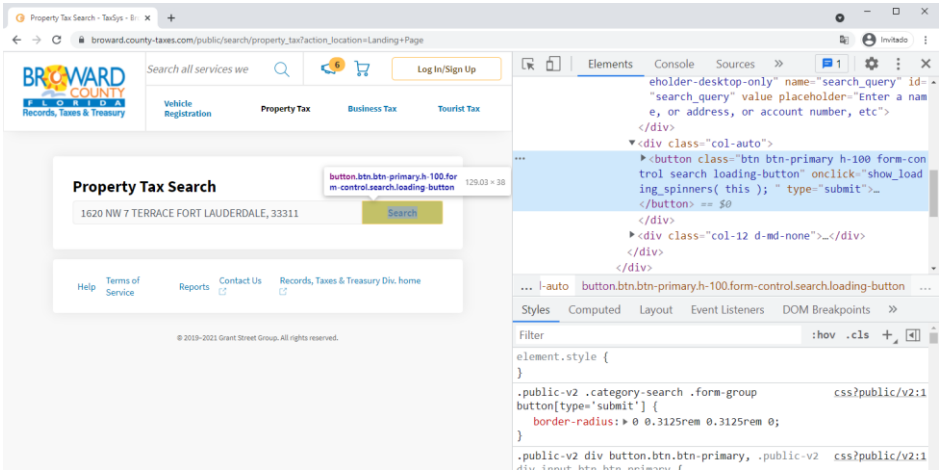

Figura 35. *Obtención del elemento “btn-primary”.*

- **findByClassName:**results:**findByTagName:**a:click
  Debido a que, para esta página en particular, al hacer clic sobre la búsqueda, el resultado es una lista con al menos un elemento en donde se puede acceder al registro encontrado, se hace necesario realizar otro clic para ingresar a la información completa sobre esa propiedad. Primero se busca un elemento con la clase “results”, a partir de este se busca otro elemento cuyo nombre de etiqueta sea una “a”, para finalmente hacer clic sobre este (ver **Figura 36**).

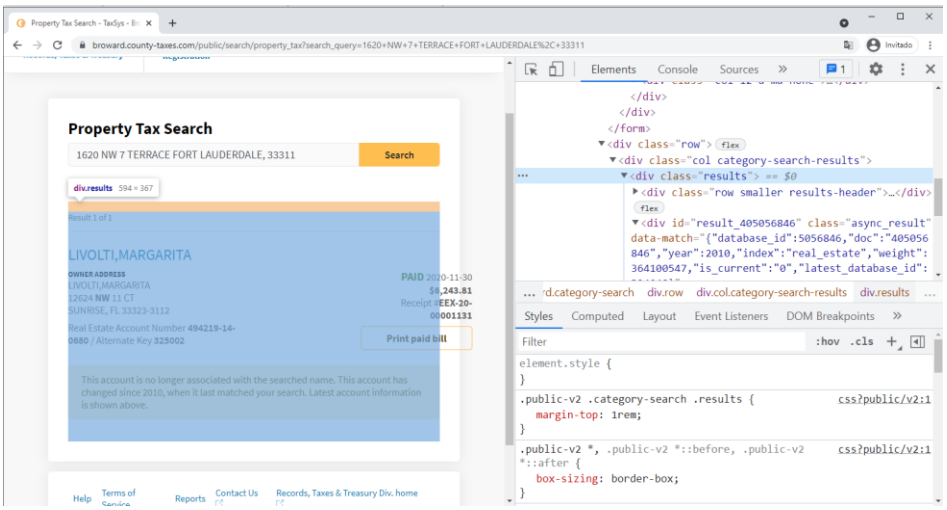

Figura 36. *Obtención del elemento con clase “result”.*

- **findByStringExact:**Account history/**findNextSibling:**div/**findSelector:**table/**findAllByClassName:**description

  Una vez se ha ingresado al resultado encontrado, la pantalla presenta una lista que representa el historial de impuestos calculados para esa propiedad en cada año. Esta instrucción se encarga de dejar lista la referencia a cada uno de los elementos de esa lista para su posterior procesamiento: Primero se obtiene un elemento que contenga exactamente un texto con el valor “Account history”, a partir de este se encuentra al primer elemento hermano (o del mismo nivel) que sea de tipo “div”. Dentro de este último elemento encontrado ahora se busca otro elemento cuyo valor de selector o etiqueta sea un “table”. Finalmente, se le indica al scraper que encuentre todos los elementos cuyo nombre de clase contengan “description” (ver **Figura 37**).

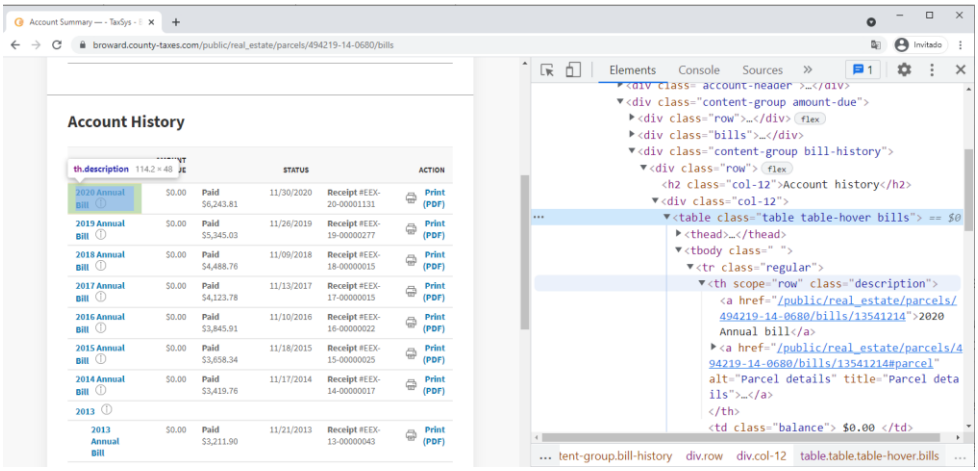

Figura 37. *Obtención del elemento con clase “description”.*

Ahora bien, lo que se necesita para obtener la información objetivo, es ingresar a cada uno de los elementos obtenidos en el paso anterior e interactuar con ellos. Para eso, dentro del cuerpo de la petición de este servicio se definió un parámetro denominado “iterate”, este será agregado a las instrucciones adicionales que se prometieron mencionar anteriormente, y dentro de este parámetro, se especifica la acción que se realizará sobre cada uno de los elementos obtenidos en el anterior paso.

```
"iterate": "findSelector:a/openInNewTab"
```

La instrucción anterior ordena al scraper que, a partir de cada uno de los elementos obtenidos de la tabla, primero se encuentre un elemento dentro de ellos cuyo nombre de etiqueta sea un “a”, y sobre ese elemento se ejecute una acción en la cual se obliga a abrir el contenido de ese elemento en una nueva pestaña del navegador. Esto con la intención de no interactuar sobre una sola pantalla, porque esto implicaría que, al hacer clic sobre un elemento para abrirlo, este actualizaría

toda la vista de la página, haciendo que muchas de las referencias obtenidas anteriormente se pierdan porque ya no se presentan en pantalla. El resultado de uno de los elementos de la lista abierto en una nueva pestaña del navegador se muestra en la **Figura 38**.

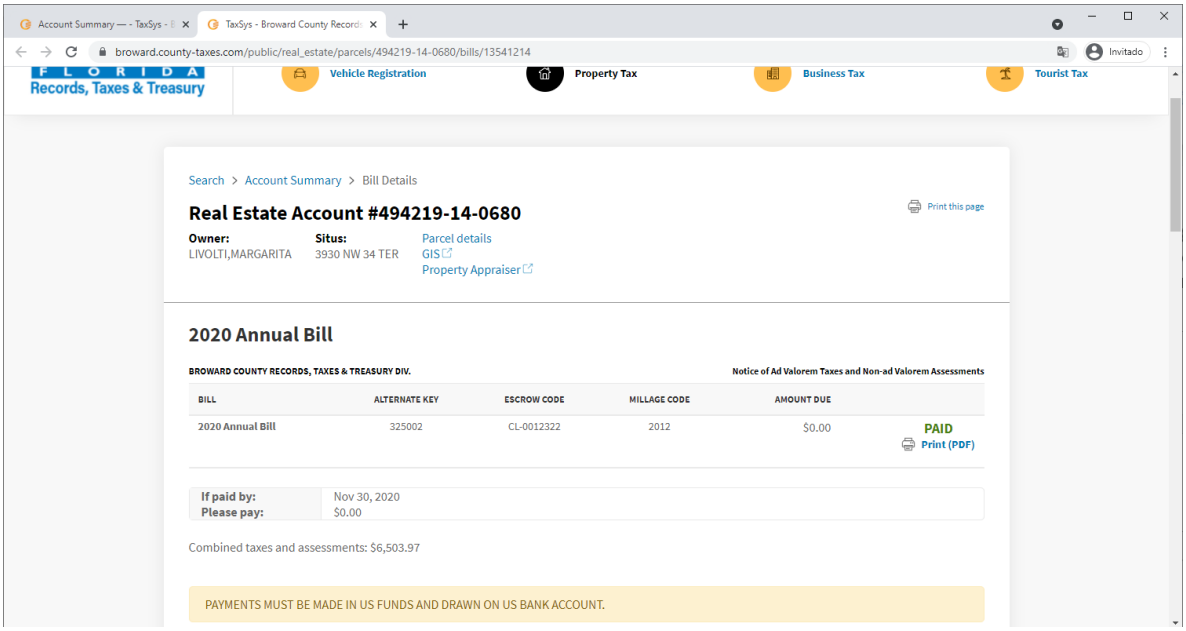

Figura 38. *Pestaña de navegador abierta con información de uno de los elementos de la tabla de impuestos.*

Una vez llegada a la fuente de los datos requeridos, se procede a obtener los atributos con las instrucciones seleccionadas. Para este propósito se hace uso del Path Processor, dado que solo se requiere extraer datos y no interactuar con ningún elemento en pantalla. La **Figura 39** muestra las instrucciones requeridas para obtener los datos (no se explican, puesto que más adelante se presenta en detalle el trabajo del PathProcessor). Como lo que se está haciendo es un ciclo iterativo para obtener la información, el resultado final de este proceso de scraping será una lista de objetos con información sobre impuestos.

```
"routes": [
    {
        "tax_year": "findById:bill/getString",
        "tax_paid": "findByClassName:col-12 col-md-4 bill-details/findChildAtPosition:14/splitString:line_break:1",
        "tax_due": "findByClassName:table table-hover bills/findByClassName:balance for-cart/getString"
    }
]
```

Figura 39. *Arreglo de instrucciones para obtención de atributos de impuestos.*

PathProcessor: Esta clase se centra en implementar la lógica de las instrucciones que indican cómo llegar a través de la estructura y los elementos dentro del documento HTML hasta un objetivo específico y extraer los datos que este contenga. A diferencia de la clase FolioAccessProcessor, este componente provee funcionalidades más avanzadas y específicas para la extracción de datos, como por ejemplo instrucciones con las cuales se puede extraer una lista de nombres o la

información contenida en varios elementos similares, o traer solo una sección específica de una cadena de caracteres dividida por un símbolo especificado.

El uso de este componente de procesamiento de lenguaje es imperante en todos los scrapers, e incluso en flujos específicos, las funciones de este procesador son utilizadas por el FolioAccesProcessor para llegar hasta ciertos elementos de la página y poder empezar a interactuar con estas.

La estrategia de la cual hace uso este componente para procesar una cadena de instrucciones es mediante un ciclo recursivo, en el que para llegar de un elemento "A" a un elemento "B" dentro del documento, se ejecutan secuencialmente cada una de las instrucciones de la cadena separándolas por un slash "/". El resultado del procesamiento de una instrucción es la entrada del siguiente ciclo, para empezar a buscar un nuevo elemento a partir del último encontrado, hasta llegar a la última instrucción y finalmente encontrar el dato o la información objetivo en forma de "string". El siguiente es el ejemplo de una instrucción para extraer un campo.

```
"add_address":"findById:situsAddressId/findBySelectorName:a/getString"
```

La sentencia anterior representa la secuencia de instrucciones necesarias para extraer el campo add_address. De la cadena de instrucciones se deducen tres instrucciones separadas por barras inclinadas:

- **findById:**situsAddressId
  En esta instrucción se le indica al scraper que encuentre dentro del documento HTML un elemento cuyo valor de la propiedad id sea "situsAddressId" (ver **Figura 40**).

- **findBySelectorName:**a
  Con esta instrucción se está indicando que, a partir del elemento encontrado anteriormente, se encuentre un nuevo elemento cuyo nombre de etiqueta HTML sea "a", es decir: <a> (ver **Figura 41**).

- **getString**
  Una vez estando en la referencia del último elemento obtenido en la instrucción anterior, por medio de esta instrucción se obtiene el texto contenido dentro del elemento en cuestión. De la anterior manera, se pudo obtener el dato correspondiente al campo add_address.

  Existen casos particulares en los que es necesario el uso de una función un poco más avanzada, como en la **Figura 42**, en el que el atributo específico que se quiere obtener está contenido dentro de una cadena de datos delimitada por un carácter separador como la barra inclinada.

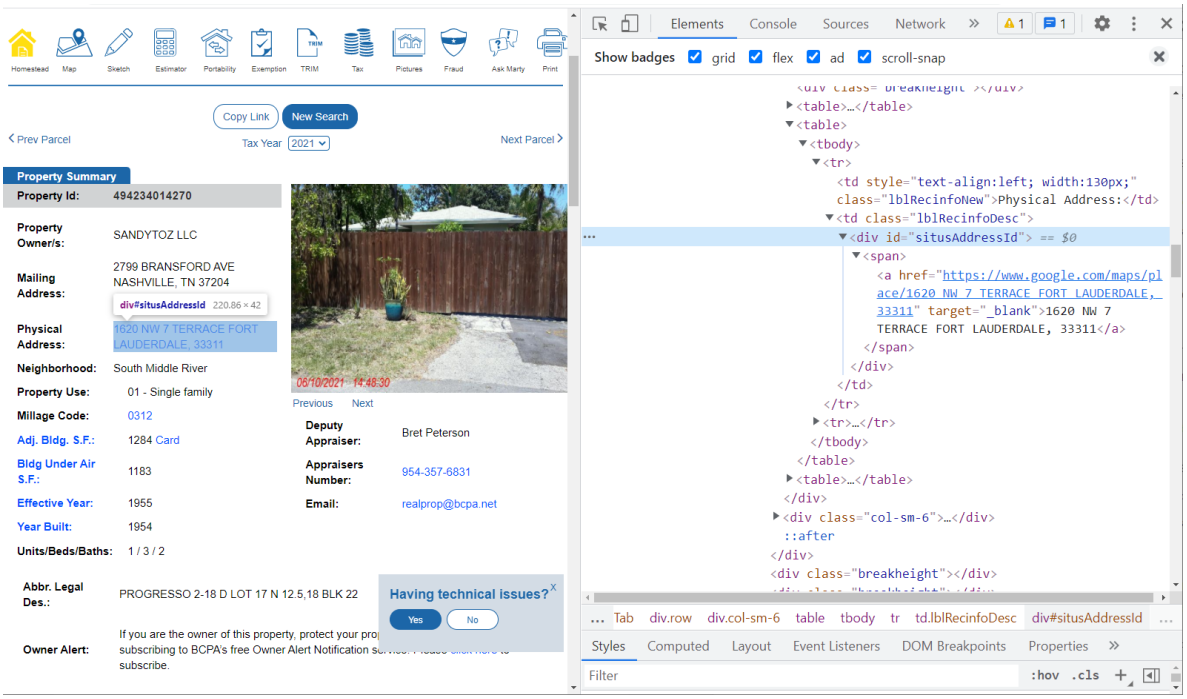

Figura 40. *Obtención del elemento con id "situsAddressId".*

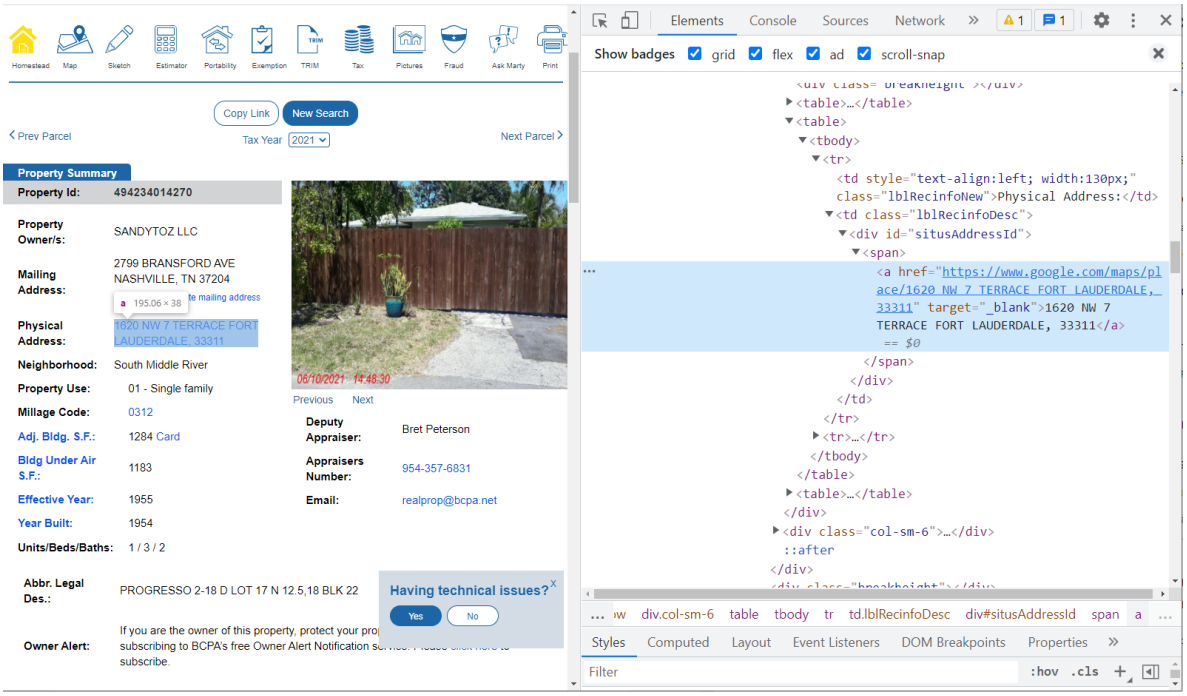

Figura 41. *Obtención del elemento con etiqueta "a" en la estructura HTML de la página web.*

Nótese que los atributos Units, Beds, Baths, se muestran dentro de una misma línea, separados por un slash: 1 / 3 / 2. Si, por ejemplo, se quiere obtener solo el número de baños, se hace uso de la propiedad **splitString** para dividir la cadena y obtener uno de los resultados productos de la división. El siguiente ejemplo muestra la instrucción requerida para extraer dicho atributo.

Figura 42. *Elemento "unitBedsBathsId" con datos seccionados en la página web.*

```
"str_number_baths":"findById:unitsBedsBathsId/splitString:slash:2"
```

- **splitString:**slash:2
  Esta instrucción indica que una vez ubicado sobre el elemento con el id "unitsBedsBathsId", se obtenga la cadena contenida dentro de ese elemento, se haga la división de esta usando el slash como carácter de división, y se obtenga el dato en la posición dos de ese resultado. Así, el resultado de la instrucción para la cadena "1 / 3 / 2" sería el número 3 (índice de base 1 que es más natural para el usuario).

A pesar de que la mayoría de las pruebas con el Scraper haciendo uso del lenguaje de instrucciones fueron satisfactorias, la realización del web scraping si puede llegar a tener limitaciones importantes a considerar: La primera es que muchas de las páginas web actuales implementan estrategias para evitar la extracción automatizada y masiva de su información, por lo que no es raro encontrarse con frecuencia Captchas anti-bot que cada vez son más complejos y sofisticados. A lo largo del desarrollo de este scraper se pudo evidenciar como en una de las páginas web, la prueba anti-bot que presentaba cambió a lo largo de los meses, presentando inicialmente un cuadro de texto para ingresar letras de una imagen distorsionada, y terminando en pruebas de selección y clasificación de imágenes en varias fases. Encontrar una forma de saltar o resolver estas pruebas por medio de un programa scraper es una tarea ardua y de considerable dificultad que bien podrían ameritar todo un trabajo de estudio e investigación en torno a esto. Por el momento lo único que se puede hacer es evitar extraer información de las páginas que implementan este tipo de barrera de protección de datos.

Otra de las dificultades más comunes con las que se puede encontrar el scraper, es la implementación de bloqueos de IP's en las páginas cuando se detecta que se están realizando demasiadas peticiones de una misma máquina. Este problema es relativamente más fácil de abordar debido a que la solución generalmente se puede conseguir simplemente cambiando la dirección IP de la máquina en donde está el programa de web scraping. Para esto actualmente existen VPN's o softwares de subscripción que pueden proveer pools de direcciones IP's por medio de las cuales poder conectarse a las páginas web con bloqueos.

También existen otros problemas externos al web scraper y las medidas de seguridad de las páginas, como por ejemplo la disponibilidad de las mismas páginas

y el ancho de banda de la conexión, que puede afectar la carga correcta de la página y la renderización de los datos. Estos problemas no se pueden abordar con el web scraper debido a que dependen enteramente de la página o la fuente misma de la información.

En resumen: La aplicación web construida se basa en una arquitectura principal orientada a microservicios y está compuesta por 3 partes fundamentales: 1) El Frontend, desarrollado con el framework Angular y haciendo uso de algunas librerías de los servicios de Google Maps, por medio de las cuales los usuarios pueden elegir una propiedad específica desde un mapa e iniciar el proceso de web scraping, además de la posibilidad de configurar la forma en que se extraerá la información en cada página de información general de una propiedad, de sus impuestos, escrituras e hipotecas. 2) El Backend, desarrollado en el framework de .NET, el cual está compuesta por cuatro microservicios, tres de ellos correspondiendo a cada uno de los modelos de información planteados en el modelo genérico de datos, y un microservicio específico para las funciones de autenticación y autorización. Los tres microservicios que corresponden al modelo, hacen uso de la tercera parte fundamental de la aplicación. 3) El programa de web scraping que cuenta con una interfaz REST para recibir las peticiones desde cada uno de los microservicios del Backend, ingresa a cada una de las páginas web donde están los datos de interés de cada submodelo, extrae la información especificada en la petición y retorna los datos en bruto para ser almacenados en la base de datos de la aplicación. Este programa de web scraping, basado en el framework Selenium, hace uso de un lenguaje de instrucciones diseñado específicamente para facilitar la conversión de instrucciones del usuario al lenguaje de Selenium.

CAPÍTULO 5

5 RESULTADOS DE EVALUACIÓN DE LA APLICACIÓN WEB

Para cumplir con el tercer objetivo específico de este trabajo se hizo necesario verificar la satisfacción de la empresa interesada con el producto entregado, en nuestro caso, la evaluación del prototipo final de la aplicación web por parte de ATIX DIGITAL S.A.S., quien es la entidad interesada y quien apoyó su proceso de desarrollo y despliegue. Con sus opiniones y respuestas a diferentes preguntas planteadas se logró calificar el nivel de satisfacción general del producto y saber si el desarrollo cumplió con los objetivos propuestos para la empresa. Esto se realizó mediante la adaptación de una encuesta de satisfacción originaria del sitio web www.encuestafacil.com.

Antes de que los usuarios respondieran la encuesta, el equipo de trabajo se aseguró de que los usuarios interesados conocieran la aplicación web a través de una exposición donde se explicó una a una las diferentes funcionalidades disponibles en esta. También se permitió a los usuarios el acceso a la aplicación para que tuvieran una exposición directa a esta.

La principal funcionalidad que llamó la atención a quienes luego serían encuestados y que se proponen a probar a fondo es la obtención de información por parte de un agente inmobiliario. Ellos seleccionaron una la parcela y verificaron la veracidad y concordancia de los datos que el scraper logra obtener, verificaron el tiempo de procesamiento que toma realizar el scraping de una parcela, verificaron el diseño y la fluidez de la interfaz gráfica, y también comprobaron la facilidad con la que un agente inmobiliario busca y obtiene dicha información.

Luego de esta exposición directa a la aplicación se mostraron las funcionalidades adicionales de la aplicación, las cuales permiten la configuración del scraper a cada uno de los dominios de información que se manejan (información general, impuestos y documentos) y se incentiva a que cada uno pruebe y configure un dominio de información. Después, se explican detalles de despliegue y arquitectura de la aplicación para así lograr dar una visión general de la forma cómo se estructuró la aplicación.

Finalmente, se brinda un espacio para que los interesados realicen su retroalimentación acerca de PiExtractor. Luego, se realiza la encuesta de satisfacción al usuario.

El número de personas encuestadas vino dado de acuerdo con la capacidad de expertos en el tema que nos proporcionó la empresa ATIX DIGITAL S.A.S. En este caso fueron tres personas expertas en web scraping con conocimiento en temas inmobiliarios y aplicaciones relacionadas con este negocio.

La plantilla seleccionada de encuestafacil fue la denominada satisfacción del Cliente (Producto) y se modificó para definir un índice CSAT (Customer SATisfaction) numérico para la característica satisfacción definida en la norma ISO 9001:2015. Este rango numérico se encuentra entre 1 y 5, donde el número 5 es el mayor y el número 1 es el menor. Las preguntas fueron modificadas buscando la evaluación de las distintas funcionalidades que contiene la aplicación web desarrollada. Primero se buscó medir el grado de satisfacción del usuario con cada una de las funcionalidades de la aplicación, luego se buscó que el usuario brindara su opinión general sobre el producto y finalmente se le permitió libremente describir mejoras para la aplicación. Las preguntas y los resultados de la encuesta se muestran en la **Tabla 5**.

***Tabla 5**. Preguntas y respuestas de la encuesta aplicada*

No	**Pregunta**		**Número de personas que marcaron la opción**				
			5	**4**	**3**	**2**	**1**
1	¿Cuál es su grado de satisfacción con la interacción del mapa presentado en la aplicación?	Fluidez de la interfaz	-	3	-	-	-
		Facilidad de aprendizaje para usar esta funcionalidad	-	2	1	-	-
		Interfaz amigable al usuario	-	2	1	-	-
		La funcionalidad resulta útil para el usuario	-	2	1	-	-
2	¿Cuál es su grado de satisfacción con la obtención y la presentación de la información inmobiliaria?	Fluidez de la interfaz	-	3	-	-	
		Facilidad de aprendizaje para usar esta funcionalidad	-	2	1	-	-
		Interfaz amigable al usuario	-	2	1	-	-
		La funcionalidad resulta útil para el usuario	-	2	1	-	-
3	¿Cuál es su grado de satisfacción con el apartado de configuración del scraper?	Fluidez de la interfaz	-	3	-	-	-
		Facilidad de aprendizaje para usar esta funcionalidad	-	2	1	-	-
		Interfaz amigable al usuario	-	2	1	-	-

		La funcionalidad resulta útil para el usuario	-	2	1	-	-
4	¿Cuál es su grado de satisfacción general con Property Information extractor (PiExtractor)?		-	3	-	-	-
5	¿Recomendaría usted Piextractor a otras personas del sector inmobiliario?		2	-	1	-	-
6	¿En comparación con otras alternativas de Piextractor, piextractor es ...?		-	-	-	-	-
7	¿Utilizaría usted Piextractor de nuevo?		-	3	-	-	-
8	¿No he tenido ningún inconveniente a la hora de usar Piextrator?		-	2	1	-	-
9	Piextrator cubre mis necesidades.		-	2	1	-	-
10	Piextrator es fácil de usar.		-	3	-	-	-

Las primeras tres preguntas que muestra la **Tabla 5** buscan definir el nivel de satisfacción con las tres funcionalidades principales de la aplicación, la primera es la interacción con el mapa de la aplicación, mediante el cual un agente inmobiliario tiene que interactuar para lograr obtener la información de la parcela o propiedad. La segunda muestra el grado de satisfacción del usuario con la información que logra extraer el scraper y que se muestra en la aplicación. La última describe el grado de satisfacción del usuario cuando requiere realizar la configuración del scraper para un nuevo condado o modificar un condado que realizó un cambio en su página web. Para medir la satisfacción del usuario con cada uno de estos apartados se dividen las preguntas en diferentes características, fluidez de la interfaz, facilidad de aprendizaje para usar la funcionalidad, amigabilidad de la interfaz y utilidad de la funcionalidad para el usuario. Los resultados obtenidos para la primera pregunta (Interacción con el mapa) estuvieron entre satisfecho y no aplicable, teniendo respuestas mayoritarias en satisfecho. Los resultados obtenidos para la segunda pregunta (presentación de información) estuvieron entre satisfecho y no aplicable, donde también se tuvieron respuestas mayoritarias en satisfecho. Finalmente, Los resultados obtenidos para la tercera pregunta (configuración del scraper) estuvieron entre satisfecho e insatisfecho, donde también se tuvieron respuestas mayoritarias en satisfecho, pero se precisa mejorar en la siguiente versión cada una de las características de esta funcionalidad.

Las siguientes preguntas se enfocaron en la satisfacción general del interesado con la aplicación desarrollada. Como se muestra en esta figura, todas las respuestas están en el rango de 3 (indiferente, tal vez lo recomendaría, Igual, Talvez, entre otras) a 5 (muy satisfecho, lo recomendaría completamente, muy superior, definitivamente sí, entre otras), Las respuestas presentan mayoría en el valor 4 y en una pregunta con 5. Los usuarios no identificaron un producto competidor con las mismas o similares funcionalidades provistas por Piextractor, por esta razón no calificaron una pregunta.

Por último, se tienen los resultados de las preguntas libres al usuario para que especifique mejoras de producto, como se muestra en la **Tabla 6**.

Luego, se calculó el porcentaje de satisfacción del usuario para cada tema tratado en las diferentes preguntas mediante el índice CSAT, para esto se sumaron todos los interesados que marcaron las puntuaciones satisfecho y muy satisfecho (4 y 5) y se dividen entre el número total de personas que respondieron la encuesta. Los resultados se muestran en la **Tabla 7**.

Después de calcular el porcentaje de satisfacción por pregunta se obtuvo un resultado numérico general de la encuesta. Se toman en cuenta las 10 primeras preguntas, exceptuando la 6, la cual no fue respondida por ningún encuestado. Para esto se calculó el promedio de la puntuación CSAT; para las tres primeras preguntas se calcula el promedio y cada resultado se suma con el promedio de las demás preguntas (Ver **Tabla 8**). Finalmente, se calculó el puntaje general, el cual fue de 3.83.

Los resultados obtenidos en la encuesta contrastan en gran medida con las opiniones recibidas verbalmente por parte de los usuarios en el momento de la presentación de la aplicación web, donde se destacaron positivamente aspectos relacionados con la presentación y la usabilidad. También se pueden evidenciar que algunos de los aspectos funcionales de la aplicación aún requieren refinamiento y mejora para lograr dar mayor satisfacción a las necesidades de los potenciales usuarios de la aplicación.

Los resultados obtenidos con la encuesta permiten plantear un índice de satisfacción base para el prototipo actual, el cual puede verse refinado si se amplía la población encuestada y/o se realizan las diferentes mejoras para la siguiente versión de la aplicación web propuestas por los usuarios encuestados. Esto también abre las puertas para que nuevos proyectos den continuidad al trabajo ya desarrollado a la fecha.

Tabla 6 *Recomendaciones futuras de interesados*

No	Pregunta	Respuesta	Usuario No
11	Del estado actual de la aplicación ¿Qué cree usted que se pueda mejorar?	Mejorar el rendimiento, el scraping sirve para extraer información masiva de internet, por lo tanto que sea rápido debe ser una parte importante del sistema.	1
		Extraer información previamente en paralelo.	2
		En general el UX puede ser más llamativo para que pueda convertirse en producto.	3

12	Del estado actual de la aplicación ¿Qué nuevos aspectos cree usted que se deberían incluir en el producto?	búsqueda de barrios o zonas completas de una ciudad específica. Filtrado de esa data y análisis de resultados que le permiten al usuario obtener información valiosa.	1
		Para búsquedas más específicas.	2
		Reportes masivos de propiedades para decisiones relevantes de negocio.	3

Otra fase importante para la evaluación de la aplicación que se logró realizar es la presentación de ésta a diferentes personas que trabajan en el mercado inmobiliario de Estados Unidos, los cuales se denominan Real Estates. La empresa ATIX DIGITAL S.A.S. convocó a dos Real Estates y para ellos se logró materializar una exposición de la herramienta en funcionamiento, con esto logramos obtener una basta retroalimentación para el proyecto; se obtuvieron diferentes opiniones e ideas realizadas hacia el estado actual de la aplicación y de mejoras futuras, las cuales se podrán desarrollar en una versión posterior para tener éxito en el paso a producción de la herramienta.

***Tabla 7**. Porcentaje de satisfacción del interesado por cada pregunta.*

Satisfacción del interesado para preguntas específicas	
Numero pregunta	**Satisfacción del interesado**
1.1	100%
1.2	66.6%
1.3	66.6%
1.4	66.6%
2.1	100%
2.2	66.6%
2.3	66.6%
2.4	66.6%
3.1	66.6%
3.2	66.6%
3.3	66.6%
3.4	66.6%
4	100%
5	66.6%
6	No fue respondida
7	100%
8	66.6%
9	66.6%
10	100%

El principal problema que tienen los Real Estates es la usabilidad de las aplicaciones existentes, generalmente cuando requieren información de parcelas, les envían archivos de Excel con los respectivos datos, ellos no tienen la oportunidad de interactuar con el programa cuando deseen y no pueden seleccionar diferente tipo de configuraciones que si puede tener con la aplicación desarrollada en el presente trabajo. Por lo anterior, esto se convierte en un proceso bastante costoso en tiempo, debido a que ellos tienen que comunicarse con los encargados de ventas y esperar a que les envíen la información. Existen pocas herramientas con una interfaz amigable para ellos, las herramientas que utilizan y poseen interfaz, no llegan a tener funcionalidades amigables a la hora de obtener información de las parcelas. Una de las mejores observaciones realizadas al proyecto fue la sencillez al momento de usarla y que incluso se podrían atender mercados distintos a los representados por los agentes inmobiliarios, como, por ejemplo, abrir una versión gratuita de la aplicación al público en general con el ánimo de ayudarles a encontrar oportunidades de negocios inmobiliarios.

***Tabla 8**. Promedio de satisfacción de los interesados por pregunta.*

Promedio resultante por pregunta	
Número de pregunta	**Puntuación promedio**
1	3.75
2	3.75
3	3.5
4	4
5	4.3
7	4
8	3.6
9	3.6
10	4

Otro problema que tienen los agentes inmobiliarios está relacionado con la actualización de los datos que tienen las plataformas que ellos usan. Ellos han comprado distinta información poco actualizada, información que puede o no llegar a ser útil. Eso ya se convierte en un problema que se resuelve mucho más fácilmente con la aplicación desarrollada. Por esto recalcaron que la funcionalidad desarrollada en la aplicación es muy útil a la hora de obtener la información actualizada que se requiere.

También surgieron nuevas ideas para la aplicación, ideas que para los Real Estates serían fundamentales a la hora de hacer la compra de una licencia o de pagar una suscripción a la aplicación web desarrollada, entre las propuestas están: la generación de reportes, haciendo referencia a realizar scraping masivo de diferentes parcelas en diferentes estados y obtener reportes de acuerdo con diferentes parámetros; la nacionalidad de los propietarios, las parcelas con deudas de impuestos o con hipotecas y parcelas en un rango de precios.

Uno de los usuarios interesados propuso un método alterno para conseguir los datos de la página web, el cual consistía en consumir directamente los servicios REST utilizados por las páginas, que se ven en forma de peticiones HTTP lanzadas por el navegador web y que pueden ser vistas en el apartado de Network en las opciones de desarrollador. De esta forma, la obtención de la información se podría hacer de manera más rápida, visto que no dependería de que un navegador renderice y visualice toda una página web para después extraer cada dato deseado dentro de ésta. Sin embargo, la factibilidad de esta solución depende de que las peticiones HTTP de los servicios de la página no estén encriptados o protegidos con tokens de autenticación. Esta propuesta resulta de gran valor puesto que puede añadirse como una nueva funcionalidad de la siguiente versión de la aplicación web, buscando construir así lo que podría ser una herramienta híbrida (servicios REST + Web scrapping), en la que se tenga la posibilidad de extraer la información por medio de servicios REST si es posible, o realizar la extracción de la información haciendo uso de las funcionalidades de web scraping que actualmente ya se tienen desarrolladas.

Con la información que se obtuvo de los usuarios de ATIX DIGITAL S.A.S. y de los Real Estates, se definieron diferentes funcionalidades adicionales que mejorarían el valor agregado que la herramienta ofrece, entre las funcionalidades más relevantes y que se lograron implementar en el prototipo luego de aplicar la encuesta, se destaca la posibilidad de cargar la información que se tiene de la base de datos y aplicar a esta diferentes filtros de búsqueda de información existente como precios, propietarios y tipos de viviendas.

En resumen: Se elaboró una encuesta útil para medir el nivel de satisfacción de los interesados con el producto. Se encuestaron diferentes expertos en temas inmobiliarios y de web scraping con gestión de la empresa ATIX DIGITAL S.A.S. Los resultados obtenidos permitieron cuantificar la característica de satisfacción definida en la norma ISO 9001:2015, mediante la implementación de respuestas a preguntas relacionadas con el índice CSAT. Además, se realizaron mejoras al producto presentado en la evaluación para obtener el producto final de este trabajo de grado, mejoras desde el punto de vista técnico y de mercado propuestas por los usuarios encuestados. Adicionalmente, se adquirió una visión futura del producto, dado que se brindaron diversas ideas por parte de dos Real Estates de Estados Unidos, expertos en bienes inmobiliarios.

CAPÍTULO 6

6 CONCLUSIONES Y TRABAJO FUTURO

Se propuso un modelo genérico de información que representa la mayor cantidad de atributos comunes (información general, impuestos, escrituras e hipotecas) sobre bienes raíces disponible en los sitios web de Real Estate de varias ciudades de Estados Unidos (Fort Lauderdale, Miami Dade, Jacksonville, Tampa), el cual fue la base fundamental para desarrollar el producto software planteado y brindar información útil para que los agentes (corredores) de bienes raíces puedan realizar análisis de esta información y establecer estrategias de negocio como buscar propiedades con problemas en pago de impuestos o hipotecas que pueda significar un precio de compra más bajo. Este modelo se convierte en una base sólida para el futuro desarrollo de soluciones software, como el desarrollo de APIs para la comercialización de información inmobiliaria y proyectos de análisis de datos inmobiliarios.

El modelo genérico de información es alimentado mediante una técnica de web scraping que consiste en un programa que simula el comportamiento de un usuario dentro de un navegador web para acceder a las páginas de Real Estate previamente configuradas y luego extraer el contenido relevante de estas.

Para el funcionamiento del programa de scraping, fue necesario definir un nuevo lenguaje de instrucciones que facilita la extracción de los datos requeridos que se encuentran en los elementos HTML de las páginas web. A pesar de que este lenguaje no estaba previsto como un objetivo del trabajo de grado, su definición e implementación se convirtió en el núcleo del funcionamiento de la aplicación de web scraping, ya que el motor desarrollado para interpretar este lenguaje es el que se encarga de convertir instrucciones sencillas y más entendibles para un usuario “avanzado” en las funciones o instrucciones que utiliza el software Selenium para manipular el navegador e interactuar con las páginas web. Además, aunque el lenguaje propuesto se realizó en el marco de páginas web de Real Estate, en realidad puede usarse para interactuar y extraer datos de una página web de cualquier naturaleza, ya que sus instrucciones son generales y sirven para buscar elementos y extraer datos dentro de la estructura de “virtualmente” cualquier página web.

Se desarrolló y desplegó un prototipo de aplicación web con una arquitectura basada en microservicios que permite gestionar los datos inmobiliarios del modelo

genérico de información previamente propuesto, configurar cada una de las instrucciones del lenguaje construido para estructurar la manera en que el scraper obtiene la información, y brindar a los diferentes usuarios una interfaz gráfica, mediante la cual puedan obtener y visualizar la información inmobiliaria de manera sencilla. La aplicación web desarrollada fue evaluada y aprobada por diferentes usuarios expertos en el sector inmobiliario y su funcionalidad actual la convierte en una base sólida para obtener a corto plazo un producto software novedoso en ese sector de Estados Unidos.

Una de las mejoras sugeridas por los usuarios que evaluaron la fase final de la aplicación web fueron inmediatamente incorporadas en la versión final desarrollada en el marco del presente trabajo de grado, además se plantean diversas actualizaciones como trabajo futuro, como lo son: la generación de reportes especializados (listar las propiedades que presentan problemas en sus pagos de impuestos o hipotecas), el scraping masivo por zonas específicas, la posibilidad de comercializar información inmobiliaria por localidades y el análisis en los valores del mercado inmobiliario en distintas zonas de cada ciudad.

A pesar de que el uso de técnicas de web scraping no ha sido tan conocidas ni implementadas de manera profunda, estas técnicas denotan un gran potencial como una alternativa viable para recolección de datos de manera masiva y automatizada. Sin embargo, aún existen muchas dificultades por superar, que van desde problemas mayormente técnicos (bloqueos de direcciones IP, anti-bots, entre otros), hasta limitantes en el ámbito legal.

Uno de los mayores obstáculos técnicos para realizar web scraping hoy en día, es que cada vez más páginas y portales web refuerzan sus aplicaciones para evitar que programas automatizados extraigan masivamente su información o evitar sobrecargas en sus servidores, razón por la cual, el uso de los Captchas se ha vuelto tan popular y actualmente la seguridad en estos se ha incrementado significativamente, lo que obliga a los robots a no utilizar únicamente técnicas de navegación y extracción, sino también a usar algoritmos de reconocimiento de imágenes o algoritmos de simulación de comportamiento humano durante la interacción humano-computador. Este item en sí mismo amerita todo un foco de estudio aparte debido a su complejidad.

Otra de las formas de evitar este tipo de “fugas” de información de los portales web, es el bloqueo de la dirección IP de la máquina de la que se sospecha comportamiento no humano. La solución más sencilla a este problema es el cambio dinámico de la dirección IP, por lo que no representa un problema tan complejo como el tratamiento de un Captcha. Todos estos retos pueden ser objeto para abordar en un trabajo futuro centrado en fortalecer los puntos débiles que pueda presentar un programa de scraping, ya que es un tema transversal a cualquier área o sector de aplicación de estas técnicas.

Cabe resaltar que el perfeccionamiento del lenguaje de instrucciones creado para esta solución es una de las bases fundamentales para el desarrollo futuro de este y de diferentes proyectos software que incluyan web scraping. Aunque el lenguaje se diseñó e implementó para responder a las necesidades específicas de este proyecto, puede ser usado en otras áreas de aplicación y modificado en aspectos como la fiabilidad, rendimiento y la sintaxis misma del lenguaje propuesto, volviéndolo aún más fácil de usar e incluyéndole más características que permitan mayores posibilidades de interacción con cualquier página.

La retroalimentación recibida de la aplicación facilitó identificar un potencial desarrollo futuro, el cual consiste de una herramienta híbrida que permita la extracción de datos de páginas web haciendo uso de distintas técnicas de extracción, dando prioridad al llamado de los servicios o APIs que dispongan los sitios web para consultar los datos, y en caso de no proporcionar esta opción, usar la extracción por medio de web scraping.

Por último, este proyecto demostró ir más allá, apuntando a convertirse en un producto software con potencial de mercado internacional que puede cumplir y alinearse a las demandas de los futuros avances tecnológicos y a posibles actualizaciones del producto. Cabe resaltar que para el caso específico de Colombia, una aplicación de web scraping inmobiliario como la desarrollada en este proyecto, no se podría explotar en su máximo potencial actualmente, debido a la escasez o ausencia de sitios web municipales y departamentales que ofrezcan información abierta al público de bienes inmuebles de una manera tan completa y accesible como en EE. UU.

CAPÍTULO 7

7 BIBLIOGRAFÍA

[1] "United States Home Prices & Home Values | Zillow," *Zillow*. 2017, Accessed: Aug. 28, 2020. [Online]. Available: https://www.zillow.com/home-values/.

[2] "Agencias inmobiliarias en los Estados Unidos." https://www.expat.com/es/empresas/america-del-norte/ee-uu/2_inmobiliario/agencias-inmobiliarias/ (accessed Aug. 28, 2020).

[3] F. Blazheski, "Perspectivas del Sector Inmobiliario para 2017," *17-01-17*, p. 1, 2017, Accessed: Sep. 02, 2020. [Online]. Available: https://www.fincaraiz.com.co/perspectivas-del-sector-inmobiliario-para-2017-noticia-718.aspx.

[4] A. Daher, "El sector inmobiliario y las crisis económicas," *Eure*, vol. 39, no. 118, pp. 47–76, Sep. 2013, doi: 10.4067/S0250-71612013000300003.

[5] R. Diouf, E. N. Sarr, O. Sall, B. Birregah, M. Bousso, and S. N. Mbaye, "Web Scraping: State-of-the-Art and Areas of Application," in *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, Dec. 2019, pp. 6040–6042, doi: 10.1109/BigData47090.2019.9005594.

[6] U.S. Census Bureau, "U.S. Census Bureau QuickFacts: UNITED STATES," *Quick Facts*, 2020. https://www.census.gov/quickfacts/fact/table/US/PST045219# (accessed Jun. 22, 2020).

[7] P. Information, A. Information, and B. Information, "Property Search Application - Miami-Dade County Search : Property Search Application - Miami-Dade County," pp. 1–3, 2016, Accessed: Sep. 02, 2020. [Online]. Available: https://www.miamidade.gov/Apps/PA/propertysearch/#/.

[8] "DCAD Property Map." https://maps.dcad.org/prd/dpm/ (accessed Sep. 02, 2020).

[9] "¿Qué es un CAPTCHA? - Ayuda de Administrador de G Suite." https://support.google.com/a/answer/1217728?hl=es (accessed Aug. 28, 2020).

[10] Scrapy developers, "Scrapy | A Fast and Powerful Scraping and Web Crawling Framework," 2008. https://scrapy.org/ (accessed Jun. 14, 2020).

[11] G. Kiradoo, "Software Engineering Quality to Enhance the Customer Satisfaction Level of the Organization." Jun. 30, 2019, Accessed: Aug. 27, 2021. [Online]. Available: https://papers.ssrn.com/abstract=3539958.

[12] "ISO 9001:2015(es), Sistemas de gestión de la calidad — Requisitos." https://www.iso.org/obp/ui/#iso:std:iso:9001:ed-5:v1:es (accessed Aug. 27, 2021).

[13] A. Schulz, J. Lassig, and M. Gaedke, “Practical Web Data Extraction: Are We There Yet?-A Short Survey,” in *Proceedings - 2016 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2016*, Jan. 2017, pp. 562–567, doi: 10.1109/WI.2016.0096.

[14] J. Wang, F. H. Lochovsky, and W. Su, “ODE: Ontology-Assisted Data Extraction,” 2009, doi: 10.1145/1538909.1538914.

[15] M. I. Varlamov and D. Y. Turdakov, “A survey of methods for the extraction of information from Web resources,” *Program. Comput. Softw.*, vol. 42, no. 5, pp. 279–291, Sep. 2016, doi: 10.1134/S0361768816050078.

[16] L. Richardson, “Beautiful Soup: We called him Tortoise because he taught us.,” *Crummy*. 2021, Accessed: Jun. 14, 2020. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/.

[17] T. Cervenka, “Jaunt - Java Web Scraping & JSON Querying.” https://jaunt-api.com/ (accessed Jun. 14, 2020).

[18] “Jauntium - Java Browser Automation for Webscraping and Automated Testing.” https://jauntium.com/ (accessed Jun. 14, 2020).

[19] Jonathan Magnan, “Html Agility pack | Html Agility Pack,” 2017. https://html-agility-pack.net/ (accessed May 23, 2020).

[20] “Welcome to MechanicalSoup’s documentation! — MechanicalSoup 0.12.0 documentation.” https://mechanicalsoup.readthedocs.io/en/stable/ (accessed Jun. 14, 2020).

[21] B. Muthukadan, “Selenium with Python — Selenium Python Bindings 2 documentation,” 2018. https://selenium-python.readthedocs.io/ (accessed Oct. 10, 2021).

[22] “Tap Into Web Content at Scale | Crawled Web Data | Webhose.” https://webhose.io/ (accessed Oct. 10, 2021).

[23] “scrapestack - Free Proxy & Web Scraping API.” https://scrapestack.com/ (accessed Jun. 14, 2020).

[24] Import.io, “Extracción de datos, datos web, recolección web, preparación de datos, integración de datos,” 2019. https://www.import.io/ (accessed May 24, 2020).

[25] “Zyte (formerly Scrapinghub) #1 Web Scraping Service.” https://www.zyte.com/ (accessed Oct. 10, 2021).

[26] “ParseHub Pricing | Free plan and paid subscriptions.” https://www.parsehub.com/pricing (accessed May 24, 2020).

[27] “Agenty® - Agents for Machine Intelligence.” https://www.agenty.com/ (accessed May 24, 2020).

[28] “Web Scraper - Free Web Scraping - Chrome Web Store.” https://chrome.google.com/webstore/detail/web-scraper-free-web-scra/jnhgnonknehpejjnehehllkliplmbmhn (accessed Aug. 19, 2020).

[29] “DataScraper – Consigue esta extensión para ? Firefox (es).” https://addons.mozilla.org/es/firefox/addon/datascraper/ (accessed Jun. 14, 2020).

[30] G. Boeing and P. Waddell, “New Insights into Rental Housing Markets across the United States: Web Scraping and Analyzing Craigslist Rental Listings,” *J. Plan. Educ. Res.*, vol. 37, no. 4, pp. 457–476, Dec. 2017, doi: 10.1177/0739456X16664789.

[31] P. Kumkar, I. Madan, A. Kale, O. Khanvilkar, and A. Khan, “Comparison of Ensemble Methods for Real Estate Appraisal,” in *Proceedings of the 3rd International Conference on Inventive Computation Technologies, ICICT 2018*, Nov. 2018, pp. 297–300, doi: 10.1109/ICICT43934.2018.9034449.

[32] H. Li, Y. D. Wei, Y. Wu, and G. Tian, “Analyzing housing prices in Shanghai with open data: Amenity, accessibility and urban structure,” *Cities*, vol. 91, pp. 165–179, Aug. 2019, doi: 10.1016/j.cities.2018.11.016.

[33] S. H. Hong, S. K. Lee, and J. H. Yu, “Automated management of green building material information using web crawling and ontology,” *Autom. Constr.*, vol. 102, pp. 230–244, Jun. 2019, doi: 10.1016/j.autcon.2019.01.015.

[34] H. Ahmed, T. Ahmed, W. Haider, S. Noman, M. Asad, and A. Masroor, “Producing Standard Rules for Smart Real Estate Property Buying Decisions based on Web Scraping Technology and Machine Learning Techniques,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 3, 2020, doi: 10.14569/IJACSA.2020.0110363.

[35] R. S. M. R. A. Apoorva Pasbola, “Housing Property Recommendation with Automated Requirement Prediction | International Journal of Advanced Science and Technology,” May 2020. Accessed: Jun. 14, 2020. [Online]. Available: http://sersc.org/journals/index.php/IJAST/article/view/13681.

[36] J. C. Correa *et al.*, “Evaluation of collaborative consumption of food delivery services through web mining techniques,” *J. Retail. Consum. Serv.*, vol. 46, pp. 45–50, 2019, doi: 10.1016/j.jretconser.2018.05.002.

[37] J. Zhou, C. Cheng, L. Kang, and R. Sun, “Integration and Analysis of Agricultural Market Information Based on Web Mining,” 2018, vol. 51, no. 17, pp. 778–783, doi: 10.1016/j.ifacol.2018.08.101.

[38] P. Pillai and D. Amin, “ScienceDirect ScienceDirect Understanding the requirements Of the Indian IT industry using web scrapping,” *Procedia Comput. Sci.*, vol. 172, pp. 308–313, 2020, doi: 10.1016/j.procs.2020.05.050.

[39] A. K. Sharma, V. Shrivastava, and H. Singh, “Experimental performance analysis of web crawlers using single and Multi-Threaded web crawling and indexing algorithm for the application of smart web contents,” in *Materials Today: Proceedings*, 2020, vol. 37, no. Part 2, pp. 1403–1408, doi: 10.1016/j.matpr.2020.06.596.

[40] S. Yang, S. Wi, J. H. Park, H. M. Cho, and S. Kim, “Framework for developing a building material property database using web crawling to improve the applicability of energy simulation tools,” *Renew. Sustain. Energy Rev.*, vol. 121, p. 109665, 2020, doi: 10.1016/j.rser.2019.109665.

[41] H. C. Lo, K. G. Koedijk, X. Gao, and Y. T. Hsu, “How do job vacancy rates predict firm performance? A web crawling massive data perspective,” *Pacific Basin Financ. J.*, vol. 62, p. 101371, Sep. 2020, doi: 10.1016/j.pacfin.2020.101371.

[42] J. Zhang, T. Zou, and Y. Lai, “Novel method for industrial sewage outfall detection: Water pollution monitoring based on web crawler and remote sensing interpretation techniques,” *J. Clean. Prod.*, vol. 312, p. 127640, 2021, doi: 10.1016/j.jclepro.2021.127640.

[43] V. Stallone, M. Wetzels, and M. Klaas, “Applications of Blockchain Technology in marketing systematic review of marketing technology companies,”

Blockchain Res. Appl., p. 100023, Jul. 2021, doi: 10.1016/j.bcra.2021.100023.
[44] C. Muehlethaler and R. Albert, “Collecting data on textiles from the internet using web crawling and web scraping tools,” *Forensic Sci. Int.*, vol. 322, 2021, doi: 10.1016/j.forsciint.2021.110753.
[45] U. Baskaran and K. Ramanujam, “Automated scraping of structured data records from health discussion forums using semantic analysis,” *Informatics Med. Unlocked*, vol. 10, pp. 149–158, 2018, doi: 10.1016/j.imu.2018.01.003.
[46] B. S. Santos, I. Silva, M. da C. Ribeiro-Dantas, G. Alves, P. T. Endo, and L. Lima, “COVID-19: A scholarly production dataset report for research analysis,” *Data Br.*, vol. 32, p. 106178, Oct. 2020, doi: 10.1016/j.dib.2020.106178.
[47] R. Azibeiro Melchor *et al.*, “Abstracts CT-152 Application of Web-Scraping Techniques for Autonomous Massive Retrieval of Hematologic Patients’ Information During SARS-CoV2 Pandemic Ageing of Long-term Allogeneic Hematopoietic Cell Recipients Compared to Their Donors,” 2020. Accessed: Oct. 11, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2152265020307795.
[48] J. Schedlbauer, G. Raptis, and B. Ludwig, “Medical informatics labor market analysis using web crawling, web scraping, and text mining,” *Int. J. Med. Inform.*, vol. 150, p. 104453, Jun. 2021, doi: 10.1016/J.IJMEDINF.2021.104453.
[49] V. Catalani *et al.*, “Psychonauts’ psychedelics: A systematic, multilingual, web-crawling exercise,” *Eur. Neuropsychopharmacol.*, vol. 49, pp. 69–92, Aug. 2021, doi: 10.1016/J.EURONEURO.2021.03.006.
[50] “Best Real-Estate Markets.” https://wallethub.com/edu/best-real-estate-markets/14889 (accessed Nov. 19, 2021).
[51] “Best Real Estate Investing Housing Markets in the U.S. | Investor Junkie.” https://investorjunkie.com/real-estate/best-us-housing-markets/ (accessed Oct. 12, 2021).
[52] “2020’s Hottest Housing Markets Will Come From the South, While California Sees a Cooldown - Jan 2, 2020.” http://zillow.mediaroom.com/2020-01-02-2020s-Hottest-Housing-Markets-Will-Come-From-the-South-While-California-Sees-a-Cooldown (accessed Oct. 12, 2021).
[53] “Zillow 2020 Urban-Suburban Market Report - Zillow Research.” https://www.zillow.com/research/2020-urb-suburb-market-report-27712/ (accessed Oct. 12, 2021).
[54] Zillow Inc., “Zillow: Real Estate, Apartments, Mortgages & Home Values.” pp. 1–18, 2019, Accessed: Oct. 10, 2021. [Online]. Available: https://www.zillow.com/.
[55] “The Best Investment Reviews, Promotions and Education | Investor Junkie.” https://investorjunkie.com/ (accessed Oct. 10, 2021).
[56] “WalletHub: Free Credit Scores, Reports & Credit Improvement.” https://wallethub.com/ (accessed Oct. 10, 2021).
[57] “OpenStreetMap Wiki.” https://wiki.openstreetmap.org/wiki/Main_Page (accessed Oct. 11, 2021).
[58] “OpenAddresses.” https://openaddresses.io/ (accessed Oct. 11, 2021).
[59] “Decompose by subdomain.” https://microservices.io/patterns/decomposition/decompose-by-

subdomain.html (accessed Oct. 11, 2021).
[60] “Papertrail - cloud-hosted log management, live in seconds.” https://www.papertrail.com/ (accessed Oct. 11, 2021).