



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

“Plataforma para la estimación de precios de mercado de departamentos en la Ciudad de México”

2024-A053

Presenta
Humberto Alejandro Ortega Alcocer

Director
M. en C. Ariel López Rojas



Junio 2024



No de TT: 2024-A053

13 de Junio 2024

Documento Técnico

**“Plataforma para la estimación de precios de mercado
de departamentos en la Ciudad de México”**

Presenta
Humberto Alejandro Ortega Alcocer¹

Director
M. en C. Ariel López Rojas

RESUMEN

En este reporte de trabajo terminal se describe el desarrollo de una aplicación web que permite a los usuarios estimar los precios de mercado de departamentos en la Ciudad de México. La plataforma utiliza modelos de regresión avanzados para generar estimaciones precisas basadas en datos en tiempo real. Estos modelos fueron desarrollados empleando técnicas de machine learning y análisis de datos geoespaciales.

Palabras clave: Estimación de precios, aplicación web, inteligencia artificial, modelos de regresión, análisis geoespacial, bienes raíces.

¹hortegaa1500@alumno.ipn.mx

CARTA RESPONSIVA

Que otorga visto bueno y avala la conclusión de documentación del Trabajo Terminal bajo los lineamientos establecidos por la Comisión Académica de Trabajos Terminales (CATT)

CDMX, a 20 de Junio de 2024

**M. EN C. ANDRÉS ORTIGOZA CAMPOS
PRESIDENTE DE LA COMISIÓN ACADÉMICA DE TRABAJOS TERMINALES
P R E S E N T E**

EN ATENCIÓN A:

M. EN A. N. MARÍA MAGDALENA SALDÍVAR ALMOREJO
SECRETARIA EJECUTIVA

Por medio de la presente, se informa que el Trabajo Terminal Núm. | 2024-A053

Que lleva por Título: Plataforma para la estimación de precios de mercado

De departamentos en la Ciudad de México

Fue concluido satisfactoriamente por:

Humberto Alejandro Ortega Alcocer

Se evalúa que la documentación entregada mediante discos en formato DVD fue **revisada de manera precisa y exhaustiva** con el propósito de asegurar que los avances desarrollados bajo la supervisión de quien o quienes suscriben, hayan cumplido con lo planteado en el protocolo original, así como en lo establecido por el Documento Rector de Operación y Evaluación para los Trabajos Terminales de la ESCOM.

ATENTAMENTE

"LA TÉCNICA AL SERVICIO DE LA PATRIA"

Mtro. Ariel López Rojas
Director

Advertencia

- || “Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan.”
 - || La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.
 - || Información adicional sobre este reporte técnico podrá obtenerse en:
 - || La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000, extensión 52000.

Agradecimientos

En primer lugar, agradezco al Instituto Politécnico Nacional (IPN) y a la Escuela Superior de Cómputo (ESCOM) por brindarme la oportunidad y los recursos necesarios para formarme como profesional y llevar a cabo este proyecto.

Quiero expresar mi más profundo agradecimiento a mi director de trabajo terminal, Ariel López Rojas, por su invaluable apoyo, orientación y paciencia a lo largo de este proyecto. Su experiencia y dedicación han sido fundamentales para la culminación de este trabajo.

A mi familia, les agradezco desde el fondo de mi corazón por su amor incondicional y por estar siempre a mi lado, brindándome la fuerza y el ánimo necesarios en cada etapa de mi vida académica. En particular, quiero mencionar con especial cariño a mi madre, Lourdes, quien ha sido mi principal fuente de inspiración y motivación. A mis abuelos, Lourdes y Alejandro, cuyo sabio consejo y apoyo constante me han guiado siempre. A mi hermano, Gerardo, por su comprensión y respaldo incondicional.

A Maura, cuya comprensión y afecto han sido fundamentales para llegar hasta aquí. Su amor y apoyo han sido una fuente constante de fortaleza.

A mis amigos, gracias por su inquebrantable apoyo moral y por creer en mí incluso en los momentos más difíciles. Su compañía y aliento han sido esenciales para llegar hasta aquí.

Este logro es también de ustedes. Gracias a todos.

Humberto Alejandro Ortega Alcocer

Índice.

Índice	3
1. Introducción	12
1.1. Estado del Arte	13
1.2. Objetivos	15
1.2.1. Objetivo general	15
1.2.2. Objetivos particulares	15
1.3. Justificación	15
1.4. Metodología	17
1.4.1. Implementación de la metodología	18
1.4.2. Iteraciones (Sprints)	20
1.5. Organización del documento	20
2. Marco teórico	23
2.1. Valuación Inmobiliaria	23
2.1.1. Importancia de la valuación inmobiliaria	24
2.1.2. Valuación inmobiliaria en México	24
2.1.3. Métodos de valuación inmobiliaria tradicionales	25
2.1.4. Retos y limitantes de la valuación inmobiliaria tradicional en México	26
2.1.5. Inteligencia Artificial en la valuación inmobiliaria	27
2.2. Inteligencia Artificial	27
2.2.1. Aprendizaje Automático	28
2.2.2. Aprendizaje Supervisado	28
2.2.3. Librerías de Aprendizaje Automático en Python	31
2.3. Conjunto de Datos	33
2.3.1. Conjuntos de Datos ya existentes	34
2.3.2. Web Scraping	36
2.3.3. Archivos CSV	38
2.4. Servicios Web	39

2.4.1.	Python	40
2.4.2.	API	45
2.4.3.	JSON	45
2.4.4.	SSL	46
2.4.5.	HTTPS	47
2.4.6.	DNS	47
2.4.7.	CDN	47
2.4.8.	Docker	48
2.5.	Aplicación Web	48
2.5.1.	JavaScript	49
2.5.2.	HTML	49
2.5.3.	CSS	50
2.5.4.	React	52
2.5.5.	Vue	54
2.6.	Plataforma de nube	54
2.6.1.	Google Cloud Platform	55
2.6.2.	Amazon Web Services	57
2.7.	GIS	59
2.7.1.	QGIS	60
2.7.2.	kepler.gl	61
2.7.3.	Google Maps	62
3.	Análisis	64
3.1.	Problemática	64
3.2.	Propuesta de Solución	64
3.2.1.	¿Por qué se eligió emplear Inteligencia Artificial?	65
3.2.2.	¿Por qué se eligió desarrollar una plataforma web?	65
3.3.	Herramientas a Emplear	66
3.3.1.	Herramientas de Software	66
3.3.2.	Herramientas de Hardware	72
3.4.	Análisis Exploratorio de Datos	73
3.4.1.	Análisis Exploratorio Geoespacial	74
3.4.2.	Análisis Exploratorio Econométrico	83
3.5.	Algoritmos de Aprendizaje Automático	92
3.5.1.	Máquinas de Soporte Vectorial	92
3.5.2.	Random Forest	92
3.5.3.	Redes Neuronales Artificiales	92
3.6.	Estudio de Factibilidad	93
3.6.1.	Factibilidad Técnica	93
3.6.2.	Factibilidad Económica	93
3.6.3.	Factibilidad Operativa	94

3.6.4. Factibilidad General	95
3.6.5. Consideraciones Legales	95
3.7. Análisis de Riesgos	95
3.7.1. Riesgo de Cronograma	97
3.7.2. Riesgo de Costos	98
3.7.3. Riesgo Tecnológico	99
3.7.4. Riesgo Operacional	100
3.7.5. Riesgos Externos	100
3.7.6. Riesgos Legales	101
3.7.7. Riesgos generalizados	102
3.7.8. Exposición al Riesgo	103
4. Diseño	104
4.1. Arquitectura del Sistema	104
4.1.1. Descripción de la arquitectura	105
4.1.2. Caso de Uso	106
4.1.3. Reglas de Negocio	107
4.1.4. Historias de Usuario	108
4.1.5. Diagrama de Secuencia	109
4.1.6. Requerimientos Funcionales	110
4.1.7. Requerimientos No Funcionales	110
4.2. Conjunto de Datos	111
4.2.1. Descripción del Conjunto de Datos	111
4.2.2. Obtención del Conjunto de Datos	112
4.2.3. Preprocesamiento del Conjunto de Datos	114
4.3. Limpieza del Conjunto de Datos	114
4.3.1. Detección de Outliers	115
4.3.2. Detección de Valores Faltantes	115
4.3.3. Detección de Valores Atípicos	115
4.3.4. Detección de Valores Negativos	115
4.3.5. Detección de Valores Categóricos	115
4.3.6. Detección de Valores Numéricos	115
4.4. Modelo de Estimación de Precios	116
4.4.1. Arquitectura del Modelo	116
4.4.2. Entrenamiento del Modelo	117
4.4.3. Evaluación del Modelo	117
4.4.4. Despliegue del Modelo	117
4.5. Aplicación Web	118
4.5.1. Wireframes	118
4.5.2. Dependencias externas	119

5. Implementación	121
5.1. Arquitectura del sistema	121
5.2. Desarrollo de los modelos	122
5.2.1. Conjunto de datos	122
5.2.2. Preprocesamiento de los datos	123
5.2.3. Entrenamiento de los modelos	123
5.3. Servicio web	128
5.3.1. Importación de los modelos	128
5.3.2. Evaluación de los modelos	129
5.3.3. API REST	131
5.3.4. Documentación	139
5.4. Interfaz gráfica	141
5.4.1. Estructura del proyecto	141
5.4.2. Navegación y traducciones	143
5.4.3. Estimación de precio	144
5.4.4. Acerca de	150
5.4.5. Mapa interactivo	155
5.4.6. Diseño responsivo	159
5.5. Despliegue en la nube	161
5.5.1. Dominio	162
5.5.2. Empaque y distribución del servicio web	163
5.5.3. Despliegue del servicio web	166
5.5.4. Despliegue de la interfaz gráfica	170
5.5.5. Servicios externos	174
5.6. Pruebas del sistema	177
5.6.1. Google Page Speed Insights	177
5.6.2. Google Analytics	180
5.7. Costos	183
5.7.1. AWS	183
5.7.2. Google Maps	184
5.7.3. ExchangeRate-API	185
6. Conclusiones	187
7. Trabajo a futuro	189
Anexos	192
.1. Código fuente para preprocesamiento de datos y entrenamiento de modelos	193
.2. Código fuente de la API	199

Índice de figuras.

1.1. Ejemplo de un tablero de Trello	19
1.2. Ejemplo de un tablero de Trello con la metodología SCRUM	19
2.1. Diagrama de Aprendizaje Automático	29
2.2. Neurona	30
2.3. Red Neuronal	31
2.4. Logo de Scikit-Learn	32
2.5. Logo de TensorFlow	33
2.6. Logo de Keras	33
2.7. Conjunto de Datos del Catastro de la Ciudad de México	35
2.8. Arquitectura de Selenium	37
2.9. Arquitectura de un Servicio Web	40
2.10. Logo de Python	41
2.11. Logo de Flask	42
2.12. Logo de Django	43
2.13. Logo de FastAPI	44
2.14. Arquitectura de una API REST	46
2.15. Arquitectura de Docker	49
2.16. Logo de JavaScript	50
2.17. Logo de HTML	50
2.18. Logo de CSS	51
2.19. Ejemplo de sitio web desarrollado con Bootstrap	52
2.20. Ejemplo de sitio web desarrollado con Material-UI	53
2.21. Logo de React	53
2.22. Logo de Vue	54
2.23. Distribución del mercado de proveedores de servicios de nube en 2023	55
2.24. Servicios de Google Cloud Platform	56
2.25. Servicios de Amazon Web Services	58
2.26. Aplicaciones de los GIS en el análisis de datos espaciales	60
2.27. Interfaz de usuario de QGIS	61

2.28. Interfaz de usuario de kepler.gl	62
2.29. Interfaz de usuario de Google Maps	63
3.1. Captura de pantalla de QGIS con los datos de bienes raíces de la Ciudad de México.	71
3.2. Visualización de puntos de anuncios.	75
3.3. Mapa de de densidad de anuncios por alcaldía.	76
3.4. Mapa de calor de precios.	78
3.5. Mapa de calor de número de recámaras.	79
3.6. Mapa de calor de número de baños.	80
3.7. Mapa de calor de metros cuadrados.	81
3.8. Mapa de calor de antigüedad.	82
3.9. Mapa de calor de número de estacionamientos.	83
3.10. Visualización de los componentes principales.	91
3.11. Matriz de riesgos inicial.	96
3.12. Matriz de riesgo de cronograma.	97
3.13. Matriz de riesgo de costos.	98
3.14. Matriz de riesgo tecnológico.	99
3.15. Matriz de riesgo operacional.	100
3.16. Matriz de riesgo externo.	101
3.17. Matriz de riesgo legal.	102
3.18. Matriz de riesgos generalizados.	103
3.19. Exposición al riesgo.	103
4.1. Arquitectura del sistema	105
4.2. Diagrama de caso de uso	106
4.3. Diagrama de secuencia	110
4.4. Diagrama de flujo del <i>web scraper</i>	112
4.5. Diagrama de flujo del <i>Convertidor</i>	114
4.6. Arquitectura del modelo	116
4.7. Wireframe con el estado inicial de la aplicación web	118
4.8. Wireframe con la estimación de precio del departamento	119
4.9. Funcionalidad de geo-localización de Google Maps	120
5.1. Arquitectura final del sistema.	121
5.2. Vistazo al conjunto de datos utilizado para el entrenamiento de los modelos.	122
5.3. Gráfica de evaluación de los modelos de aprendizaje automático.	131
5.4. Documentación de la API para la ruta principal.	134
5.5. Documentación de la API para la ruta de predicción.	135
5.6. Documentación de la API para la ruta de gráficas de evaluación.	137

5.7. Documentación de la API para la ruta de información de modelos.	139
5.8. Documentación de la API en formato OpenAPI (Swagger).	140
5.9. Documentación de la API en formato ReDoc.	141
5.10. Estructura del proyecto de la interfaz gráfica.	142
5.11. Interfaz gráfica en francés.	143
5.12. Sección de estimación de precio de la interfaz gráfica.	145
5.13. Formulario con datos de una propiedad y mapa resultante de la geocodificación.	148
5.14. Sección de resultados de la predicción de la interfaz gráfica.	149
5.15. Opciones de conversión de moneda en la interfaz gráfica.	150
5.16. Sección de información general de la página de Acerca de.	151
5.17. Sección de información sobre los datos de la página de Acerca de.	151
5.18. Sección de información sobre el mapa interactivo de la página de Acerca de.	152
5.19. Sección de información sobre los modelos de aprendizaje automático de la página de Acerca de.	153
5.20. Sección de gráficas de la página de Acerca de.	154
5.21. Sección de documentación de la página de Acerca de.	155
5.22. Página de Mapa interactivo con alerta de carga de datos.	156
5.23. Página de Mapa interactivo con capa de datos de propiedades en la Ciudad de México.	157
5.24. Mapa interactivo con capa de datos de propiedades, alcaldías y contaminación acústica.	159
5.25. Pantalla principal en diseño móvil	160
5.26. Pantalla de resultados en diseño móvil	160
5.27. Pantalla de Acerca de en diseño móvil	161
5.28. Pantalla de Mapa interactivo en diseño móvil	161
5.29. Información del dominio <code>skyprice.xyz</code> en Namecheap.	162
5.30. Imagen de Docker publicada en Docker Hub [1].	166
5.31. Certificado SSL en AWS Certificate Manager.	167
5.32. Registros DNS para el dominio de la aplicación.	167
5.33. Entorno de Elastic Beanstalk en ejecución.	169
5.34. Verificación SSL para la URL <code>https://api.skyprice.xyz</code>	170
5.35. Compilación de la interfaz gráfica con NextJS.	171
5.36. Bucket de AWS S3 con la interfaz gráfica.	172
5.37. Distribución de contenido de AWS CloudFront.	173
5.38. Verificación SSL para la URL <code>https://skyprice.xyz</code>	174
5.39. Configuración de la API de Google Maps en Google Cloud.	175
5.40. Página principal de la API de ExchangeRate-API.	176

5.41. Resultados de las pruebas de Lighthouse en un entorno de escritorio.	178
5.42. Resultados de las pruebas de Lighthouse en un entorno móvil.	179
5.43. Página principal de Google Analytics con información sobre el tráfico del sitio web.	180
5.44. Información sobre los usuarios que visitan el sitio web.	182
5.45. Información sobre la tecnología utilizada por los usuarios para acceder al sitio web.	183
5.46. Costos diarios de la aplicación en la nube.	184
5.47. Costos de la API de Google Maps para la geocodificación de direcciones.	185
5.48. Costos de la API de ExchangeRate-API para la conversión de monedas.	186

Índice de cuadros.

1.1. Resumen de productos similares	14
2.1. Métodos de Valuación Inmobiliaria Tradicionales en México . .	26
2.2. Comparativa de algoritmos de aprendizaje supervisado [2]. .	29
3.1. Datos correspondientes al número de anuncios encontrados por alcaldía.	77
3.2. Estadísticas Descriptivas de los Listados de Bienes Raíces . .	85
3.3. Matriz de Correlación entre Variables	86
3.4. Resultados de la Regresión Simple	88
3.5. Resultados de la Regresión Múltiple	88
3.6. Resultados del Análisis de Componentes Principales	90
3.7. Factibilidad Técnica	93
3.8. Factibilidad Económica	94
3.9. Costos del Proyecto (6 meses) en Moneda Nacional	94
3.10. Factibilidad Operativa	95
3.11. Riesgo de Cronograma	97
3.12. Riesgo de Costos	98
3.13. Riesgo Tecnológico	99
3.14. Riesgo Operacional	100
3.15. Riesgo Externo	101
3.16. Riesgo Legal	102
4.1. Descripción del caso de uso	107
4.2. Historia de usuario	109
4.3. Datos utilizados para el desarrollo del proyecto	111
5.1. Capas de datos disponibles en el mapa interactivo.	158
5.2. Registros DNS configurados en AWS Route 53 para el dominio <code>skyprice.xyz</code>	163

Capítulo 1

Introducción

La industria inmobiliaria en la actualidad enfrenta grandes desafíos en la valuación de bienes inmuebles, donde la precisión y la eficiencia son cruciales para el éxito de las transacciones [3]. Los profesionales del sector y los inversionistas a menudo encuentran dificultades para determinar los precios más apropiados de cada inmueble debido a la falta de herramientas eficientes y la complejidad del mercado inmobiliario [4]. La propuesta de este trabajo terminal implica el uso de nuevas tecnologías que le ofrezcan al usuario la posibilidad de obtener un precio de mercado.

La inteligencia artificial ha demostrado un gran potencial en el análisis y la toma de decisiones en diversos campos de alto impacto en nuestra sociedad como lo son la medicina, la construcción, las finanzas, el comercio electrónico y el sector inmobiliario [3]. Algunas investigaciones han propuesto el uso de técnicas de inteligencia artificial, como el aprendizaje automático y las redes neuronales, para mejorar la cobertura geográfica en la estimación de valores de propiedades [5, 6]. La valuación de bienes inmuebles requiere la consideración de múltiples factores, como la ubicación, las características de la propiedad, las tendencias del mercado y las condiciones económicas [6]. Estos factores hacen que la valuación de propiedades sea una tarea compleja y desafiante, que podría beneficiarse del uso de técnicas avanzadas de inteligencia artificial para analizar la información y generar estimaciones precisas.

En este contexto, se han comparado múltiples algoritmos y metodologías de aprendizaje automático, como Extra Trees (ET), k–Nearest Neighbors (KNN), y Random Forest (RF), dónde se observa que cada mercado cuenta con características específicas que requieren incluirse en el proceso de estimación y por lo tanto deben considerarse para lograr mejores resultados [7].

Sin embargo, el planteamiento fundamental puede ser adaptado fácilmente en función de la cobertura geográfica deseada.

La propuesta del presente Trabajo Terminal se enfoca en el desarrollo de un método de estimación de costo para departamentos que, mediante el uso de Inteligencia Artificial, sea capaz de realizar estimaciones de costo útiles para un precio de mercado en la Ciudad de México a manera de facilitar y optimizar el tiempo que toma obtener un vistazo inicial a lo que podría ser el valor del inmueble en cuestión.

Actualmente, en el mercado se encuentran algunas soluciones que hacen uso de inteligencia artificial para la valuación de propiedades, como Zillow's Zestimate [8] y HouseCanary [9]. Sin embargo, estas soluciones aún presentan limitaciones en términos de cobertura y adaptabilidad a diferentes mercados y contextos. La propuesta busca abordar estas limitaciones y ofrecer una herramienta más adaptable al mercado mexicano, más específicamente al de la Ciudad de México.

1.1. Estado del Arte

A continuación, se presenta el Cuadro 1.1 con las características de aplicaciones web similares y comparándolos con la propuesta. Dentro de las características se incluye el tipo de aplicación, las características de estimación de costo, la cobertura geográfica y el precio en el mercado.

Cuadro 1.1: Resumen de productos similares comparados con la propuesta

Software	Características	Precio en el mercado
Zillow's Zestimate [8]	Aplicación web que ofrece una estimación de costo de bienes inmobiliarios, hace uso de inteligencia artificial para obtener los valores de cada característica, los datos usados son los propios de Zillow (marketplace de bienes raíces) permite ingresar detalles específicos a cada tipo de inmueble y ofrece una valuación muy precisa y detallada, sin embargo sólo tiene cobertura en el mercado de EE.UU. y Canadá.	Gratis
House Canary [9]	Aplicación web que ofrece una estimación de costo de bienes inmobiliarios, hace uso de inteligencia artificial para obtener las estimaciones de cada característica, permite ingresar detalles específicos a cada inmueble, sólo tiene cobertura en EE.UU. y Canadá.	Considerando el producto <i>Agile Evaluation</i> , \$ 0.80 USD por valuación
Estimador de costo de Inmuebles24 [10]	Aplicación web que ofrece una estimación aproximada de costo para bienes raíces, hace uso de inteligencia artificial para obtener la estimación de costo general, los datos utilizados son de su propio sitio web (marketplace de bienes raíces), su cobertura es baja y únicamente entrega el precio de mercado.	Gratis
Propuesta de herramienta en el presente trabajo terminal	Aplicación web que ofrecerá una estimación aproximada de costo para departamentos en la Ciudad de México, hace uso de inteligencia artificial para obtener la estimación de costo general, la cobertura será determinada por la disponibilidad del conjunto de datos a emplear.	Gratis

1.2. Objetivos

1.2.1. Objetivo general

Crear una herramienta de apoyo para profesionales del sector inmobiliario, inversionistas o cualquier persona interesada en la estimación de costo de algún departamento en la Ciudad de México para que puedan obtener un precio de mercado.

Mediante el uso de un conjunto de datos (dataset) y servicios de la nube (Amazon Web Services) los datos se procesarán a fin de crear un módulo que pueda identificar patrones a través de redes neuronales para ofrecer un estimado de precio a partir de un conjunto de características que clasifiquen al inmueble. A su vez, se desarrollará una interfaz web intuitiva a modo de prototipo con el fin de que el usuario pueda visualizar el estimado de precio del inmueble en cuestión.

Dicha herramienta pretende ayudar a los usuarios descritos con el fin de que puedan obtener una estimación de costo que les permita tener un aproximado de precio de mercado.

1.2.2. Objetivos particulares

- Generar un conjunto de datos (dataset) con precios históricos de bienes inmobiliarios en la Ciudad de México para efecto de entrenamiento del modelo de aprendizaje automático.
- Hacer uso de alguna herramienta que utilice aprendizaje automático (machine learning) e implementar su uso en la nube para ayudar a procesar los anuncios de un tipo de inmueble particular.
- Desarrollar una interfaz web intuitiva a modo de prototipo que utilice una aplicación web alojada en infraestructura de nube para la captura de los datos del usuario así como la entrega de la estimación de precio calculada por el modelo.

1.3. Justificación

La generación de estimaciones precisas y eficientes para la valuación de bienes inmuebles puede ser un desafío abrumador para profesionales y aficionados, debido a la falta de herramientas adecuadas y la complejidad del mercado inmobiliario. En ocasiones, se pueden recurrir a servicios especializados en la

valuación de propiedades, pero estos pueden resultar costosos y no siempre cumplir con las expectativas del usuario. Por tanto, este trabajo terminal busca desarrollar una herramienta que facilite el proceso de estimación de costos para departamentos en la Ciudad de México, beneficiando a profesionales del sector inmobiliario, inversionistas e interesados en general en el sector inmobiliario.

Por lo general, la valuación de bienes inmuebles implica la consideración de múltiples factores, como la ubicación, las características de la propiedad y las tendencias del mercado [6]. Estos patrones pueden ser identificados mediante el uso de inteligencia artificial y el análisis de datos históricos y actuales del mercado inmobiliario. Al enfocar el análisis en el mercado de la Ciudad de México, se espera que el modelo propuesto sea más relevante y preciso para este contexto específico.

El trabajo terminal propone utilizar técnicas avanzadas de inteligencia artificial, como el aprendizaje automático y las redes neuronales, para mejorar la cobertura en la estimación de costos de departamentos en la Ciudad de México [5]. Además, se plantea implementar la solución en un entorno en la nube, lo que facilitaría la adaptación y flexibilidad de la plataforma, permitiendo su actualización y mantenimiento independientemente de la implementación.

A diferencia de las soluciones existentes en el mercado, como Zillow's Zestimate y HouseCanary, este proyecto busca desarrollar un método de estimación de costo específicamente adaptado al mercado inmobiliario de la Ciudad de México, utilizando tecnologías y enfoques diferentes para lograr resultados más precisos y relevantes [9, 8].

Se utilizarán conocimientos adquiridos durante la carrera para llevar a cabo un análisis eficaz y detallado de la problemática y las opciones de solución. Para ello, se retomarán técnicas de gestión y diseño de proyectos vistas en *Administración de Proyectos, Gestión Empresarial e Ingeniería de Software*, conocimientos de programación como en *Programación Orientada a Objetos, Sistemas Distribuidos y Desarrollo Web*, técnicas de *Inteligencia Artificial y Análisis de Datos* y, finalmente, se hará uso de herramientas de comunicación y redacción científica vistas en *Comunicación Oral y Escrita* para un análisis detallado del ámbito de la valuación inmobiliaria permitiendo un detallado y eficaz análisis del problema y las soluciones disponibles.

1.4. Metodología

Para el desarrollo de este trabajo terminal se optó por la metodología ágil SCRUM, debido a que, como proceso de gestión de proyectos, se reduce complejidad y permite actuar rápidamente ante cualquier cambio para poder satisfacer las necesidades de los clientes. Una de las cualidades más importantes de SCRUM es que nos ofrece una clara visibilidad sobre el avance del proyecto así como de los pasos intermedios a realizar en cada sprint para alcanzar los objetivos. Además, permite trabajar de manera eficiente colaborativamente, es decir, en equipo, para obtener el mejor resultado posible.

Ken Schwaber y Jeff Sutherland [11] explican Scrum de una manera clara y simple. Dicen que no es una colección de partes y/o componentes definidos de manera prescriptiva, sino que está basado en un modelo de proceso empírico, basado en la autoorganización de los equipos los cuales logran lidiar con lo imprevisible, resolviendo los problemas complejos inspeccionándolos y adaptándose continuamente.

SCRUM contiene los siguientes eventos:

- **Sprint Planning:** Reunión de planificación del sprint, en la que el equipo de desarrollo selecciona las tareas que se van a realizar durante el sprint.
- **Daily Scrum:** Reunión diaria de seguimiento, en la que el equipo de desarrollo se sincroniza sobre el estado de las tareas.
- **Sprint Review:** Reunión de revisión del sprint, en la que el equipo de desarrollo muestra los resultados del sprint.
- **Sprint Retrospective:** Reunión de retrospectiva del sprint, en la que el equipo de desarrollo analiza el sprint y propone mejoras.

Estos eventos existen con el fin de establecer una regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Estos eventos son bloques de tiempo (*time boxes*), de tal forma que todos cuentan con una duración máxima.

También se definen los siguientes artefactos:

- **Product Backlog:** Lista de requisitos del producto, ordenados por prioridad.
- **Sprint Backlog:** Lista de requisitos del producto seleccionados para el sprint actual.

- **Incremento:** Conjunto de requisitos del producto completados durante el sprint actual.

Los artefactos en SCRUM se definen para así fomentar la transparencia de la información de tal manera que todos los involucrados tengan el mismo entendimiento de que es lo que se está llevando a cabo, además de que facilita oportunidades para realizar inspecciones y adaptaciones.

De igual manera, se pueden crear sprints, los cuales son ciclos breves de un mes o menos con diferentes fases, en las cuales al final de cada ciclo se define la fecha para la entrega de una versión del producto deseado. Debido a que se trata de una versión, no se indica la finalización del proyecto, en otro caso que habrá un mantenimiento constante para que se obtenga un producto final óptimo.

En virtud de que con el uso de SCRUM se logra la integración de todas las partes involucradas en el proyecto, la administración y participación es sencilla y fácil de manejar para todas las etapas. Asimismo, se cuenta con un registro de las labores realizadas y se le da un seguimiento. De igual forma nos proporciona una respuesta rápida a los cambios, así como la implementación de pruebas funcionales durante el proceso.

1.4.1. Implementación de la metodología

Para la implementación de la metodología SCRUM se utilizará la herramienta *Trello* [12], la cual es una aplicación web que permite organizar proyectos en tableros. En cada tablero se pueden crear listas, las cuales pueden contener tarjetas, las cuales a su vez pueden contener listas de chequeo, etiquetas, fechas de vencimiento, archivos adjuntos, entre otras características. En la Figura 1.1 se muestra un ejemplo de un tablero de *Trello*.

Para implementar la metodología SCRUM en *Trello*, se creará un tablero para el proyecto, en el cual se crearán las siguientes listas:

- **Product Backlog:** Lista de requisitos del producto, ordenados por prioridad.
- **Sprint Backlog:** Lista de requisitos del producto seleccionados para el sprint actual.
- **Sprint:** Lista de tareas que se realizarán durante el sprint actual.
- **Done:** Lista de tareas que se han completado.

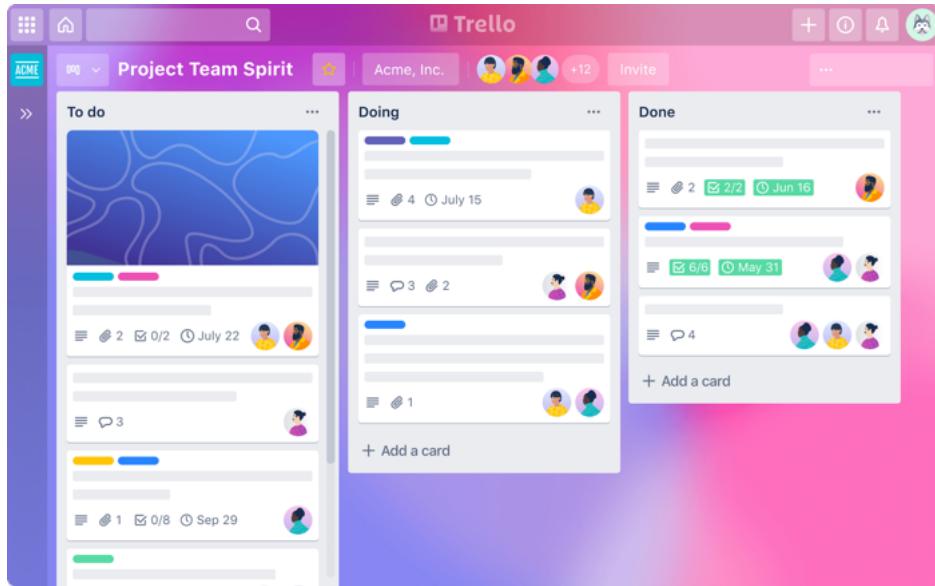


Figura 1.1: Ejemplo de un tablero de Trello [12]

En la Figura 1.2 se muestra un ejemplo de un tablero de *Trello* con las listas descritas anteriormente.

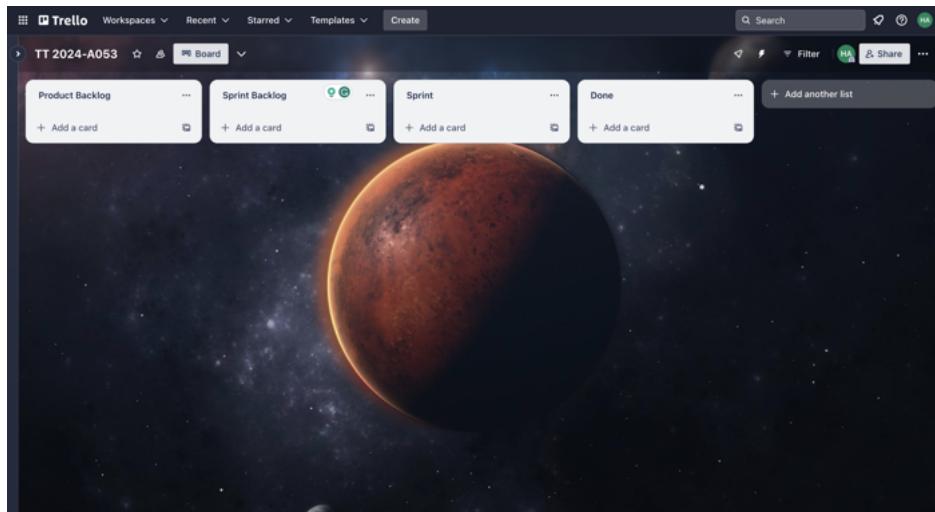


Figura 1.2: Ejemplo de un tablero de Trello con la metodología SCRUM

La intención será que cada tarjeta en la lista *Product Backlog* represente una tarea, la cual se moverá a la lista *Sprint Backlog* cuando se decida que se realizará en el sprint actual. Posteriormente, cada tarea se dividirá en subtareas, las cuales se agregarán a la lista *Sprint*. Finalmente, cuando se

complete una subtarea, se moverá a la lista *Done*.

1.4.2. Iteraciones (Sprints)

Para la implementación de la metodología SCRUM se realizarán iteraciones (o sprints) de una duración de 2 semanas, las cuales se llevarán a cabo a lo largo del desarrollo del presente trabajo terminal.

Trabajo Terminal I

- Construir una base de datos de anuncios de bienes inmuebles en la Ciudad de México, la cual contendrá las características fundamentales de cada anuncio, así como su precio de venta.
- Realizar una limpieza a los datos extraídos del dataset.
- Realizar un análisis exploratorio de datos para identificar patrones relevantes en los datos.
- Seleccionar los algoritmos de aprendizaje automático que mejor se adapten a los datos.
- Seleccionar las herramientas necesarias para trabajar en la construcción de un modelo utilizando redes neuronales.

Trabajo Terminal II

- Generar un modelo en la nube para su entrenamiento utilizando las redes neuronales previamente elegidas.
- Implementar la interfaz que el usuario final verá.
- Integrar la interfaz con la aplicación que utiliza el modelo utilizando un servidor web.
- Realizar las pruebas pertinentes, así como la reingeniería requerida.
- Desarrollar el manual técnico y el de usuario.

1.5. Organización del documento

A manera de que el lector pueda tener un entendimiento más claro de la estructura de este documento, se presenta a continuación una breve descripción de cada uno de los capítulos que lo conforman.

Capítulo 2. Marco teórico

La sección de *Marco teórico* se divide en dos partes, la primera parte se enfoca en la descripción del contexto en el que se desarrolla el trabajo terminal, es decir, se habla de la industria inmobiliaria y la valuación de bienes inmuebles, así como la importancia de la inteligencia artificial en este sector. La segunda parte se enfoca en la descripción de los conceptos teóricos que se utilizarán a lo largo del trabajo terminal, es decir, se habla de las redes neuronales, el aprendizaje automático y las herramientas que se utilizarán para el desarrollo del trabajo terminal.

Capítulo 3. Análisis

En el tercer capítulo, se describe el análisis realizado para la elaboración de este trabajo terminal.

Para esto, se divide en tres secciones, la primera sección se enfoca en la descripción del problema, es decir, se habla de la problemática que se busca resolver con este trabajo terminal, así como los objetivos que se pretenden alcanzar y las delimitaciones que se tienen para este trabajo terminal.

La segunda sección se enfoca en la descripción de la propuesta, es decir, se habla de la solución que se propone para resolver la problemática planteada, así como los beneficios que se obtendrán al implementar esta solución.

La tercera sección se enfoca en la descripción de la factibilidad, es decir, se habla de los recursos necesarios para la elaboración de este trabajo terminal, así como las herramientas que se utilizarán para el desarrollo del mismo.

Capítulo 4. Diseño

En el cuarto capítulo, se describe el diseño realizado para la elaboración de este trabajo terminal.

Para esto, se divide en dos secciones, la primera sección se enfoca en la descripción del diseño de la base de datos, es decir, se habla de la estructura de la base de datos que se utilizará para el desarrollo de este trabajo terminal.

La segunda sección se enfoca en la descripción del diseño de la aplicación web, es decir, se habla de la estructura de la aplicación web que se utilizará para el desarrollo de este trabajo terminal.

Capítulo 5. Implementación

En el quinto capítulo, se describe la implementación realizada para la elaboración de este trabajo terminal. Se divide en: desarrollo de los modelos, desarrollo del servicio web, desarrollo de la interfaz gráfica, despliegue en la nube, pruebas del sistema y costos de operación.

Capítulo 6. Conclusiones

En el sexto capítulo, se presentan las conclusiones obtenidas a lo largo de la realización de este trabajo terminal, incluyendo los resultados obtenidos y los retos técnicos enfrentados.

Capítulo 7. Trabajo a futuro

En el séptimo capítulo, se presentan las propuestas de trabajo a futuro para continuar con el desarrollo de este trabajo terminal.

Capítulo 2

Marco teórico

2.1. Valuación Inmobiliaria

La valuación inmobiliaria es el proceso mediante el cual se determina el valor de un bien inmueble, es decir, el precio estimado en que podría ser vendido o comprado. Para esto, se toman en cuenta las características físicas, económicas, sociales y legales del inmueble, así como las condiciones del mercado inmobiliario en el que se encuentra.

Un bien inmueble se define como: "... todos los intereses, beneficios, derechos y gravámenes inherentes a la propiedad de bienes raíces físicos, donde los bienes raíces son la tierra junto con todas las mejoras que están permanentemente adheridas a ella y todos los accesorios asociados a la misma" [3]. La Consejería Jurídica del Gobierno de la Ciudad de México, por su parte, define a los inmuebles como: "... aquella cosa que no puede trasladarse de un lugar a otro, por ejemplo, un terreno o edificio." [13].

A pesar de que existen múltiples criterios de clasificación de los bienes inmuebles, en la práctica se utilizan dos criterios principales: el uso y la naturaleza del inmueble. El criterio de uso se refiere a la función que cumple el inmueble, mientras que el criterio de naturaleza se refiere a las características físicas del inmueble. Así, para fines de este trabajo terminal, se considerarán los bienes inmobiliarios de tipo de uso *multifamiliar* y de tipo de naturaleza *desarrollos inmobiliarios verticales*, mejor conocidos como *departamentos*.

2.1.1. Importancia de la valuación inmobiliaria

El proceso de valuación inmobiliaria es requerido para tener una medida cuantitativa de los beneficios y riesgos de poseer un bien inmueble. Es decir, permite determinar el valor de un bien inmueble, el cual es un factor determinante en la toma de decisiones de inversión, financiamiento, compra, venta, arrendamiento, entre otras.

Algunos de los jugadores clave en el proceso de valuación inmobiliaria son [3]:

- **Agentes inmobiliarios:** Son los encargados de la promoción y venta de bienes inmuebles.
- **Valuadores:** Son personas especializadas en la valuación de bienes inmuebles, generalmente realizando inspecciones físicas y análisis de mercado.
- **Asesores financieros:** Son aquellos individuos especializados en el análisis de la situación financiera de una persona o empresa, con el fin de recomendar las mejores opciones de inversión, financiamiento, compra, venta, arrendamiento, entre otras.
- **Instituciones financieras:** Son aquellas instituciones que ofrecen servicios y productos financieros, como créditos hipotecarios, préstamos, entre otros.
- **Desarrolladores inmobiliarios:** Se refiere a las personas u organizaciones que se dedican a la construcción de bienes inmuebles.
- **Inversionistas:** Son aquellas personas o grupos empresariales que realizan inversiones en bienes inmuebles o desarrollos inmobiliarios con el fin de obtener un beneficio económico.
- **Analistas de mercado:** Son personas especializadas en el análisis de las condiciones del mercado inmobiliario, con el fin de determinar las tendencias y proyecciones del mismo.
- **Gobierno:** Se refiere a las instituciones gubernamentales y mecanismos de gobierno que regulan el mercado inmobiliario.

2.1.2. Valuación inmobiliaria en México

En México la valuación de bienes nacionales están regulados por la Secretaría de Economía mediante la norma oficial mexicana NMX-C-459-SCFI-ONNCCE-2007 [14], los cuales son generalmente administrados por el Ins-

tituto de Administración y Avalúos de Bienes Nacionales (INDAABIN). A pesar de que en su primer implementación en 2007 no se tenía un equivalente internacional, recientemente esta norma fue compatibilizada con la provista por el International Valuation Standards Council (IVSC). En esta norma se establecen los requisitos mínimos que deben cumplir los valuadores para realizar una valuación inmobiliaria, así como los requisitos mínimos que debe cumplir un reporte de valuación inmobiliaria. La misma Secretaría de Economía, a través de la Sociedad Hipotecaria Nacional (SHN), establece las "Reglas de carácter general que establecen la metodología para la valuación de inmuebles" [15].

Por su parte, la Comisión Nacional Bancaria y de Valores (CNBV) establece los criterios para la valuación de inmuebles que sirven como garantía de créditos otorgados por las instituciones de crédito [16].

En el caso de la Ciudad de México, la Secretaría de Desarrollo Económico (SEDECO) ofrece dos certificaciones en materia de valuación inmobiliaria [17]:

- **Corredor Inmobiliario:** es la persona física que se dedica a los temas legales y financieros de la compraventa de bienes inmuebles.
- **Administrador Inmobiliario:** es la persona que se especializa en los aspectos fiscales, tributarios y administrativos de los bienes inmuebles.

2.1.3. Métodos de valuación inmobiliaria tradicionales

Existen múltiples métodos de valuación inmobiliaria, los cuales se pueden clasificar en tres categorías principales: métodos basados en el mercado, métodos basados en el ingreso y métodos basados en el costo. En la práctica, los métodos basados en el mercado son los más utilizados, ya que son los que requieren de menor información y son más fáciles de implementar. Sin embargo, los métodos basados en el ingreso y en el costo son utilizados cuando no se cuenta con información suficiente del mercado inmobiliario [18].

Existen múltiples regulaciones y criterios en relación a la valuación inmobiliaria, y dependerá del objetivo de la misma la regulación a acatar y el método disponible a utilizar. En el cuadro 2.1 se muestran los métodos de valuación inmobiliaria tradicionales más utilizados en México [18].

Cuadro 2.1: Métodos de Valuación Inmobiliaria Tradicionales en México

Método	Descripción	Aplicación
Enfoque de Comparación de Ventas	Compara la propiedad con otras similares que se hayan vendido recientemente en la misma zona. Basado en el supuesto de que propiedades con características similares deberían tener valores similares.	Ampliamente utilizado en México para todo tipo de propiedades.
Enfoque de Costos	Estima el coste de construcción de un inmueble similar a partir de cero y deduce la depreciación. Basado en el supuesto de que el valor de una propiedad viene determinado por el coste de su sustitución.	Utilizado para propiedades nuevas o recientemente renovadas.
Enfoque Basado en los Ingresos	Estima los ingresos que probablemente generará el inmueble a lo largo de su vida útil. Basado en el supuesto de que el valor de una propiedad comercial viene determinado por su potencial de generación de ingresos.	Utilizado para valorar inmuebles comerciales.

2.1.4. Retos y limitantes de la valuación inmobiliaria tradicional en México

A pesar de que los métodos de valuación inmobiliaria tradicionales son sostenidos por múltiples actores e instituciones del sector inmobiliario, el Banco Interamericano de Desarrollo (BID) ha identificado múltiples retos y limitantes en su implementación desde una perspectiva catastral [19]. Entre los retos y limitantes se encuentran:

- **Falta de información:** La información catastral es limitada y no se encuentra disponible en formatos digitales, lo que dificulta su procesamiento y análisis.
- **Falta de estandarización:** La información catastral no se encuentra

estandarizada, lo que dificulta su procesamiento y análisis.

- **Falta de transparencia:** La información catastral no se encuentra disponible para el público en general, lo que dificulta su procesamiento y análisis.
- **Falta de actualización:** La información catastral no se encuentra actualizada, lo que dificulta su procesamiento y análisis.
- **Falta de integración:** La información catastral no se encuentra integrada con otras fuentes de información, lo que dificulta su procesamiento y análisis.

Además, para determinar si uno los métodos de valuación tradicionales ofrece resultados confiables, en el año 1999 se llevó a cabo un estudio en el que se compararon múltiples mecanismos de valuación y se analizaron los resultados obtenidos. La conclusión es que la valuación posee inherentemente un alto grado de subjetividad y, a pesar de que existen mecanismos como Density Estimation and Profit Simulation (DEPS) que buscan minimizar y compensar el error, su implementación es sumamente compleja ya que requiere de una gran cantidad de consideraciones y supuestos que deben ser tomados en cuenta [20].

2.1.5. Inteligencia Artificial en la valuación inmobiliaria

A fin de poder ofrecer una valuación inmobiliaria que sea confiable, se requiere del uso de estrategias que permitan minimizar el error a la vez que se consideran e integran múltiples fuentes de información. En este sentido, la Inteligencia Artificial (IA) ha demostrado ser más flexible para resolver problemas de predicción de valores de mercado que otros enfoques metodológicos [19].

En particular, los modelos de valuación automática (Modelo de Valuación Automática (AVM), por sus siglas en inglés) basados en IA han demostrado ser más precisos que los métodos tradicionales de valuación inmobiliaria [19]. Dentro de los AVM se destecan los métodos basados en Redes Neuronales Artificiales (RNA), precios hedónicos, análisis geoespacial, lógica difusa y Modelo Autoregresivo de Media Móvil (ARIMA) [3].

2.2. Inteligencia Artificial

La inteligencia artificial (IA) se define como la ciencia e ingeniería de crear máquinas inteligentes, particularmente programas de computadora inteligen-

tes. Aunque está relacionada con la tarea de usar computadoras para comprender la inteligencia humana, la IA no se limita a métodos que son biológicamente observables. La inteligencia, en este contexto, se refiere a la parte computacional de la capacidad para lograr objetivos en el mundo, la cual varía en personas, muchos animales y algunas máquinas. Hasta ahora, no existe una definición sólida y universal de inteligencia que sea independiente de la inteligencia humana, principalmente porque aún no se caracterizan en general los procedimientos computacionales que se quieren calificar como inteligentes. La IA implica mecanismos que han sido parcialmente descubiertos por la investigación, permitiendo a las computadoras realizar algunas tareas con gran eficacia, mientras que otras aún no están al alcance. La IA no siempre busca simular la inteligencia humana; en muchas ocasiones, se centra más en resolver problemas presentados por el mundo que en imitar a las personas o animales. Esto permite a los investigadores de IA utilizar métodos que no se observan en humanos o que requieren más capacidad de cómputo de la que los humanos pueden realizar. La IA actual se caracteriza por tener mucha velocidad y memoria, pero sus habilidades se limitan a los mecanismos intelectuales que los diseñadores de programas comprenden lo suficiente como para implementar en programas. La inteligencia artificial es un campo en constante evolución, buscando alcanzar niveles de inteligencia comparables a los humanos, aunque esto aún parece requerir ideas fundamentales nuevas y una comprensión más profunda de los mecanismos de la inteligencia [21].

2.2.1. Aprendizaje Automático

El aprendizaje se refiere a un amplio espectro de situaciones en las cuales el aprendiz incrementa su conocimiento o sus habilidades para cumplir una tarea. El aprendizaje aplica inferencias a determinada información para construir una representación apropiada de algú aspecto relevante de la realidad o de algún proceso [22]. En la Figura 2.1 se muestra un diagrama general de aprendizaje automático y sus componentes.

2.2.2. Aprendizaje Supervisado

El aprendizaje supervisado implica aprender un mapeo entre un conjunto de variables de entrada \mathbf{X} y una variable de salida Y , y aplicar este mapeo para predecir las salidas de datos no vistos. El aprendizaje supervisado es la metodología más importante en el aprendizaje automático y también tiene una importancia central en el procesamiento de datos inmobiliarios. En este capítulo, nos centramos en enfoques basados en kernels para el aprendizaje supervisado. Revisamos las máquinas de vectores de soporte, que represen-

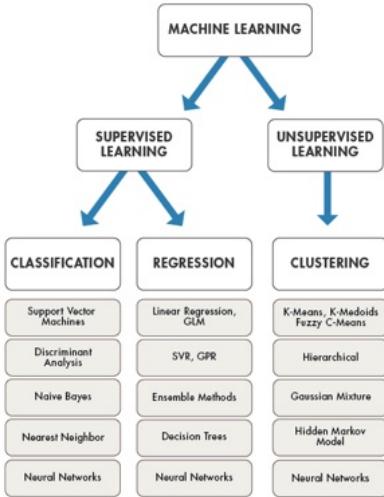


Figura 2.1: Diagrama general de algoritmos de Aprendizaje Automático [23].

tan la tecnología de aprendizaje supervisado dominante en la actualidad, particularmente en el procesamiento de datos multimedia. También revisamos los clasificadores de vecinos más cercanos que pueden considerarse, en términos generales, una estrategia basada en kernel. Las técnicas de vecinos más cercanos son populares en valuaciones porque el énfasis en la similitud es apropiado para múltiples contextos [24].

En el Cuadro 2.2 se muestra una comparativa de los algoritmos de aprendizaje supervisado más utilizados en la actualidad, así como sus ventajas, desventajas y librerías disponibles en Python [2].

Nombre	Descripción	Ventajas	Desventajas	Librería Python
Regresión Logística	Clasificación lineal.	Interpretarable, simple.	Limitada en complejidad.	sklearn
SVMs	Alto rendimiento en espacios dimensionales.	Bueno para muchas dimensiones.	Selección de kernel crucial.	sklearn
Decision Trees	Estructura en forma de árbol.	Fácil interpretación.	Riesgo de sobreajuste.	sklearn
Random Forests	Conjunto de árboles.	Reduce sobreajuste.	Menos interpretable.	sklearn
Naive Bayes	Basado en teorema de Bayes.	Eficiente y sencillo.	Suposiciones de independencia.	sklearn
Neural Networks	Modelos tipo cerebro.	Muy flexibles.	Tiempo de entrenamiento.	tensorflow, keras
Memory-Based	Aprendizaje por instancias.	No necesita modelo.	Ineficiente en grandes datos.	sklearn
Bagged Trees	Árboles ensacados.	Menor varianza.	Menor interpretabilidad.	sklearn
Boosted Trees	Árboles potenciados.	Alta eficacia en clasificación.	Riesgo de sobreajuste.	xgboost
Boosted Stumps	Tocones potenciados.	Bueno para datos dimensionales.	Menos potentes que árboles.	xgboost

Cuadro 2.2: Comparativa de algoritmos de aprendizaje supervisado [2].

Máquinas de Soporte Vectorial

Método efectivo en espacios de alta dimensión. Utiliza hiperplanos para separar clases de datos. Los hiperplanos se eligen maximizando el margen entre las clases. Las SVMs pueden usar diferentes tipos de kernels, como lineal, polinomial y radial, para adaptarse a la naturaleza no lineal de algunos datos [2].

Bosques Aleatorios

Método de ensamblaje que utiliza múltiples árboles de decisión para mejorar la robustez y precisión. Cada árbol se entrena con una muestra aleatoria del conjunto de datos. La decisión final se toma por votación mayoritaria o promediando las salidas de todos los árboles. Efectivo para reducir el sobreajuste en comparación con un solo árbol de decisión [2].

Redes Neuronales Artificiales

Las Redes Neuronales Artificiales (RNA) son sistemas adaptativos inspirados en cómo funciona el cerebro humano, con la capacidad de modificar su estructura interna para alcanzar un objetivo específico. Son especialmente útiles para resolver problemas no lineales, logrando descifrar las reglas difusas que rigen las soluciones óptimas para estos. Compuestas por nodos o elementos de procesamiento (PE), cada uno con su entrada y salida, permiten la intercomunicación entre ellos y con el entorno [25]. En la Figura 2.2 se muestra una neurona biológica, la cual es la inspiración de la unidad básica de procesamiento de una RNA.

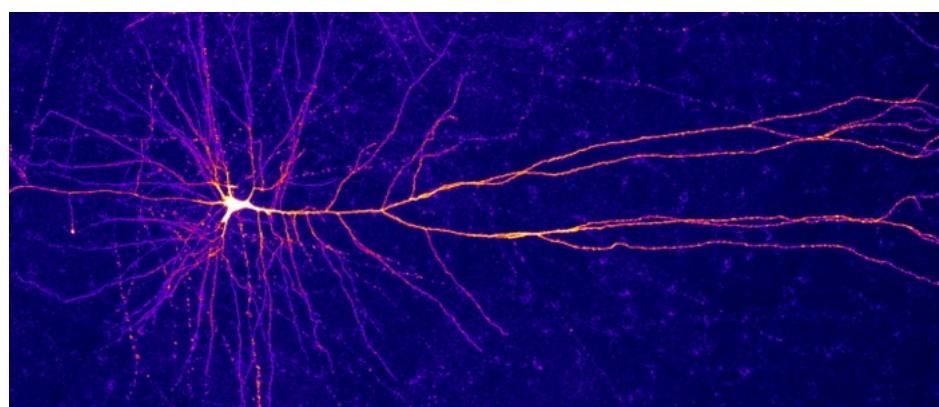


Figura 2.2: Neurona [26]

Una red neuronal artificial está compuesta por múltiples capas de nodos, cada

una de las cuales está conectada con la siguiente. La primera capa se conoce como capa de entrada, la última como capa de salida y las capas intermedias como capas ocultas. Cada nodo de una capa está conectado con todos los nodos de la capa siguiente, pero no con los de la misma capa. Cada conexión entre nodos tiene un peso asociado, el cual se utiliza para determinar la salida de un nodo [27]. La figura 2.3 muestra un ejemplo de una red neuronal artificial.

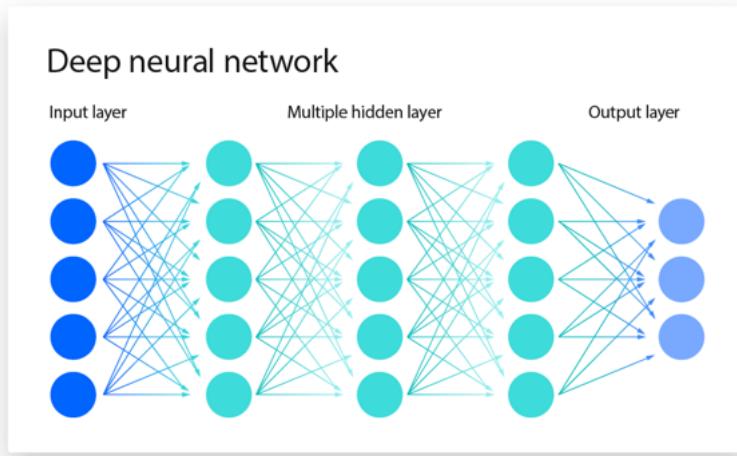


Figura 2.3: Red Neuronal [27]

2.2.3. Librerías de Aprendizaje Automático en Python

Para realizar la implementación de alguno de los algoritmos de aprendizaje mencionados anteriormente, se requiere de librerías que implementen dichos algoritmos para facilitar su uso. Esto resulta en un ahorro de tiempo y esfuerzo para el desarrollador, ya que no es necesario implementar los algoritmos desde cero y se tiene certeza sobre la calidad de los mismos. En el caso de Python, existen múltiples librerías que implementan los algoritmos de aprendizaje automático más utilizados en la actualidad. En esta sección se describen las librerías con mayor popularidad y que son utilizadas en el presente trabajo terminal.

Scikit-Learn

Scikit-learn es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas

supervisados y no supervisados de escala media. Este paquete se centra en llevar el aprendizaje automático a los no especialistas utilizando un lenguaje de alto nivel de propósito general. Se pone énfasis en la facilidad de uso, el rendimiento, la documentación y la consistencia de la API. Tiene dependencias mínimas y se distribuye bajo la licencia BSD simplificada, fomentando su uso tanto en entornos académicos como comerciales [28].



Figura 2.4: Logo de Scikit-Learn [29]

TensorFlow

TensorFlow es un sistema de aprendizaje automático que opera a gran escala y en entornos heterogéneos. Su modelo computacional se basa en grafos de flujo de datos con estado mutable. Los nodos del grafo pueden mapearse a diferentes máquinas en un clúster, y dentro de cada máquina a CPUs, GPUs y otros dispositivos. TensorFlow admite una variedad de aplicaciones, pero se enfoca particularmente en el entrenamiento y la inferencia con redes neuronales profundas. Sirve como plataforma tanto para la investigación como para la implementación de sistemas de aprendizaje automático en muchas áreas, tales como reconocimiento de voz, visión por computadora, robótica, recuperación de información y procesamiento de lenguaje natural. En esta charla, describimos TensorFlow y esbozamos algunas de sus aplicaciones. También discutimos la cuestión de qué relación puede tener TensorFlow y el aprendizaje profundo con la programación funcional. Aunque TensorFlow no es puramente funcional, muchos de sus usos están relacionados con la optimización de funciones (durante el entrenamiento), y luego con la aplicación de esas funciones (durante la inferencia). Estas funciones se definen como composiciones de primitivas simples (como es común en la programación funcional), con representaciones internas de datos que se aprenden en lugar de diseñarse manualmente. TensorFlow es un trabajo conjunto con muchas otras personas del equipo de Google Brain y otros lugares [30].

Keras

Keras es una API de aprendizaje profundo escrita en Python, que se ejecuta sobre TensorFlow, una plataforma de aprendizaje automático. Fue desarro-



Figura 2.5: Logo de TensorFlow [30]

llada con el objetivo de facilitar la experimentación rápida, permitiendo a los usuarios pasar de la idea al resultado de manera ágil, lo cual es fundamental para realizar una buena investigación. Las características principales de Keras son su simplicidad, flexibilidad y potencia. Keras reduce la carga cognitiva del desarrollador, permitiéndole concentrarse en los aspectos más importantes del problema. Su diseño flexible se basa en el principio de revelación progresiva de la complejidad: los flujos de trabajo simples deben ser rápidos y fáciles de implementar, mientras que los flujos de trabajo avanzados deben ser posibles mediante un camino claro que se apoya en los conocimientos previos del usuario. En cuanto a potencia, Keras ofrece un rendimiento y una escalabilidad de nivel industrial, siendo utilizada por organizaciones y empresas de renombre, incluyendo NASA, YouTube y Waymo. Esto subraya su capacidad para manejar proyectos de aprendizaje profundo a gran escala y con requerimientos complejos [31].



Figura 2.6: Logo de Keras [31]

2.3. Conjunto de Datos

Para poder realizar el entrenamiento de un modelo de RNA se requiere de un conjunto de datos (dataset) que contenga información histórica de los inmuebles a evaluar [3]. Dentro de las opciones que tenemos para obtener un dataset se encuentran:

- **Construir un dataset propio:** Se requeriría de un formulario para recabar información y de un mecanismo para almacenarla. Sin embargo, la cantidad de información sería mínima y no nos permitiría obtener un modelo representativo del mercado en cuestión.

- **Utilizar un dataset existente:** En este punto se podrían utilizar los datos del catastro, o aquellos disponibles de alguna investigación similar previa. El problema con usar los valores catastrales es que no son confiables (en términos de precios de mercado) y están sujetos a múltiples factores que no son considerados en la valuación inmobiliaria tradicional [19]. Referente a investigaciones previas, los datos que pudieron observarse están desactualizados (no representan un valor de mercado actualizado), o bien, no se encuentran disponibles para la Ciudad de México y corresponden a otras ciudades del país.
- **Obtener un dataset mediante *Web Scraping*:** En este caso, se crearía un dataset a partir de la información disponible en algún portal inmobiliario. Para esto, se requiere de la construcción de un *web scraper* el cual permita extraer la información de interés y la almacene en una base de datos. El problema sería realizar la limpieza de los datos obtenidos a bien de no entrenar el modelo con información errónea.

2.3.1. Conjuntos de Datos ya existentes

Con el objetivo de encontrar un conjunto de datos que pudiera ser utilizado para el entrenamiento de un modelo de regresión, se realizó una búsqueda en diferentes plataformas de datos abiertos. En la Figura 2.7 se muestra el único conjunto encontrado que contiene información de la Ciudad de México, el cual corresponde a los valores catastrales de la Ciudad de México.

Datos Catastrales de la Ciudad de México

A pesar de que el conjunto de datos del catastro de la Ciudad de México es el único dataset que contiene información de la Ciudad de México, existen múltiples limitantes que impiden su uso para el entrenamiento de un modelo de RNA:

Los datos catastrales a menudo no reflejan las condiciones actuales del mercado inmobiliario. El catastro se enfoca en la descripción física y la ubicación de la propiedad, pero generalmente no actualiza los valores de manera frecuente para reflejar las tendencias actuales del mercado. Esto significa que los precios catastrales pueden estar desactualizados y no coincidir con los precios de mercado reales, que son influenciados por factores dinámicos como la demanda y oferta actual, el estado económico general, y las tendencias específicas de la zona [19].

Por otro lado, los valores catastrales a menudo se calculan con fines fiscales y pueden estar basados en criterios que no necesariamente coinciden con los que


**GOBIERNO DE LA
CIUDAD DE MÉXICO**

**SISTEMA ABIERTO DE INFORMACIÓN
GEOGRÁFICA DE LA CIUDAD DE MÉXICO**

**SECRETARÍA
DE DESARROLLO
URBANO Y VIVIENDA**

Sobre los Datos

El SIGCDMX cuenta con tres fuentes de datos:

- El catastro de la Ciudad de México. Administrado por la Secretaría de Administración y Finanzas.
- El registro de Uso de Suelo por predio administrado por la Secretaría de Desarrollo Urbano y Vivienda.
- Datos del Registro Público de la Propiedad a nivel colonia.

[Descarga de datos del catastro](#)
[Diccionario de datos del catastro](#)
[Ejemplo del cálculo de impuesto predial](#)
[Descarga de datos SEDUVI](#)
[Diccionario de datos SEDUVI](#)
[Capas adicionales](#)

Descarga de datos del catastro

Descarga en esta sección los datos del catastro en formato de archivo plano o geográfico por alcaldía. En el formato de archivo plano se encuentran todos los datos por cuenta catastral. En el archivo geográfico, se encuentran todos los polígonos por predio. Un predio puede tener varias cuentas catastrales. Para relacionar una cuenta a su predio, utiliza la columna FID para relacionar la(s) cuenta(s) catastral(es) con su respectivo polígono.

Alcaldías	Formatos de archivo plano	Formato de archivo geográfico
Álvaro Obregón	↓ Descargar CSV	↓ Descargar Shapefile
Azcapotzalco	↓ Descargar CSV	↓ Descargar Shapefile
Benito Juárez	↓ Descargar CSV	↓ Descargar Shapefile

Figura 2.7: Conjunto de Datos del Catastro de la Ciudad de México [32].

determinan el valor de mercado de una propiedad. Por ejemplo, un catastro puede valorar las propiedades basándose en su tamaño y ubicación, pero no necesariamente toma en cuenta factores como el estado de la propiedad, las mejoras realizadas, o características únicas que podrían aumentar su valor de mercado. Además, en algunos casos, los valores catastrales pueden estar influenciados por políticas fiscales, lo que podría llevar a una subestimación o sobreestimación del valor real del inmueble en el mercado [32].

2.3.2. Web Scraping

El *web scraping* es la técnica en la que se extraen datos de una o varias páginas web para almacenarlos en una base de datos de manera de que resulte sencillo analizar o visualizar la información [33].

Para realizar *web scraping* se requiere de un *web scraper*, el cual es un programa que se encarga de extraer la información de interés de una página web. Para esto, el *web scraper* realiza una petición a la página.

Selenium

Selenium es un proyecto que agrupa herramientas y bibliotecas para la automatización de navegadores web usando Python. Ofrece extensiones para emular la interacción del usuario con los navegadores y un servidor de distribución para la asignación de navegadores. Impulsado por contribuyentes voluntarios, Selenium facilita la escritura de código intercambiable para todos los navegadores principales bajo la especificación W3C WebDriver. El proyecto, que reúne a vendedores de navegadores, ingenieros y entusiastas, organiza una conferencia anual para educar y nutrir a la comunidad. El núcleo de Selenium es WebDriver, una interfaz para escribir conjuntos de instrucciones ejecutables en múltiples navegadores [34].

Este proyecto se caracteriza por ser una herramienta de automatización de navegadores web, por lo que puede ser capaz de emular comportamientos de usuarios mucho más avanzados que otras herramientas de *web scraping*.

En la Figura 2.8 se muestra un diagrama general de arquitectura de Selenium.

Beautiful Soup

Beautiful Soup es una biblioteca de Python utilizada para extraer datos de archivos HTML y XML. Funciona junto con tu analizador de sintaxis preferido, ofreciendo maneras idiomáticas de navegar, buscar y modificar el

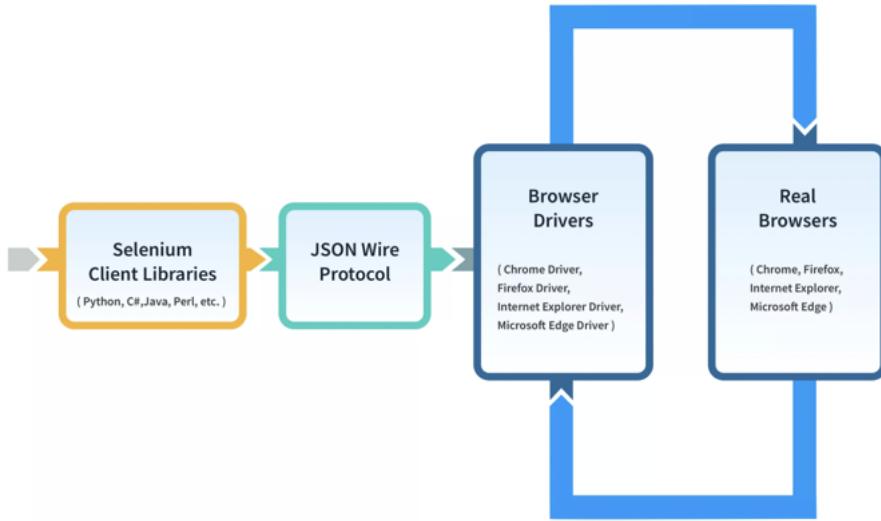


Figura 2.8: Arquitectura de Selenium [34].

árbol de análisis. Esta biblioteca suele ahorrar a los programadores horas o días de trabajo [35].

Esta librería se caracteriza por enfocarse en la extracción de datos de archivos cuyo contenido sea estático y por lo tanto no se recomienda su uso en páginas cuyo contenido sea dinámico [35]. La web hoy en día está compuesta en su gran mayoría por páginas dinámicas, por lo que el uso de esta librería no es recomendable si se desea extraer información de este tipo de páginas.

Scrapy

Scrapy es un marco de aplicación para rastrear sitios web y extraer datos estructurados que pueden ser utilizados en una amplia gama de aplicaciones útiles, como minería de datos, procesamiento de información o archivo histórico. Aunque originalmente se diseñó para el web scraping, también es útil para extraer datos mediante APIs o como un *web scraper* de propósito general [36].

cURL

cURL es un proyecto enfocado en dos productos principales: curl, una herramienta de línea de comandos, y libcurl, una biblioteca de transferencia con una API en C. Ambos productos realizan transferencias en Internet de

recursos especificados como URLs mediante protocolos de Internet. cURL se dedica a todo lo relacionado con las transferencias de protocolos de Internet, evitando manejar los datos transferidos en sí. Por ejemplo, carece de conocimiento sobre HTML, pero domina cómo transferir esos datos a través de HTTP. Estos productos se utilizan no solo en una multitud de scripts y aplicaciones conectadas a Internet, sino también en pruebas de servidores y experimentación con protocolos, estando presentes en una variedad de dispositivos integrados donde se requieren transferencias por Internet [37].

A pesar de que cURL no es necesariamente una herramienta de *web scraping*, es una herramienta que permite realizar peticiones HTTP de forma clara y transparente, por lo que puede ser utilizada para realizar *web scraping* en situaciones dónde se requieren emular comportamientos de usuarios de forma más avanzada que con otras herramientas de *web scraping*.

2.3.3. Archivos CSV

Un archivo Comma Separated Values (Valores Separados por Comas) es un archivo de texto que contiene valores separados por comas. Estos archivos se utilizan para el intercambio de datos entre diferentes aplicaciones. Por ejemplo, bases de datos y hojas de cálculo. Los archivos CSV contienen datos tabulares, es decir, datos presentados en columnas separadas por un carácter separador (generalmente una coma, pero a veces un punto y coma o una pestaña) y filas (líneas). Cada fila debe contener la misma cantidad de campos [38].

El formato de valores separados por comas (CSV) se ha utilizado para intercambiar y convertir datos entre varios programas de hojas de cálculo durante bastante tiempo. Aunque este formato es muy común, nunca ha sido documentado formalmente. Además, a pesar de que el árbol de registro MIME de IANA incluye un registro para el tipo "text/tab-separated-values", nunca se han registrado tipos MIME para CSV con IANA. Diferentes programas y sistemas operativos han comenzado a usar diferentes tipos MIME para este formato. Este RFC documenta el formato de los archivos CSV y registra formalmente el tipo MIME "text/csv" para CSV de acuerdo con el RFC 2048 [38].

Aunque existen varias especificaciones e implementaciones para el formato CSV, no existe una especificación formal, lo que permite una amplia variedad de interpretaciones de los archivos CSV. Esta sección documenta el formato que parece ser seguido por la mayoría de las implementaciones:

1. Cada registro se ubica en una línea separada, delimitada por un salto

de línea (CRLF). Por ejemplo:

aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF

2. El último registro en el archivo puede o no tener un salto de línea final. Por ejemplo:

aaa,bbb,ccc CRLF
zzz,yyy,xxx

3. Puede haber una línea de encabezado opcional que aparece como la primera línea del archivo con el mismo formato que las líneas de registro normales. Este encabezado contendrá nombres correspondientes a los campos en el archivo y debe contener el mismo número de campos que los registros en el resto del archivo (la presencia o ausencia de la línea de encabezado debe indicarse mediante el parámetro opcional "header" de este tipo MIME). Por ejemplo:

field_name,field_name,field_name CRLF
aaa,bbb,ccc CRLF
zzz,yyy,xxx CRLF

Debido a la naturaleza del proyecto, los archivos CSV juegan un papel fundamental en el almacenamiento de los datos obtenidos mediante *web scraping*. Y serán referidos en múltiples puntos del mismo para detallar su estructura y contenido.

2.4. Servicios Web

Internet representa una red global que interconecta una inmensa cantidad de computadoras de distintos tipos y redes. Los servicios web en la World Wide Web actúan como métodos estandarizados para facilitar la comunicación entre aplicaciones cliente y servidor. Estos servicios son módulos de software diseñados para realizar funciones específicas y son accesibles en la computación en la nube a través de la red. Su principal función es proporcionar capacidades específicas a los clientes que los solicitan [39].

Los servicios web se definen como un conjunto de protocolos y estándares abiertos que posibilitan el intercambio de datos entre diferentes aplicaciones o sistemas. Son fundamentales para permitir la interacción entre programas escritos en diversos lenguajes de programación y ejecutados en múltiples plataformas, facilitando así la transferencia de datos a través de redes como

Internet. Estos servicios utilizan protocolos web estandarizados, principalmente HTTP o HTTPS, y a menudo intercambian información en formato XML o JSON, lo que permite una interacción fluida y eficaz entre clientes y servidores a través de la red [40].

En la Figura 2.9 se muestra un diagrama general de arquitectura de un servicio web.

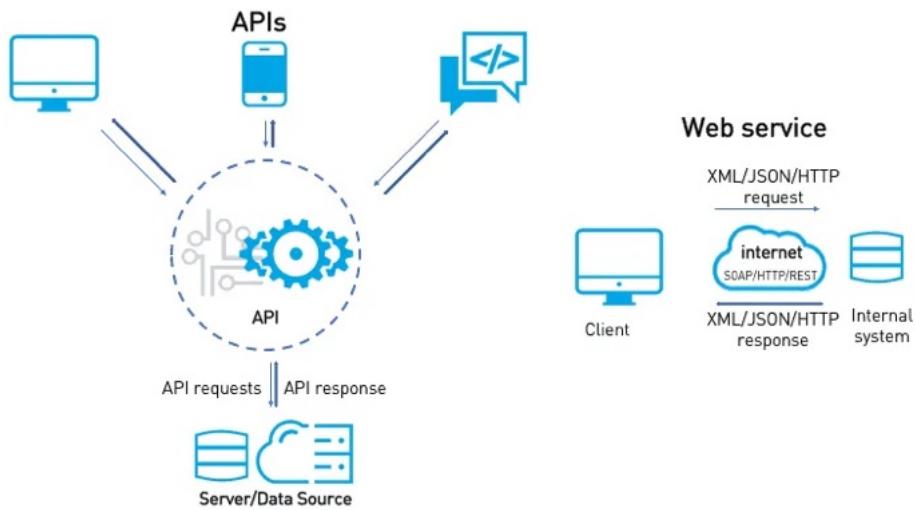


Figura 2.9: Arquitectura de un Servicio Web [41].

2.4.1. Python

Python es un lenguaje de programación de alto nivel, interpretado y orientado a objetos, que se caracteriza por su dinamismo semántico. Sus estructuras de datos integradas de alto nivel, junto con la tipificación y vinculación dinámicas, lo hacen especialmente atractivo para el Desarrollo Rápido de Aplicaciones, así como para ser utilizado como lenguaje de script o de “pegamiento” para integrar componentes ya existentes. La sintaxis de Python es sencilla y fácil de aprender, lo que destaca su legibilidad y reduce así el costo del mantenimiento de programas. Python apoya el uso de módulos y paquetes, fomentando la modularidad de los programas y la reutilización de código. El intérprete de Python y su amplia biblioteca estándar están disponibles de forma gratuita, tanto en código fuente como en forma binaria, para todas las plataformas principales y pueden distribuirse libremente [42].



Figura 2.10: Logo de Python [42].

Usos de Python

- Desarrollo web (lado del servidor)
- Desarrollo de software
- Matemáticas
- Scripting de sistemas

¿Qué puede hacer Python?

- Puede usarse en un servidor para crear aplicaciones web.
- Puede emplearse junto con software para crear flujos de trabajo.
- Puede conectarse a sistemas de bases de datos y leer y modificar archivos.
- Es útil para manejar grandes volúmenes de datos y realizar matemáticas complejas.
- Sirve tanto para prototipos rápidos como para el desarrollo de software listo para producción.

¿Por qué Python?

- Funciona en diferentes plataformas (Windows, Mac, Linux, Raspberry Pi, etc.).
- Tiene una sintaxis simple similar al idioma inglés.
- La sintaxis de Python permite a los desarrolladores escribir programas con menos líneas que en otros lenguajes de programación.
- Es un lenguaje interpretado, lo que permite ejecutar el código tan pronto como se escribe, facilitando un prototipado rápido.

- Puede tratarse de manera procedural, orientada a objetos o funcional.

Sintaxis de Python comparada con otros lenguajes de programación

- Fue diseñado para ser legible, con similitudes al idioma inglés y con influencia de las matemáticas.
- Utiliza nuevas líneas para completar un comando, a diferencia de otros lenguajes de programación que a menudo usan punto y coma o paréntesis.
- Se basa en la indentación, utilizando espacios en blanco, para definir el ámbito, como en bucles, funciones y clases. Otros lenguajes de programación a menudo usan llaves para este propósito.

Algunos de los *frameworks* más utilizados para el desarrollo web con Python son Django, Flask y FastAPI. A continuación se abordará cada uno de ellos.

Flask

Flask es un marco de trabajo (*framework*) web escrito en Python, conocido por ser un módulo que facilita el desarrollo de aplicaciones web. Se caracteriza por tener un núcleo pequeño y fácil de ampliar, clasificándose como un microframework. A diferencia de otros frameworks, Flask no incluye un ORM (Object Relational Manager) ni características similares [43].



Figura 2.11: Logo de Flask [43].

Entre sus características destacadas se encuentran la capacidad de enruteamiento de URL y un motor de plantillas. Flask se basa en el conjunto de herramientas WSGI (Web Server Gateway Interface) de Werkzeug y en el motor de plantillas Jinja2, ambos proyectos de Pocco [44].

Un marco de trabajo web es una colección de bibliotecas y módulos que permiten a los desarrolladores de aplicaciones web escribir aplicaciones sin preocuparse por detalles de bajo nivel como el protocolo o la gestión de

hilos. WSGI, en particular, es una especificación de interfaz común entre servidores web y aplicaciones web. Werkzeug implementa objetos de solicitud y respuesta, y funciones de utilidad, lo que permite la construcción de marcos web sobre él. Jinja2 es un motor de plantillas que permite integrar variables de Python en plantillas HTML [44].

Flask se considera un microframework porque mantiene el núcleo de la aplicación simple y escalable, permitiendo extensiones para agregar capacidades como soporte de base de datos. A diferencia de Django, Flask es muy “Pythonic” y tiene una curva de aprendizaje baja. Su simplicidad y explicitud aumentan la legibilidad y permiten empezar proyectos rápidamente con pocas líneas de código. Flask permite desarrollar tanto en un entorno local como en línea, y es popular por ser actualizado, moderno y extensible, permitiendo al desarrollador tomar decisiones sobre bases de datos, ORM, etc. Flask es adecuado tanto para prototipos rápidos como para el desarrollo de software listo para la producción y puede manejar aplicaciones complejas, a pesar de su naturaleza de microframework [43].

Django

Django es un framework web de Python de alto nivel diseñado para el desarrollo rápido de sitios web seguros y fáciles de mantener, destacándose por su comunidad activa, código abierto y amplias opciones de soporte [45].



Figura 2.12: Logo de Django [45].

Django se caracteriza por enfocarse en las siguientes características:

- **Completo:** Ofrece una amplia gama de funcionalidades integradas para un desarrollo web eficiente y coherente.
- **Versátil:** Adecuado para construir una variedad de sitios web y compatible con múltiples formatos de entrega de contenido.
- **Seguro:** Incluye medidas de seguridad incorporadas para prevenir errores comunes y proteger la web.
- **Escalable:** Su arquitectura basada en componentes independientes permite un escalado eficaz para manejar más tráfico.

- **Mantenible:** Promueve la creación de código reutilizable y bien organizado, alineado con los principios de diseño limpio.
- **Portable:** Escrito en Python, funciona en varias plataformas y es compatible con numerosos proveedores de hosting.

FastAPI

FastAPI es un moderno y rápido framework de alto rendimiento para la creación de APIs con Python 3.8 o superior, basado en las pistas de tipo estándar de Python. Este framework se destaca por su alta performance y facilidad de uso, ofreciendo una solución eficiente y escalable para el desarrollo de aplicaciones web y APIs [46].



Figura 2.13: Logo de FastAPI [46].

Las características más destacadas de FastAPI son:

- **Rápido:** Ofrece un rendimiento muy alto, comparable con NodeJS y Go, gracias a Starlette y Pydantic, siendo uno de los frameworks de Python más rápidos disponibles.
- **Ágil para programar:** Permite aumentar la velocidad de desarrollo de características en aproximadamente un 200 % a 300 %.
- **Menos errores:** Reduce en un 40 % los errores inducidos por los desarrolladores.
- **Intuitivo:** Ofrece un gran soporte para editores de código, con completado automático en todas partes, lo que disminuye el tiempo dedicado a la depuración.
- **Fácil de usar:** Diseñado para ser fácil de aprender y utilizar, minimizando el tiempo necesario para leer documentación.
- **Conciso:** Minimiza la duplicación de código. Múltiples características a partir de cada declaración de parámetro, lo que conlleva a menos errores.
- **Robusto:** Genera código listo para producción, con documentación interactiva automática.

- **Basado en estándares:** Se basa en (y es completamente compatible con) los estándares abiertos para APIs: OpenAPI (anteriormente conocido como Swagger) y JSON Schema.

2.4.2. API

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de definiciones y protocolos para construir e integrar software de aplicaciones. Funciona como un contrato entre un proveedor de información y un usuario, especificando lo que se necesita tanto en la solicitud (llamada) como en la respuesta. Permite la interacción con un sistema para obtener información o realizar funciones, actuando como mediador entre los usuarios o clientes y los recursos o servicios web que desean obtener. Las APIs son clave para compartir recursos e información manteniendo la seguridad, el control y la autenticación, y simplifican la interacción sin necesidad de conocer detalles específicos como el almacenamiento en caché. [47].

REST

REST (Representational State Transfer) es un conjunto de restricciones arquitectónicas, no un protocolo o estándar, que los desarrolladores de API pueden implementar de diversas maneras. Cuando se realiza una solicitud a través de una API REST, se transfiere una representación del estado del recurso al solicitante o endpoint, en formatos como JSON, HTML, XML, Python, PHP o texto plano, siendo JSON el más popular por su agnosticismo de lenguaje y legibilidad tanto para humanos como para máquinas [47]. En la Figura 2.14 se muestra un diagrama general de arquitectura de una API REST.

Para que una API sea considerada RESTful, debe cumplir con ciertos criterios: una arquitectura cliente-servidor con comunicación sin estado, datos cacheables, una interfaz uniforme para la transferencia estándar de información, y un sistema en capas. Incluye detalles como headers y parámetros en los métodos HTTP, importantes para la metadata y autorización. A pesar de sus criterios específicos, REST es más fácil de usar y flexible que protocolos como SOAP, lo que la hace más rápida, ligera y escalable, ideal para el desarrollo de aplicaciones móviles e Internet de las Cosas (IoT) [47].

2.4.3. JSON

JavaScript Object Notation (JSON) es un formato estándar basado en texto para representar datos estructurados basados en la sintaxis de objetos de

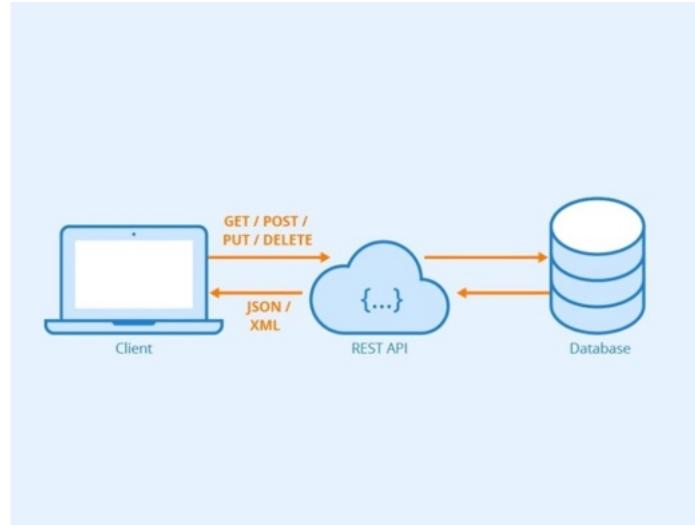


Figura 2.14: Arquitectura de una API REST [48].

JavaScript. Se utiliza comúnmente para la transmisión de datos en aplicaciones web, tanto del servidor al cliente como viceversa. JSON es un formato de datos basado en texto que sigue la sintaxis de objeto de JavaScript, popularizado por Douglas Crockford. Aunque se parece a la sintaxis literal de objeto de JavaScript, puede usarse de manera independiente y es compatible con muchos entornos de programación. JSON existe como una cadena de texto y necesita convertirse a un objeto JavaScript nativo para acceder a los datos. Esta conversión se conoce como deserialización, mientras que convertir un objeto nativo a una cadena para su transmisión se llama serialización. Una cadena JSON puede almacenarse en un archivo propio, con extensión .json y tipo MIME de application/json [49].

2.4.4. SSL

SSL (*Secure Sockets Layer*) es una tecnología de encriptación desarrollada originalmente por Netscape en la década de 1990. SSL establece una conexión encriptada entre el servidor web y el navegador del visitante, permitiendo la transmisión segura de información privada y protegiéndola contra la interceptación, manipulación de datos y falsificación de mensajes. Para habilitar SSL en un sitio web, se requiere obtener e instalar un Certificado SSL en el servidor web. Este certificado puede ser identificado en los navegadores, a menudo mediante un ícono de candado o una barra de direcciones verde. La activación de SSL se realiza cambiando el URL de `http://` a `https://`, asegurando la encriptación de la información transmitida. Los Certificados SSL

son emitidos por Autoridades Certificadoras (CAs) y son ampliamente utilizados por negocios en línea para proteger sus sitios web y generar confianza en sus clientes [50].

2.4.5. HTTPS

HTTPS (*Hypertext Transfer Protocol Secure*) es la versión segura de HTTP, el protocolo principal para enviar datos entre un navegador web y un sitio web. HTTPS encripta los datos para aumentar la seguridad en la transferencia, lo cual es crucial cuando los usuarios transmiten información sensible, como al iniciar sesión en cuentas bancarias, servicios de correo electrónico o proveedores de seguros de salud. Los sitios web, en especial aquellos que requieren credenciales de inicio de sesión, deben utilizar HTTPS. Los navegadores modernos, como Chrome, marcan los sitios que no utilizan HTTPS como no seguros. Un candado en la barra de URL indica que la página web es segura [51].

2.4.6. DNS

DNS (*Domain Name System*) es un sistema que traduce el nombre de una máquina en red (host) a una dirección IP legible por máquina, asegurando que los paquetes se enrutan correctamente en la red. Por razones de seguridad, un servidor en la red puede usar una “búsqueda inversa” para confirmar a sus administradores que las personas adecuadas se están conectando a él. Algunos servidores web o ftp no permiten conexiones a menos que puedan mapear inversamente la dirección IP a un nombre de host registrado [52].

2.4.7. CDN

Un CDN (*Content Delivery Network*) es un conjunto de servidores distribuidos en múltiples ubicaciones. Estos servidores almacenan copias duplicadas de datos para responder a solicitudes basándose en cuáles están más cerca de los usuarios finales, lo que resulta en un servicio rápido y menos afectado por alto tráfico. Los CDNs son ampliamente utilizados para entregar hojas de estilo y archivos JavaScript (activos estáticos) de bibliotecas como Bootstrap, jQuery, etc. Usar un CDN para estos archivos es preferible por varias razones: reduce la carga de solicitudes en los servidores propios de una organización, los servidores de CDN pueden estar geográficamente más cerca de los usuarios, y los CDN ya están configurados con ajustes de caché adecuados, lo que ahorra configuración adicional en los propios servidores [53].

2.4.8. Docker

Docker es una plataforma abierta para desarrollar, distribuir y ejecutar aplicaciones, permitiendo separar las aplicaciones de la infraestructura para agilizar la entrega de software. Con Docker, se puede gestionar la infraestructura de la misma manera que las aplicaciones, aprovechando sus metodologías para distribuir, probar y desplegar código, reduciendo así el tiempo entre la escritura del código y su ejecución en producción [54].

La plataforma Docker

Docker posibilita empaquetar y ejecutar aplicaciones en un entorno aislado llamado contenedor, que es seguro y permite la ejecución simultánea de múltiples contenedores en un mismo host. Los contenedores son ligeros y contienen todo lo necesario para ejecutar la aplicación, lo que elimina la dependencia del software instalado en el host [54]. Docker también proporciona herramientas y una plataforma para gestionar el ciclo de vida de los contenedores, lo que incluye:

- Desarrollar la aplicación y sus componentes de soporte usando contenedores.
- El contenedor se convierte en la unidad para distribuir y probar la aplicación.
- Al estar listo, desplegar la aplicación en el entorno de producción, ya sea como un contenedor o un servicio orquestado, funcionando igual en un centro de datos local, un proveedor de nube o una combinación de ambos.

En la Figura 2.15 se muestra un diagrama general de arquitectura de Docker.

2.5. Aplicación Web

Una aplicación web es una aplicación invocada a través de un navegador web en Internet. Desde su auge en la década de 1990, la web ha evolucionado de ser un repositorio de páginas estáticas a una poderosa plataforma para el desarrollo y despliegue de aplicaciones innovadoras. Las aplicaciones web actuales permiten una nueva modalidad de cooperación y colaboración entre usuarios, soportando tanto datos estructurados como no estructurados, y ofreciendo interfaces de navegación exploratorias. Las aplicaciones web modernas, dentro del concepto de Web 2.0, son dinámicas e invitan a la participación del usuario, siendo más responsivas y cercanas a las aplicaciones

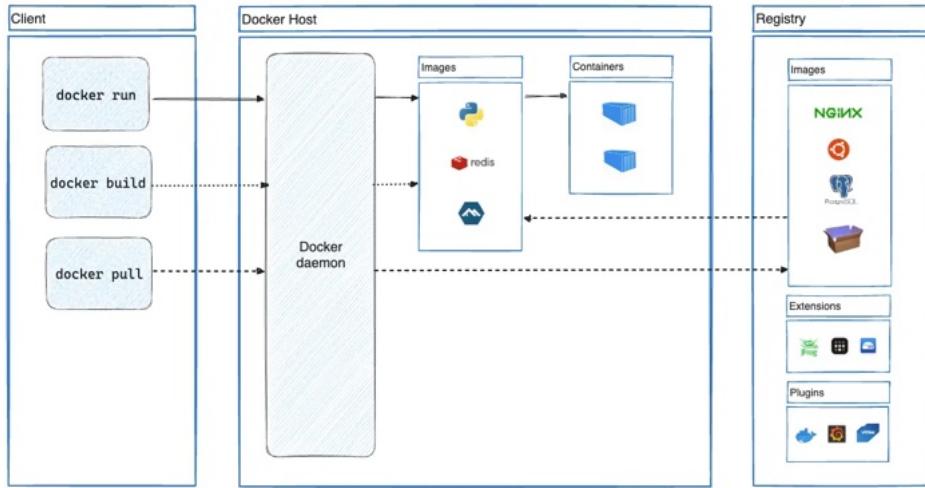


Figura 2.15: Arquitectura de Docker [54].

de escritorio. El desarrollo de aplicaciones web ha adoptado técnicas de ingeniería de software como la orientación a servicios, y sigue evolucionando rápidamente, anticipando nuevas tendencias y enfoques en el futuro [55].

2.5.1. JavaScript

JavaScript (JS) es un lenguaje de programación ligero e interpretado (o compilado justo a tiempo) con funciones de primera clase. Aunque es más conocido como el lenguaje de scripting para páginas web, también se utiliza en entornos fuera del navegador, como Node.js, Apache CouchDB y Adobe Acrobat. JS es un lenguaje basado en prototipos, de paradigma múltiple, de un solo hilo y dinámico, que admite estilos de programación orientada a objetos, imperativa y declarativa. Las capacidades dinámicas de JavaScript incluyen la construcción de objetos en tiempo de ejecución, listas de parámetros variables, variables de función, creación dinámica de scripts (a través de `eval`), introspección de objetos y recuperación del código fuente. Este lenguaje se rige por los estándares ECMAScript (ECMA-262) y la API de Internacionalización ECMAScript (ECMA-402). Es importante no confundir JavaScript con el lenguaje de programación Java, ya que, a pesar de sus nombres similares, tienen sintaxis, semánticas y usos muy diferentes [56].

2.5.2. HTML

HTML, o *Hypertext Markup Language*, es el lenguaje de marcado estándar utilizado para crear páginas web. Es uno de los pilares fundamentales de la



Figura 2.16: Logo de JavaScript [56].

World Wide Web y es utilizado para estructurar el contenido y presentarlo en Internet. HTML describe la estructura de una página web semánticamente y, originalmente, incluía señales para la apariencia del documento [57].



Figura 2.17: Logo de HTML [57].

HTML se compone de una serie de elementos que se utilizan para encerrar o envolver diferentes partes del contenido para hacerlo actuar de cierta manera, o para mostrarlo de cierta manera. Estos elementos están representados por etiquetas, escritas con corchetes angulares. Por ejemplo, `<p>` representa un párrafo. Algunas etiquetas tienen pares de apertura y cierre que encierran el contenido, mientras que otras, como ``, son autocontenidoas [57].

Con el tiempo, HTML ha evolucionado significativamente. La versión actual, HTML5, introduce una gran cantidad de tecnología y API para facilitar la creación de sitios web y aplicaciones web complejas y ricas. HTML5 está diseñado para ser utilizado en conjunto con CSS y JavaScript, permitiendo a los desarrolladores crear experiencias de usuario interactivas y atractivas en la web [57].

2.5.3. CSS

CSS, o *Cascading Style Sheets*, es un lenguaje de diseño gráfico utilizado para definir y crear la presentación de un documento estructurado escrito en HTML o XML. CSS describe cómo se deben mostrar los elementos en la pantalla, en papel, en el habla o en otras formas de medios. Es fundamental

para la creación de páginas web visualmente atractivas y profesionalmente estilizadas [58].



Figura 2.18: Logo de CSS [58].

Además de definir estilos como fuentes, colores y espacio, CSS permite una gran flexibilidad y control en la disposición visual de los elementos en una página web. Se pueden establecer estilos específicos para diferentes dispositivos y tamaños de pantalla, lo que permite que los sitios web sean responsivos y adaptables. CSS también mejora la accesibilidad del contenido web y reduce la complejidad del código HTML al separar la estructura del contenido de su presentación visual [58].

A pesar de que CSS en sí mismo ofrece una flexibilidad significativa, los *frameworks* CSS permiten un desarrollo web más rápido y eficiente, proporcionando una base sólida para el diseño de sitios web. Algunos de los *frameworks* CSS más utilizados son Bootstrap, Material UI y Tailwind.

Bootstrap

Bootstrap es un marco (*framework*) de código abierto para desarrollo web front-end, que utiliza HTML, CSS y JavaScript. Su propósito es facilitar el diseño de sitios y aplicaciones web responsivas y móviles. Bootstrap proporciona una amplia gama de herramientas reutilizables como sistemas de grillas, componentes predefinidos y potentes plugins basados en JavaScript. Diseñado inicialmente por Twitter, Bootstrap se ha convertido en una de las herramientas más populares para el desarrollo web, gracias a su enfoque en el diseño adaptativo y la compatibilidad entre navegadores [59]. En la Figura 2.19 se muestra un ejemplo de un sitio web desarrollado con Bootstrap.

Material UI

Material-UI es un *framework* de interfaz de usuario para React que implementa los principios de diseño de Material Design de Google. Proporciona un

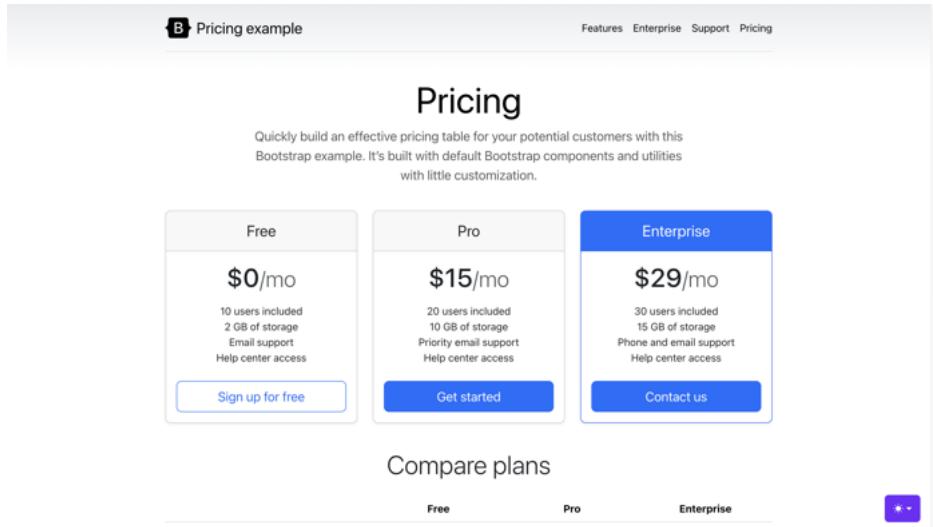


Figura 2.19: Ejemplo de sitio web desarrollado con Bootstrap [59].

conjunto de componentes de React reutilizables y personalizables que siguen las pautas de Material Design, permitiendo desarrollar interfaces de usuario estéticamente agradables y consistentes de manera eficiente. Material-UI es conocido por su enfoque en la interactividad y la experiencia del usuario, ofreciendo una amplia variedad de elementos de interfaz, desde simples botones y campos de texto hasta componentes más complejos como diálogos y barras de navegación. Su diseño modular y personalizable lo hace popular entre los desarrolladores de aplicaciones web modernas [60]. En la Figura 2.20 se muestra un ejemplo de un sitio web desarrollado con Material-UI.

2.5.4. React

React es una biblioteca de JavaScript para construir interfaces de usuario. Es mantenido por Facebook y una comunidad de desarrolladores individuales y empresas. React permite a los desarrolladores crear aplicaciones web grandes que pueden cambiar datos, sin recargar la página. Su objetivo principal es ser rápido, escalable y simple. Se puede usar con una combinación de otras bibliotecas o marcos de JavaScript, como Angular JS en MVC [61].

Una característica clave de React es su enfoque en componentes. Los usuarios definen componentes que manejan su propio estado, luego componen para formar interfaces de usuario complejas. En lugar de trabajar con el modelo de objetos de documento (DOM) del navegador, React crea un DOM virtual, donde se realizan sus componentes y la lógica de estado. Esto proporciona un

The screenshot shows a pricing section of a website. At the top, there is a navigation bar with a search input labeled "Company name" and links for "FEATURES", "ENTERPRISE", "SUPPORT", and "LOGIN". Below the navigation, the word "Pricing" is centered in a large, bold font. A descriptive text follows: "Quickly build an effective pricing table for your potential customers with this layout. It's built with default MUI components with little customization." The main content is a three-column pricing table:

Free	Pro Most popular	Enterprise
\$0/mo	\$15/mo	\$30/mo
10 users included 2 GB of storage Help center access Email support	20 users included 10 GB of storage Help center access Priority email support	50 users included 30 GB of storage Help center access Phone & email support
SIGN UP FOR FREE	GET STARTED	CONTACT US

Figura 2.20: Ejemplo de sitio web desarrollado con Material-UI [60].

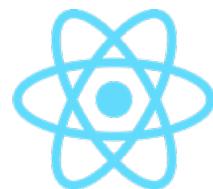


Figura 2.21: Logo de React [61].

rendimiento mejorado, ya que los cambios en el DOM virtual tienen menos costo de rendimiento que los cambios en el DOM real. Además, React utiliza un enfoque declarativo que facilita el razonamiento sobre el flujo de datos y la interfaz de usuario [61].

2.5.5. Vue

Vue.js es un marco de JavaScript progresivo utilizado para construir interfaces de usuario y aplicaciones de una sola página. Es conocido por su simplicidad, facilidad de integración y enfoque en la experiencia del desarrollador. Vue se basa en HTML, CSS y JavaScript estándar, proporcionando un modelo de programación declarativo y basado en componentes que facilita el desarrollo de interfaces de usuario, ya sean simples o complejas [62].



Figura 2.22: Logo de Vue [62].

Una de las características clave de Vue es su sistema de reactividad, que rastrea automáticamente los cambios en el estado del JavaScript y actualiza el DOM de manera eficiente cuando estos cambios ocurren. Además, Vue es flexible y adaptable, lo que permite su uso de diferentes maneras según el caso de uso, incluyendo la mejora de HTML estático, la incorporación como componentes web en cualquier página, aplicaciones de una sola página, renderizado en el lado del servidor, generación de sitios estáticos y más. Su enfoque progresivo significa que puede ser tan simple o robusto como lo necesite el proyecto, adaptándose y creciendo con las necesidades del desarrollador [62].

2.6. Plataforma de nube

La *computación en la nube* se refiere a un servicio basado en suscripción donde se pueden obtener *espacio de almacenamiento en red* y *recursos informáticos*. Similar a cómo funciona un cliente de correo electrónico como Yahoo!, Gmail o Hotmail, que administra todo el hardware y software necesario para soportar una cuenta de correo electrónico personal, la computación en la nube funciona de manera análoga, pero proporcionando acceso a una variedad más amplia de información y recursos. Esta tecnología permite acceder a la

información *desde cualquier lugar y en cualquier momento*, siempre que se tenga acceso a Internet, eliminando la necesidad de estar en la misma ubicación física que el hardware que almacena los datos. Existen diferentes tipos de nubes, como la *Nube Pública*, *Nube Privada*, *Nube Comunitaria* y *Nube Híbrida*, cada una adaptada a necesidades específicas. Las empresas pueden adaptar su uso del espacio en la nube según cambien sus necesidades, lo que es especialmente útil para pequeñas empresas que buscan soluciones económicas y escalables para el almacenamiento de datos y recursos informáticos [63].

Existen múltiples proveedores de servicios de nube, como Google Cloud Platform, Microsoft Azure y Amazon Web Services, que ofrecen una variedad de servicios para el desarrollo de aplicaciones web y móviles, incluyendo almacenamiento, bases de datos, redes, análisis, aprendizaje automático e inteligencia artificial, entre otros. En la Figura 2.23 se muestra la distribución del mercado de proveedores de servicios de nube en el año 2023.

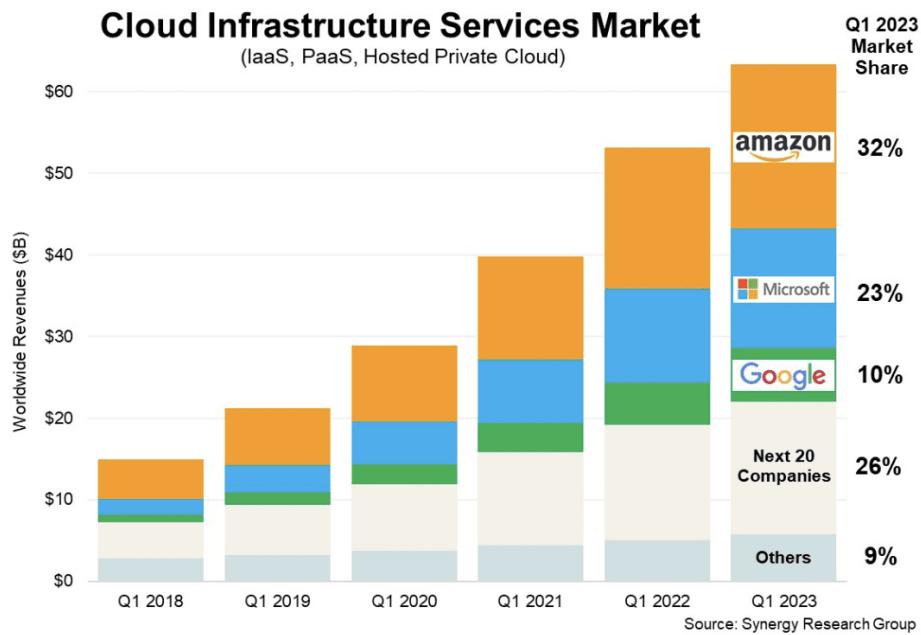


Figura 2.23: Distribución del mercado de proveedores de servicios de nube en 2023 [64].

2.6.1. Google Cloud Platform

Google Cloud Platform (GCP) ofrece una serie de productos que permiten a los usuarios aprovechar la infraestructura interna de Google. Esto

incluye recursos típicos de la nube, como máquinas virtuales bajo demanda y almacenamiento de objetos, además de APIs para tecnologías avanzadas desarrolladas por Google, como Bigtable, Cloud Datastore o Kubernetes [65].

A diferencia de otros proveedores de la nube, GCP se distingue por dos razones principales. Primero, Google ha sido la cuna de tecnologías revolucionarias, compartidas posteriormente con el mundo a través de publicaciones académicas, como MapReduce, Bigtable y Spanner. Estas innovaciones, inicialmente exclusivas de Google, ahora están disponibles para todos. Segundo, la escala de operación de Google le permite ofrecer precios más bajos gracias a su vasta infraestructura y hardware personalizado, lo que se traduce en ahorros que benefician al consumidor, similar a comprar un paquete grande de productos a precio reducido como en Costco [65]. En la Figura 2.24 se muestran algunos servicios de GCP.



Figura 2.24: Servicios de Google Cloud Platform [65].

Entre los servicios de GCP que podrían ser relevantes para el desarrollo de este proyecto se encuentran:

Cloud DNS

El servicio de DNS de Google Cloud Platform, Cloud DNS, es un servicio de DNS escalable, confiable y administrado que permite a los usuarios alojar sus zonas DNS en la infraestructura de Google. Cloud DNS traduce nombres de dominio como `www.example.com` en direcciones IP como `255.255.255.255`, lo que permite a los usuarios encontrar recursos en Internet [65].

Cloud Storage

Cloud Storage es un servicio de almacenamiento de objetos unificado para desarrolladores y empresas, que ofrece almacenamiento de objetos altamente

escalable y duradero para cualquier tipo de datos. Es un servicio de almacenamiento de objetos que permite a los usuarios almacenar y recuperar datos en Google Cloud Platform. Los objetos de Cloud Storage se almacenan en *buckets*, que son contenedores de almacenamiento de objetos. Un objeto es un archivo individual que se almacena en Cloud Storage. Los objetos de Cloud Storage se pueden organizar y controlar mediante el uso de *etiquetas* [65].

Cloud CDN

Cloud CDN es un servicio de red de entrega de contenido que proporciona distribución de contenido a través de una red global de ubicaciones de almacenamiento en caché. Cloud CDN reduce la latencia de entrega de contenido, ahorra ancho de banda y reduce los costos de origen. Cloud CDN se puede habilitar para cualquier backend que se ejecute en Compute Engine, Google Kubernetes Engine o Google Cloud Storage [65].

Cloud Run

Cloud Run es un servicio administrado que permite ejecutar contenedores sin estado que se invocan a través de solicitudes HTTP. Cloud Run es un entorno computacional completamente administrado que se encarga de la escalabilidad automática, la configuración de la instancia de contenedor y el equilibrio de carga horizontal. Cloud Run se basa en el proyecto open-source Knative [65].

2.6.2. Amazon Web Services

Amazon Web Services (AWS) es una plataforma de servicios en la nube ampliamente adoptada que proporciona una variedad de servicios de infraestructura a través de internet. AWS ofrece una gama extensa de soluciones globales de cómputo, almacenamiento, bases de datos, análisis, redes y móviles, así como herramientas de desarrollador, herramientas de gestión, IoT, seguridad y aplicaciones empresariales [66].

Con AWS, las organizaciones pueden desplegar aplicaciones y servicios de manera rápida y segura, con la flexibilidad de escalar los recursos según sea necesario. La plataforma está diseñada para ser la más flexible y segura de las plataformas de computación en la nube, ofreciendo capacidades de cómputo de alto rendimiento y almacenamiento en la nube, así como bases de datos optimizadas para diferentes tipos de cargas de trabajo. En la Figura 2.25 se muestran algunos servicios de AWS.



Figura 2.25: Servicios de Amazon Web Services [67].

AWS es reconocido por su constante innovación y expansión de servicios, facilitando a empresas de todos los tamaños, desde startups hasta empresas Fortune 500, la transición hacia la nube y la transformación digital. Su modelo de pago por uso ayuda a optimizar los costos y a operar infraestructuras más eficientes [66].

Dentro de los servicios de AWS que podrían ser relevantes para el desarrollo de este proyecto se encuentran:

Elastic Beanstalk

AWS Elastic Beanstalk es un servicio que permite a los desarrolladores desplegar rápidamente y administrar aplicaciones en la nube. Las aplicaciones se pueden desplegar utilizando una variedad de plataformas, incluyendo Java, .NET, PHP, Node.js, Python, Ruby, Go y Docker, en servidores web comunes como Apache, Nginx, Passenger y IIS [67].

ECR

Amazon Elastic Container Registry (ECR) es un servicio de registro de Docker totalmente administrado que facilita el almacenamiento, la administración y la implementación de imágenes de Docker. Los usuarios pueden almacenar, administrar y desplegar imágenes de Docker en AWS, lo que permite a los desarrolladores desplegar aplicaciones distribuidas sin necesidad de ejecutar su propio software de registro de contenedores [67].

S3

Amazon Simple Storage Service (S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendi-

miento incomparables. S3 es fácil de usar y ofrece una interfaz de servicio web simple que se puede utilizar para almacenar y recuperar cualquier cantidad de datos, en cualquier momento y desde cualquier lugar de la web. S3 también ofrece capacidades de almacenamiento de datos de nivel empresarial para crear, implementar y escalar aplicaciones, lo que resulta en un acceso rápido y confiable a los datos y un rendimiento optimizado [68].

Route 53

Amazon Route 53 es un servicio de sistema de nombres de dominio (DNS) escalable y altamente disponible. Está diseñado para proporcionar a los desarrolladores y empresas una forma confiable y rentable de dirigir los usuarios finales a aplicaciones de Internet, traduciendo nombres de dominio como `www.example.com` en direcciones IP como `255.255.255.0` [68].

CloudFront

Amazon CloudFront es un servicio de red de entrega de contenido (CDN) rápido, altamente seguro y programable. CloudFront ofrece una solución de CDN global y altamente disponible para entregar contenido a través de Internet con baja latencia y altas velocidades de transferencia de datos. CloudFront funciona con otros servicios de AWS para proporcionar a los desarrolladores y empresas una forma fácil de distribuir contenido a los usuarios finales con seguridad, bajo demanda y con un alto rendimiento [68].

2.7. GIS

Los Sistemas de Información Geográfica (GIS) son herramientas computacionales diseñadas para la adquisición, almacenamiento, análisis y visualización de datos espaciales. Históricamente, han evolucionado desde la producción de mapas con enfoques cartográficos tradicionales hacia bases de datos electrónicas flexibles, permitiendo la superposición de mapas temáticos para resolver conflictos y relaciones en la gestión del uso del suelo. Los avances técnicos han integrado el análisis de imágenes de sensores remotos dentro de las prácticas estándar de GIS, superando la división entre datos en formato *raster* y *vector*. En la actualidad, los GIS comerciales ofrecen capacidades ampliadas para trabajar con ambos tipos de datos y han incluido lenguajes de programación internos que amplían su aplicación a modelos ambientales variados, aunque tradicionalmente han ignorado aspectos como la incertidumbre y la variabilidad espacio-temporal en favor de la precisión geométrica [69].

En la Figura 2.26 se muestra un ejemplo de las aplicaciones de los GIS en el análisis de datos espaciales, en este caso, aplicado a una imagen satelital y cómo se puede descomponer la información en diferentes capas.

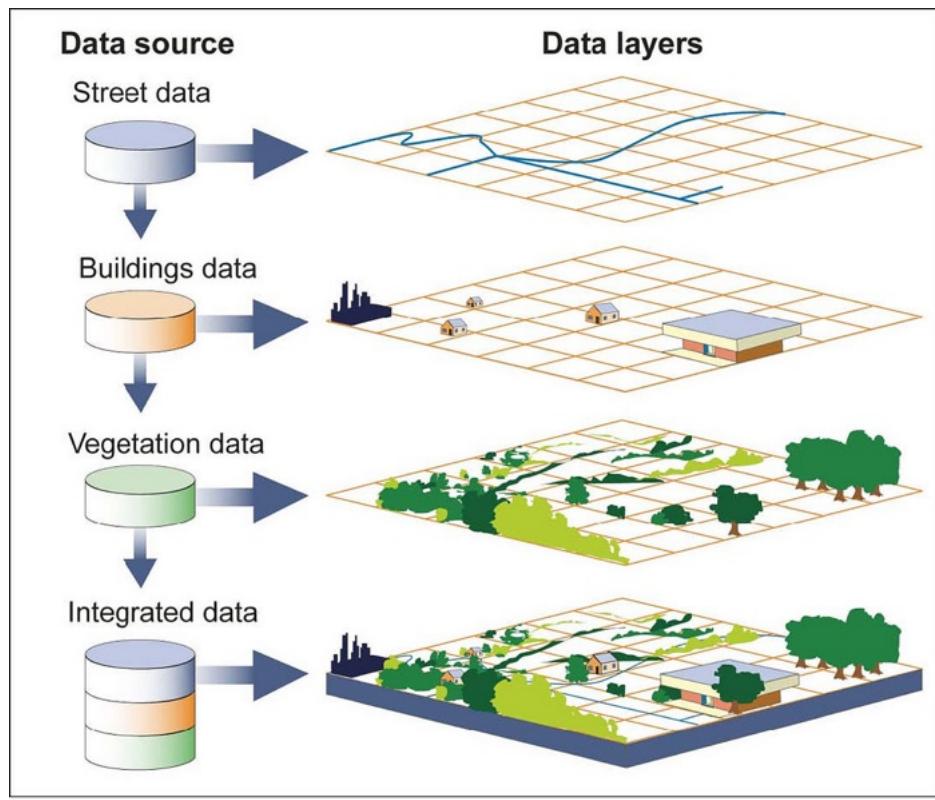


Figura 2.26: Aplicaciones de los GIS en el análisis de datos espaciales [70].

La valuación inmobiliaria es una de las aplicaciones más comunes de los GIS, permitiendo la visualización de datos espaciales y la generación de mapas temáticos para la toma de decisiones. Los GIS también se utilizan para la planificación urbana, la gestión de recursos naturales, la gestión de desastres, la gestión de infraestructura y el análisis de redes, entre otros [71].

Existen múltiples software GIS, tanto comerciales como de código abierto, que permiten la visualización y análisis de datos espaciales. A continuación se abordarán algunos de los más utilizados.

2.7.1. QGIS

QGIS es un sistema de información geográfica (SIG) de código abierto que comenzó en 2002 y opera en plataformas Unix, Windows y macOS. Desa-

rrollado con Qt y C++, ofrece una interfaz de usuario amigable, además de aplicaciones para dispositivos móviles. Inicialmente un visor de datos SIG, ahora QGIS es utilizado para visualización de datos GIS, captura de datos, análisis avanzado y presentaciones en mapas. Admite numerosos formatos de datos y se expande fácilmente con complementos. Publicado bajo la Licencia Pública General GNU (GPL), QGIS es gratuito y de código fuente abierto, asegurando acceso y modificación libres [72].

En la Figura 2.27 se muestra la interfaz de usuario de QGIS.

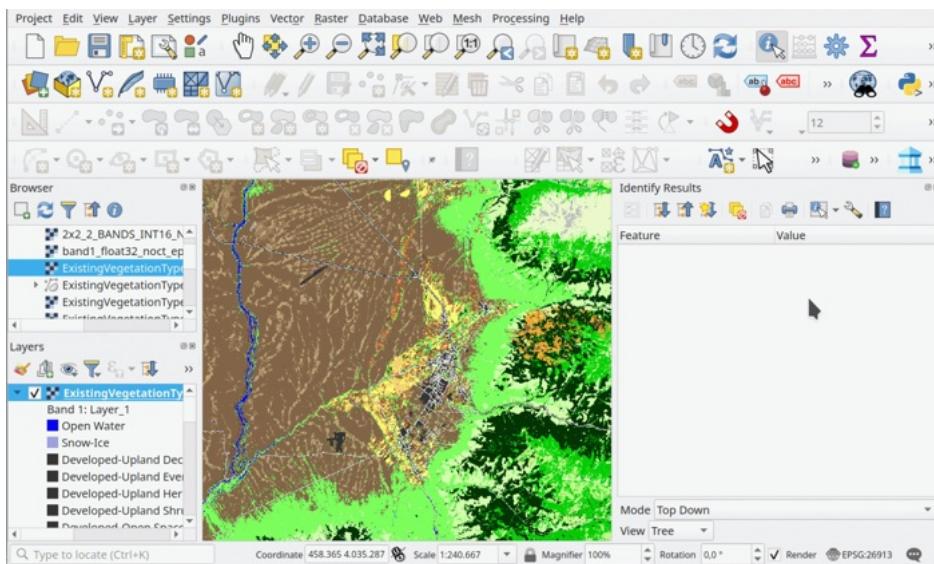


Figura 2.27: Interfaz de usuario de QGIS [73].

2.7.2. kepler.gl

Kepler.gl es una aplicación web de alto rendimiento y agnóstica a los datos, diseñada para la exploración visual de conjuntos de datos geolocalizados a gran escala. Como herramienta de *GIS*, kepler.gl se destaca por su capacidad de renderizar millones de puntos, representando miles de trayectos y realizando agregaciones espaciales al vuelo. Construido sobre tecnologías robustas como Mapbox GL y deck.gl, kepler.gl es una potente plataforma para analizar y visualizar datos espaciales en una variedad de formas, facilitando la comprensión de patrones y tendencias complejas. Además, kepler.gl funciona como un componente de React que utiliza Redux para la gestión de su estado y flujo de datos, lo que permite su integración y personalización en otras aplicaciones basadas en React-Redux, ofreciendo así una solución versátil para el análisis de datos geoespaciales en el ámbito del *GIS* [74].

En la Figura 2.28 se muestra la interfaz de usuario de kepler.gl.

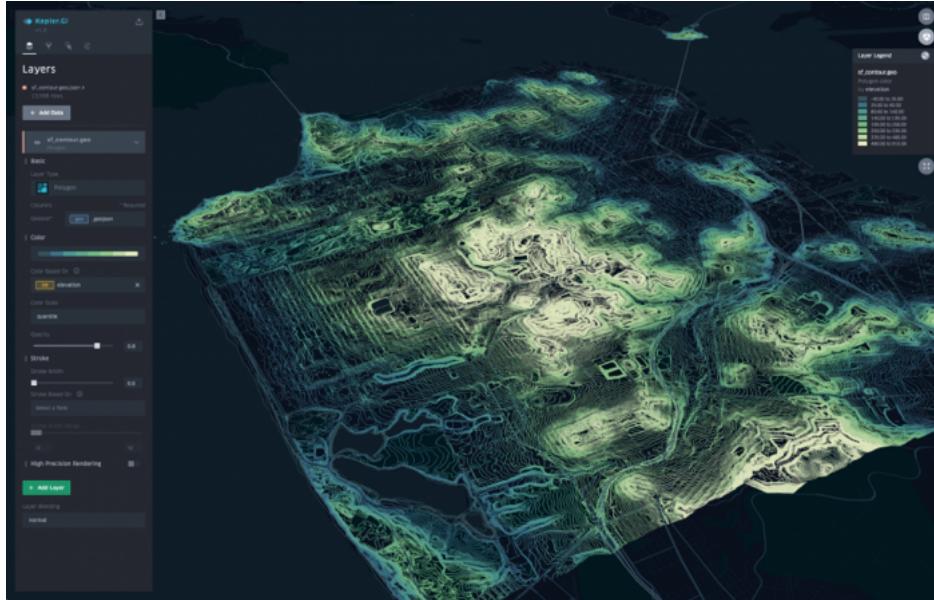


Figura 2.28: Interfaz de usuario de kepler.gl [74].

2.7.3. Google Maps

Google Maps es un servicio de mapeo web proporcionado por Google. Ofrece imágenes satelitales, fotografías aéreas, mapas de calles, vistas panorámicas de calles (Street View), condiciones de tráfico en tiempo real, y planificación de rutas para viajar a pie, en coche, bicicleta y transporte público. Desde su lanzamiento en 2005, Google Maps se ha convertido en una de las herramientas más utilizadas y avanzadas en su categoría, facilitando la exploración y navegación por todo el mundo a millones de usuarios [75]. En la Figura 2.29 se muestra la interfaz de usuario de Google Maps.

Dentro de los servicios que ofrece, los más relevantes para el desarrollo de este proyecto son:

API de Mapas

La API de Google Maps permite a los desarrolladores integrar los servicios de mapeo de Google en sus propias aplicaciones y sitios web. Esta API ofrece una amplia gama de funcionalidades, como la visualización de mapas personalizados, la superposición de datos, y la manipulación de elementos del mapa. Los desarrolladores pueden utilizar esta API para mostrar mapas

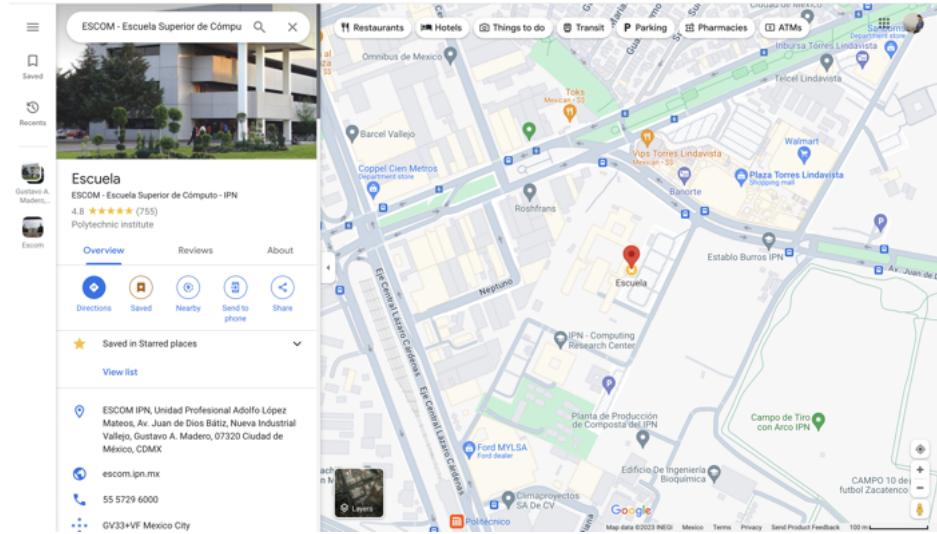


Figura 2.29: Interfaz de usuario de Google Maps.

interactivos y dinámicos, con soporte para diferentes tipos de visualización como mapas de carreteras, imágenes satelitales y terrenos. También es posible personalizar los mapas para que se ajusten al estilo y necesidades específicas de la aplicación o sitio web [75].

API de Reverse Geocoding

La API de Reverse Geocoding de Google Maps es otra herramienta poderosa para los desarrolladores. Permite convertir coordenadas geográficas (latitud y longitud) en direcciones postales o nombres de lugares. Esta funcionalidad es particularmente útil para aplicaciones que necesitan determinar la ubicación física exacta a partir de coordenadas GPS. La API proporciona respuestas detalladas, incluyendo el nombre del lugar, la dirección, el barrio y otros datos relevantes. Esto es ampliamente utilizado en aplicaciones de logística, viajes, y servicios basados en ubicación para mejorar la experiencia del usuario al proporcionar información contextual sobre su ubicación actual [76].

Capítulo 3

Análisis

3.1. Problemática

La industria inmobiliaria enfrenta grandes desafíos en la actualidad. La transformación digital dentro de la industria ha sido lenta, y para los participantes del mercado, la tecnología disponible no ha sido suficiente para resolver los problemas que enfrentan. La industria inmobiliaria es una de las más grandes del mundo, y sin embargo, es una de las que menos ha invertido en tecnología [3].

Para agentes inmobiliarios, compradores, inversionistas y desarrolladores, la falta de información repercuten en el dinamismo del mercado y limitan la toma de decisiones. A pesar de que existen plataformas que ofrecen información sobre el mercado inmobiliario, la información que ofrecen es limitada y no permite a los participantes del mercado tomar decisiones informadas.

3.2. Propuesta de Solución

La propuesta de solución consiste en desarrollar una plataforma que permita a los participantes del mercado inmobiliario tomar decisiones informadas, mediante el análisis de datos de bienes raíces con la finalidad de eficientar el proceso de estimación de valor de mercado para departamentos en la Ciudad de México, basado en sus características y ubicación.

Con la finalidad de poder obtener estimaciones de valor de mercado válidas, se hará uso de técnicas de inteligencia artificial para el entrenamiento de un modelo de aprendizaje automático que permita predecir el valor de

mercado de departamentos en la Ciudad de México según sus características y ubicación.

A fin de determinar el funcionamiento de la plataforma, se realizará un análisis exploratorio de datos para determinar la calidad de los datos disponibles para el entrenamiento del modelo, así como para determinar las características más relevantes para la predicción del valor de mercado de departamentos en la Ciudad de México.

3.2.1. ¿Por qué se eligió emplear Inteligencia Artificial?

Uno de los aspectos determinantes para realizar valuaciones inmobiliarias válidas es la precisión con la que se analiza cada inmueble, ya que cada uno de ellos es único y tiene características que lo hacen diferente a los demás. Por lo tanto, es necesario contar con un modelo que permita analizar cada inmueble de manera individual, con el fin de poder obtener estimaciones de valor de mercado válidas.

Analizar cada inmueble de forma individual es una tarea que requiere de mucho tiempo y esfuerzo, por lo que, empleando las características más distintivas e impactantes en el valor de mercado de un inmueble, se puede desarrollar un modelo que permita estimar el valor de mercado de un inmueble de manera rápida y eficiente.

En contraste con las estrategias tradicionales de valuación inmobiliaria, las cuales se basan en la comparación de inmuebles similares, el modelo propuesto se basa en el análisis de las características más distintivas, por lo que los resultados obtenidos son más precisos y confiables.

Por otro lado, ya que el flujo de información para el entrenamiento del modelo puede retomarse en distintos puntos del tiempo, es posible realizar actualizaciones a dicho modelo para que este pueda adaptarse a los cambios en el mercado inmobiliario.

3.2.2. ¿Por qué se eligió desarrollar una plataforma web?

Con el objetivo de disponibilizar las conclusiones derivadas del análisis de los datos y el modelo de aprendizaje automático, se eligió desarrollar un prototipo de plataforma web que permita a los participantes del mercado inmobiliario obtener estimaciones de valor de mercado válidas para departamentos en la Ciudad de México ya que es un medio de fácil acceso y que permite la interacción con los usuarios de manera sencilla.

La naturaleza del mercado inmobiliario obliga a sus participantes a mantenerse en movimiento para poder realizar su trabajo, por lo que es necesario que la plataforma sea accesible desde cualquier dispositivo con acceso a internet.

3.3. Herramientas a Emplear

Para desarrollar la presente propuesta de plataforma se requerirá el uso tanto de herramientas de software como de hardware. En cuestiones de software, se considerarán tanto las herramientas destinadas al desarrollo de la plataforma como las herramientas destinadas al desarrollo del modelo de aprendizaje, la extracción de datos y la limpieza de los mismos. En cuestiones de hardware, se considerarán los equipos con los que se disponen para realizar el desarrollo del proyecto.

3.3.1. Herramientas de Software

En esta sección se hará mención de las herramientas de software, lenguajes y librerías más importantes que se emplearán para el desarrollo de la plataforma.

Python

Python es un lenguaje de programación *multi-paradigma* que brinda completo soporte a la programación orientada a objetos y a la programación estructurada, además de contar con diversas características que apoyan la programación funcional y la programación orientada a aspectos, incluyendo metaprogramación y *métodos mágicos*. Python utiliza *tipado dinámico* y una combinación de *conteo de referencias* junto con un *recolector de basura detector de ciclos* para la gestión de memoria. Una característica importante de Python es la *resolución dinámica de nombres (late binding)*, que vincula nombres de métodos y variables durante la ejecución del programa. A pesar de ofrecer solo un soporte limitado para la programación funcional al estilo de Lisp, el lenguaje incluye funciones como `map()`, `reduce()` y `filter()`, comprensiones para listas, diccionarios y conjuntos, así como expresiones generadoras. Su filosofía se centra en la simplicidad y la legibilidad, rechazando la sintaxis exuberante a favor de una gramática más clara y menos recargada. Los desarrolladores de Python buscan evitar la *optimización prematura* y rechazan cambios en partes no críticas de CPython que ofrezcan un aumento marginal en velocidad a costa de la claridad. Un objetivo importante de los desarrolladores de Python es hacer que el lenguaje sea *divertido de usar*, lo

que se refleja en el origen del nombre y en un enfoque ocasionalmente lúdico hacia tutoriales y materiales de referencia [77].

Algunas ventajas de utilizar Python son:

- Es un lenguaje de programación de alto nivel, lo que permite escribir programas más cortos y fáciles de leer.
- Es un lenguaje de programación interpretado, lo que permite ejecutar programas sin necesidad de compilarlos.
- Es un lenguaje de programación multiparadigma, lo que permite utilizar distintos estilos de programación.
- Es un lenguaje de programación multiplataforma, lo que permite ejecutar programas en distintos sistemas operativos.
- Es un lenguaje de programación orientado a objetos, lo que permite reutilizar código y crear programas modulares.
- Es un lenguaje de programación extensible, lo que permite agregar funcionalidades mediante el uso de módulos.

Por otro lado, algunas desventajas de utilizar Python son:

- Es un lenguaje de programación interpretado, lo que hace que los programas sean más lentos que los programas escritos en lenguajes compilados.
- Es un lenguaje de programación de alto nivel, lo que hace que los programas sean más lentos que los programas escritos en lenguajes de bajo nivel.

Pandas

pandas es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas diseñadas para facilitar y hacer intuitivo el trabajo con datos “relacionales” o “etiquetados”. Su objetivo es ser el bloque de construcción fundamental de alto nivel para realizar análisis de datos prácticos y reales en Python. Además, tiene el objetivo más amplio de convertirse en la herramienta de análisis y manipulación de datos de código abierto más poderosa y flexible disponible en cualquier lenguaje. Ya está bien encaminado hacia este objetivo [78].

pandas es adecuado para muchos tipos diferentes de datos:

- Datos tabulares con columnas de tipos heterogéneos, como en una tabla SQL o una hoja de cálculo Excel.
- Datos de series temporales ordenados y no ordenados (no necesariamente de frecuencia fija).
- Datos de matriz arbitrarios (tipados homogéneamente o heterogéneos) con etiquetas de filas y columnas.
- Cualquier otra forma de conjuntos de datos observacionales/estadísticos. Los datos no necesitan estar etiquetados para ser colocados en una estructura de datos de pandas.

Se ha elegido **pandas** debido a que es una herramienta de análisis de datos de alto rendimiento y fácil de usar, la cual permite realizar análisis exploratorios de datos de manera sencilla.

NumPy

NumPy es el paquete fundamental para la computación científica en Python. Es una biblioteca de Python que proporciona un objeto de arreglo multidimensional, varios objetos derivados (como arreglos enmascarados y matrices) y una variedad de rutinas para operaciones rápidas en arreglos, incluyendo operaciones matemáticas, lógicas, manipulación de formas, ordenamiento, selección, E/S, transformadas de Fourier discretas, álgebra lineal básica, operaciones estadísticas básicas, simulación aleatoria y mucho más [79].

Se ha elegido **NumPy** debido a que ofrece una gran cantidad de funciones matemáticas que resultan útiles (y necesarias) para la realización de un análisis exploratorio de datos válido.

StatsModels

Statsmodels es una biblioteca para análisis estadístico y econométrico en Python, dirigida tanto a estadísticos y economistas teóricos y aplicados, como a usuarios y desarrolladores de Python de distintas disciplinas que utilizan modelos estadísticos. Es especialmente útil para aquellos usuarios de R, Stata, SAS, SPSS, NLOGIT, GAUSS o MATLAB interesados en trabajar en Python por sus beneficios. Inicialmente parte de SciPy como **models**, luego se integró en el proyecto de neuroimagen NIPY para su maduración y evolución. Mejorado durante el Google Summer of Code 2009 y 2010, ahora se distribuye como un SciKit, o paquete adicional para SciPy. Sus principales desarrolladores, formados en economía y con experiencia en econometría, han enfocado su desarrollo en aplicaciones econométricas, aunque su diseño

busca ser amigable y extensible por desarrolladores de cualquier disciplina. El trabajo continuo está ampliando su utilidad para necesidades comunes de modelado estadístico [80].

Esta librería se ha elegido ya que es idónea para realizar el análisis exploratorio econométrico con el cual determinaremos los factores más relevantes para la predicción del valor de mercado de departamentos en la Ciudad de México, así como establecer los criterios apropiados para la elección del modelo de aprendizaje automático.

Scikit-Learn

Scikit-learn es un módulo de Python que integra una amplia gama de algoritmos de aprendizaje automático de última generación para problemas supervisados y no supervisados de escala media. Este paquete se centra en llevar el aprendizaje automático a los no especialistas utilizando un lenguaje de alto nivel de propósito general. Se pone énfasis en la facilidad de uso, el rendimiento, la documentación y la consistencia de la API. Tiene dependencias mínimas y se distribuye bajo la licencia BSD simplificada, fomentando su uso tanto en entornos académicos como comerciales [28].

Se ha elegido **Scikit-Learn** debido a que es una herramienta de aprendizaje automático sencilla de usar, la cual permite entrenar modelos de aprendizaje automático de manera sencilla y, para fines de este proyecto, permite realizar análisis de componentes principales y regresión simple y múltiple.

TensorFlow

TensorFlow es un sistema de aprendizaje automático que opera a gran escala y en entornos heterogéneos. Su modelo computacional se basa en grafos de flujo de datos con estado mutable. Los nodos del grafo pueden mapearse a diferentes máquinas en un clúster, y dentro de cada máquina a CPUs, GPUs y otros dispositivos. TensorFlow admite una variedad de aplicaciones, pero se enfoca particularmente en el entrenamiento y la inferencia con redes neuronales profundas. Sirve como plataforma tanto para la investigación como para la implementación de sistemas de aprendizaje automático en muchas áreas, tales como reconocimiento de voz, visión por computadora, robótica, recuperación de información y procesamiento de lenguaje natural. En esta charla, describimos TensorFlow y esbozamos algunas de sus aplicaciones. También discutimos la cuestión de qué relación puede tener TensorFlow y el aprendizaje profundo con la programación funcional. Aunque TensorFlow no es puramente funcional, muchos de sus usos están relacionados con la optimiza-

ción de funciones (durante el entrenamiento), y luego con la aplicación de esas funciones (durante la inferencia). Estas funciones se definen como composiciones de primitivas simples (como es común en la programación funcional), con representaciones internas de datos que se aprenden en lugar de diseñarse manualmente. TensorFlow es un trabajo conjunto con muchas otras personas del equipo de Google Brain y otros lugares [30].

Se ha elegido **TensorFlow** debido a que es una herramienta de aprendizaje automático que permite entrenar modelos de aprendizaje automático de manera sencilla y, para fines de este proyecto, permite entrenar modelos de aprendizaje automático de regresión, además de integrarse directamente con FastAPI.

FastAPI

FastAPI es un moderno marco de trabajo web de API de Python de alto rendimiento, basado en OpenAPI (anteriormente conocido como Swagger) y estándares de tipo de datos estándar de Python. El marco es ideal para crear microservicios rápidos y escalables, ya que es fácil de aprender y usar, y ofrece una gran cantidad de características útiles, como la generación automática de documentación, validación de datos, pruebas unitarias y de integración, y mucho más [81].

Se ha elegido **FastAPI** debido a que es un marco de trabajo web de API que permite desarrollar APIs de manera sencilla y rápida, además de integrarse directamente con TensorFlow.

QGIS

QGIS es un sistema de información geográfica (SIG) de código abierto que comenzó en 2002 y opera en plataformas Unix, Windows y macOS. Desarrollado con Qt y C++, ofrece una interfaz de usuario amigable, además de aplicaciones para dispositivos móviles. Inicialmente un visor de datos SIG, ahora QGIS es utilizado para visualización de datos GIS, captura de datos, análisis avanzado y presentaciones en mapas. Admite numerosos formatos de datos y se expande fácilmente con complementos. Publicado bajo la Licencia Pública General GNU (GPL), QGIS es gratuito y de código fuente abierto, asegurando acceso y modificación libres [72].

Se ha elegido usar QGIS debido a que es un software de código abierto que permite realizar análisis geoespaciales de manera sencilla y rápida, además de que permite visualizar los datos de manera sencilla. En la Figura 3.1 se

muestra una captura de pantalla de QGIS con los datos de bienes raíces de la Ciudad de México.

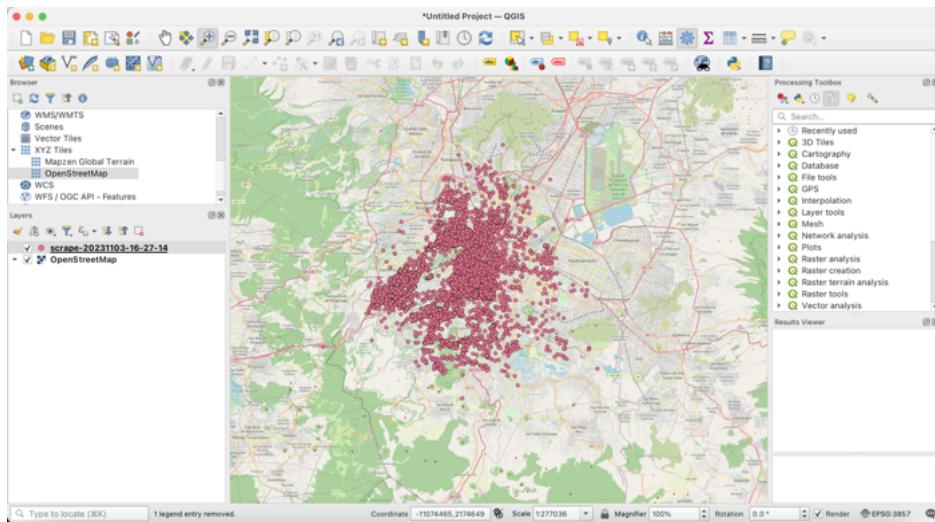


Figura 3.1: Captura de pantalla de QGIS con los datos de bienes raíces de la Ciudad de México.

React

React es un marco de trabajo de código abierto desarrollado por Facebook para crear interfaces de usuario. Se utiliza para manejar la capa de vista para aplicaciones web y móviles. Una de las principales características de React es que utiliza un enfoque basado en componentes para el desarrollo de aplicaciones. Con React, los desarrolladores pueden crear componentes reutilizables que se pueden utilizar para crear interfaces de usuario complejas. React también permite a los desarrolladores crear interfaces de usuario declarativas. Esto significa que los desarrolladores pueden escribir código que describa cómo debería ser la interfaz de usuario. React se encarga de actualizar automáticamente la interfaz de usuario cuando los datos cambian, lo que permite a los desarrolladores evitar tener que escribir código adicional para actualizar la interfaz de usuario [61].

Se eligió React debido a que es un marco de trabajo conocido por el equipo de desarrollo, además de que, al ser de código abierto y alta popularidad, goza de una gran cantidad de documentación, soporte y librerías que facilitan el desarrollo de aplicaciones web.

Material-UI

Material-UI es una biblioteca de componentes de interfaz de usuario para React que implementa el lenguaje de diseño de Google, Material Design, para la web. Material-UI hace que el desarrollo de aplicaciones web sea más rápido y fácil [60].

Se decidió trabajar con Material-UI debido a que su implementación es conocida por el equipo de desarrollo y su diseño es atractivo y moderno.

Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización de aplicaciones en múltiples sistemas operativos. Docker utiliza características de aislamiento de recursos del kernel de Linux, como cgroups y espacios de nombres (namespaces) para permitir que “contenedores” independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales (VMs) [54].

Se eligió Docker para empaquetar la aplicación web y facilitar su despliegue en cualquier servidor, en nuestro caso, en la nube.

3.3.2. Herramientas de Hardware

En tanto a hardware se refiere, para el desarrollo de este proyecto se empleará una computadora portátil con las siguientes características:

- Procesador Apple M1 de 8 núcleos, 4 de rendimiento y 4 de eficiencia.
- GPU de 8 núcleos.
- 8 GB de memoria unificada.
- 256 GB de almacenamiento SSD.
- Pantalla Retina de 13 pulgadas con tecnología True Tone.
- Touch Bar y Touch ID.
- Dos puertos Thunderbolt / USB 4.

Amazon Web Services

Debido a que se pretende que la aplicación web sea accesible desde cualquier dispositivo con acceso a internet, se consideró la posibilidad de emplear servicios de computación en la nube para el despliegue de la aplicación web. De los servicios de computación en la nube disponibles, se eligió Amazon Web Services (AWS) debido a que es el servicio de computación en la nube más popular y cuenta con una gran cantidad de documentación, soporte y librerías que facilitan el despliegue de aplicaciones web.

Aunque el entrenamiento de los modelos de aprendizaje automático se realizará en la computadora portátil mencionada anteriormente, el despliegue del sistema final se realizará en un servicio de computación en la nube.

3.4. Análisis Exploratorio de Datos

El análisis exploratorio de datos es una parte fundamental del proceso de minería de datos, ya que permite obtener información relevante sobre los datos que se utilizarán para el entrenamiento del modelo de aprendizaje automático. A su vez, permite identificar patrones, tendencias y relaciones entre los datos, lo cual permite determinar la calidad de los datos y la relevancia de las variables para el entrenamiento del modelo [82].

Debido a la naturaleza del proyecto, los datos que se utilizarán para el entrenamiento del modelo de aprendizaje automático son datos de bienes raíces georeferenciados mediante una latitud y longitud, por lo que el análisis consistirá en un análisis exploratorio geoespacial y un análisis exploratorio econométrico.

El conjunto de datos empleado para el análisis exploratorio de datos es un conjunto de datos de bienes raíces de la Ciudad de México, el cual fue obtenido de la plataforma Inmuebles24 [10]. Este conjunto de datos contiene información sobre 40,000 anuncios de bienes raíces de la Ciudad de México, los cuales fueron publicados hasta el 30 de Agosto de 2023. El conjunto de datos se obtuvo únicamente para, a partir del presente análisis exploratorio de datos, determinar la calidad de los datos y la relevancia de las variables para el entrenamiento del modelo de aprendizaje automático. Para el desarrollo de la plataforma final, se utilizará un conjunto de datos que incorpore los datos extraídos durante el desarrollo del proyecto a bien de tener un conjunto de datos más completo y actualizado.

3.4.1. Análisis Exploratorio Geoespacial

Como se mencionó, los datos tienen una componente geoespacial, por lo que es importante realizar un análisis exploratorio geoespacial para poder obtener información relevante. Dentro de este análisis se realizarán los siguientes análisis:

- Visualización de puntos de anuncios
- Mapa de densidad de anuncios por alcaldía
- Mapa de calor de precios
- Mapa de calor de número de recámaras
- Mapa de calor de número de baños
- Mapa de calor de metros cuadrados
- Mapa de calor de antigüedad
- Mapa de calor de estacionamientos

Visualización de puntos de anuncios

La visualización de puntos de anuncios permite representar individualmente cada anuncio. Esto permite realizar un análisis de distribución geográfica y relación con puntos de interés (parques, escuelas, centros comerciales, etc.). En la Figura 3.2 se muestra la visualización de puntos de anuncios para el conjunto de datos de bienes raíces.

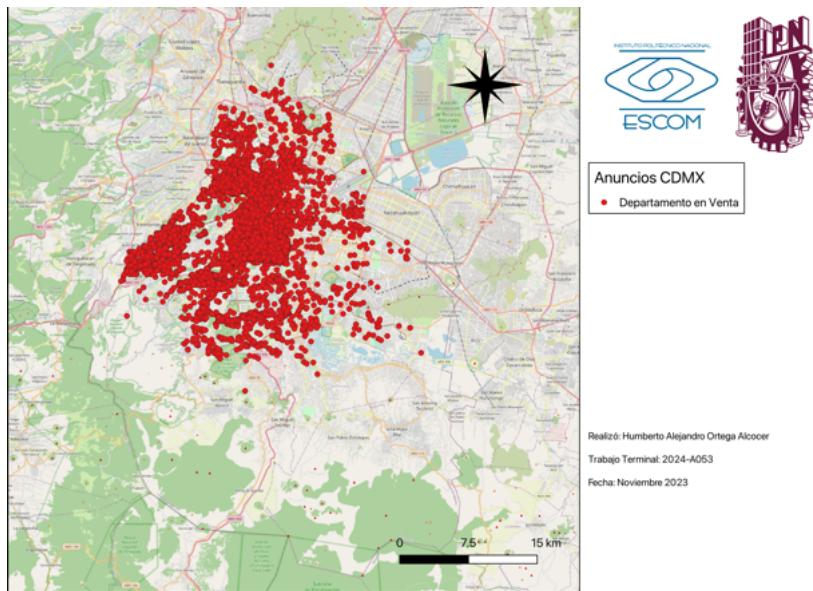


Figura 3.2: Visualización de puntos de anuncios.

De aquí, podemos derivar las siguientes conclusiones:

- Los anuncios no se encuentran distribuidos de manera uniforme en toda la Ciudad de México.
- Los anuncios se encuentran distribuidos de manera uniforme en las alcaldías Cuauhtémoc, Benito Juárez y Miguel Hidalgo.
- En las alcaldías Milpa Alta y Xochimilco existen pocos anuncios debido a que son zonas con poca densidad de población y muchos tratos aún se realizan de manera tradicional (sin el uso de plataformas online).

Mapa de densidad de anuncios por alcaldía

El mapa de densidad de anuncios permite visualizar la concentración de anuncios en las diferentes alcaldías de la CDMX. Esto permite identificar las áreas con alta y baja densidad de anuncios y posibles factores (desarrollo urbano, demanda, etc.). En la Figura 3.3 se muestra el mapa de calor de densidad de anuncios para el conjunto de datos de bienes raíces.

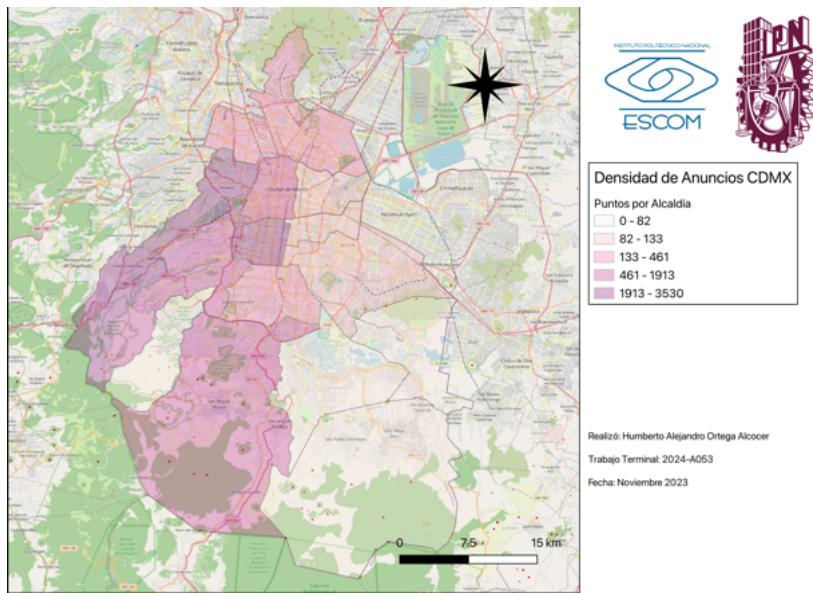


Figura 3.3: Mapa de de densidad de anuncios por alcaldía.

De aquí, podemos derivar las siguientes conclusiones:

- Las zonas con mayor densidad de anuncios se encuentran en la zona centro de la Ciudad de México, en las alcaldías Cuauhtémoc y Miguel Hidalgo.
- Las zonas con menor densidad de anuncios se encuentran en la zona sur de la Ciudad de México, en las alcaldías Tlalpan, Xochimilco y Milpa Alta.

Los datos correspondientes al mapa de la Figura 3.3 se muestran en el Cuadro 3.1.

Alcaldía	Cantidad de Anuncios
Benito Juárez	3530
Miguel Hidalgo	3114
Cuajimalpa de Morelos	2246
Cuauhtémoc	1913
Álvaro Obregón	1711
Tlalpan	483
Coyoacán	461
Azcapotzalco	301
Gustavo A. Madero	191
Iztacalco	133
Venustiano Carranza	127
Iztapalapa	117
La Magdalena Contreras	82
Tláhuac	29
Xochimilco	16
Milpa Alta	0

Cuadro 3.1: Datos correspondientes al número de anuncios encontrados por alcaldía.

Mapa de calor de precios

El mapa de calor de precios permite visualizar cómo se distribuyen los precios en diferentes áreas. Esto permite identificar las zonas con precios más altos o más bajos y posibles razones (ubicación, accesibilidad, etc.). En la Figura 3.4 se muestra el mapa de calor de precios para el conjunto de datos de bienes raíces.

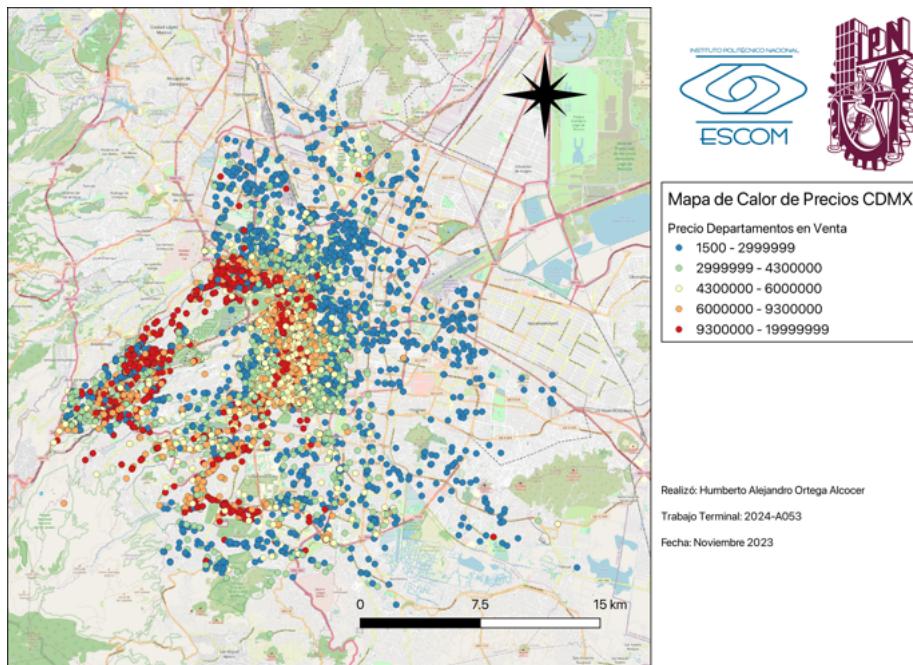


Figura 3.4: Mapa de calor de precios.

De aquí, podemos derivar las siguientes conclusiones:

- Los precios más altos se encuentran en la zona centro de la Ciudad de México, en las alcaldías Cuauhtémoc y Miguel Hidalgo.
- Los precios más bajos se encuentran en la zona sur de la Ciudad de México, en las alcaldías Tlalpan, Xochimilco y Milpa Alta.
- Los precios más altos se encuentran en las zonas con mayor densidad de vegetación, como el Bosque de Chapultepec, el Parque Ecológico de Xochimilco y el Parque Ecológico de la Ciudad de México.

Mapa de calor de número de recámaras

El mapa de calor de número de recámaras permite visualizar cómo se distribuyen los anuncios según el número de recámaras. Esto permite identificar las zonas con mayor y menor número de recámaras y analizar si el precio es un factor determinante en el número de recámaras. En la Figura 3.5 se muestra el mapa de calor de número de recámaras para el conjunto de datos de bienes raíces.

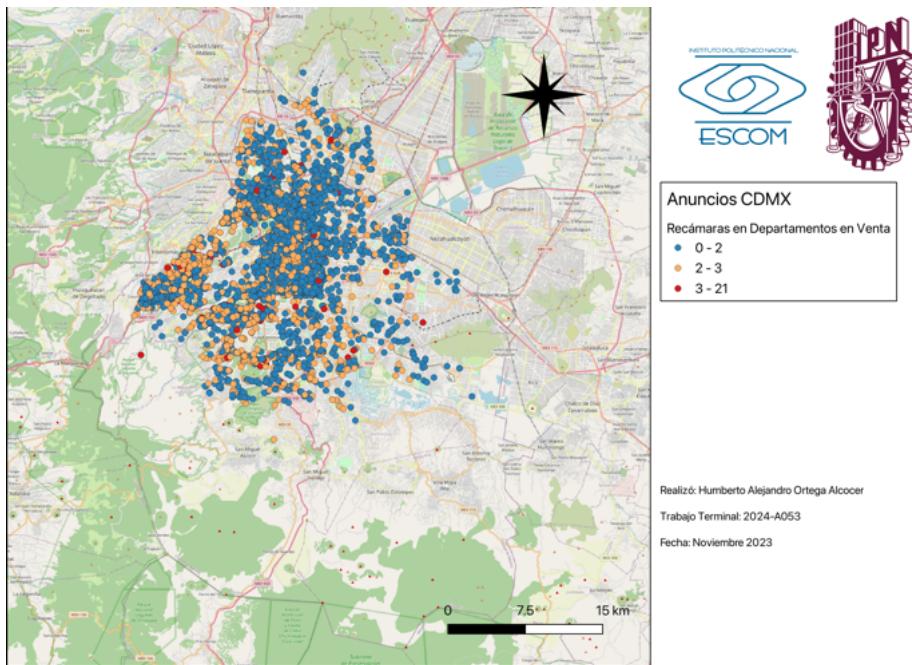


Figura 3.5: Mapa de calor de número de recámaras.

De aquí, podemos derivar las siguientes conclusiones:

- La gran mayoría de los departamentos en la Ciudad de México tienen 2 o 3 recámaras.
- Los departamentos con 1 recámara se encuentran en las zonas con mayor densidad de población, como la zona centro de la Ciudad de México.
- Los departamentos con 4 o más recámaras se encuentran en las zonas con mayor precio, como la zona poniente de la Ciudad de México.

Mapa de calor de número de baños

Los baños nos indican la disponibilidad de servicios sanitarios en el inmueble, así como un aproximado del número de personas que podrían habitar el mismo. En la Figura 3.6 se muestra el mapa de calor de número de baños para el conjunto de datos de bienes raíces.

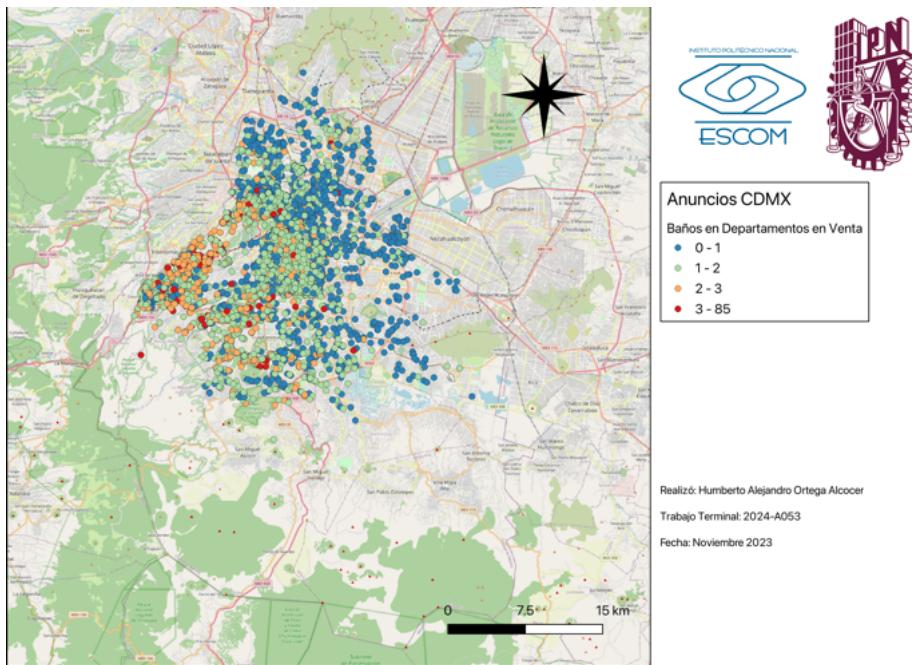


Figura 3.6: Mapa de calor de número de baños.

De aquí, podemos derivar las siguientes conclusiones:

- La gran mayoría de los departamentos en la Ciudad de México tienen 1 o 2 baños.
- Los departamentos con 1 baño se encuentran en las zonas con mayor densidad de población, como la zona centro de la Ciudad de México.
- Los departamentos con 3 o más baños se encuentran en las zonas con mayor precio, como la zona poniente de la Ciudad de México.

Mapa de calor de metros cuadrados

La distribución de las superficies de departamentos en la Ciudad de México nos permiten obtener un vistazo sobre la relación directa en el precio estimado por metro cuadrado por zona, un factor determinante en el mercado inmobiliario. En la Figura 3.7 se muestra el mapa de calor de metros cuadrados para el conjunto de datos de bienes raíces.

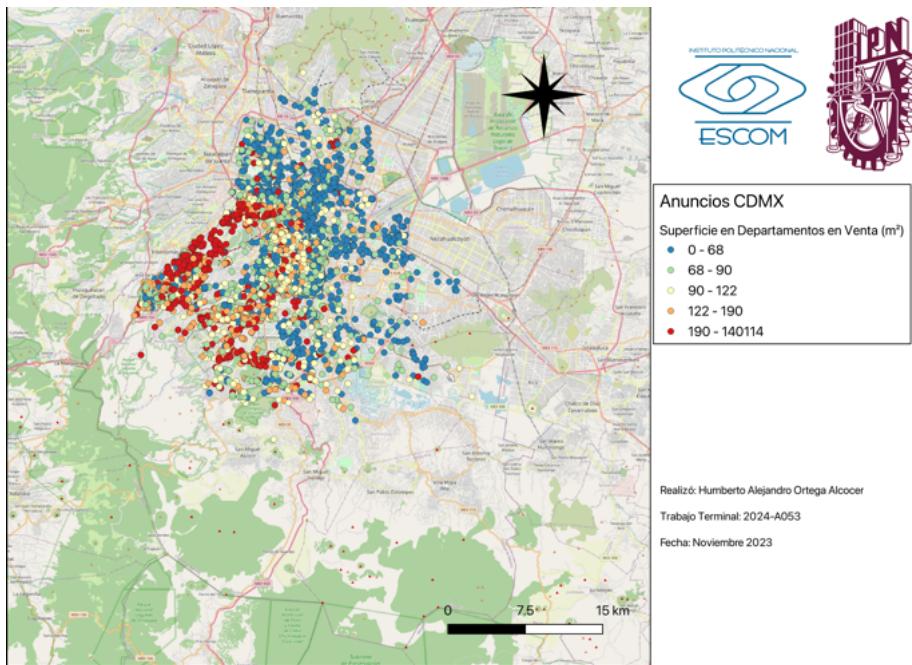


Figura 3.7: Mapa de calor de metros cuadrados.

De aquí, podemos derivar las siguientes conclusiones:

- La gran mayoría de los departamentos en la Ciudad de México tienen entre 50 y 100 metros cuadrados.
- Los departamentos con menos de 50 metros cuadrados se encuentran en las zonas con mayor densidad de población, como la zona centro de la Ciudad de México. A su vez, estos departamentos tienen un precio por metro cuadrado más alto.
- Los departamentos con más de 100 metros cuadrados se encuentran en las zonas con mayor precio, como la zona poniente de la Ciudad de México.

Mapa de calor de antigüedad

La antigüedad de los departamentos es un factor determinante en el precio de los mismos, ya que los departamentos más antiguos tienden a tener un precio más bajo que los departamentos nuevos. Por otra parte, aunque no es el único factor a considerar, al ser una zona sísmica, la antigüedad de los departamentos es un factor importante a considerar para la seguridad de los

habitantes. En la Figura 3.8 se muestra el mapa de calor de antigüedad para el conjunto de datos de bienes raíces.

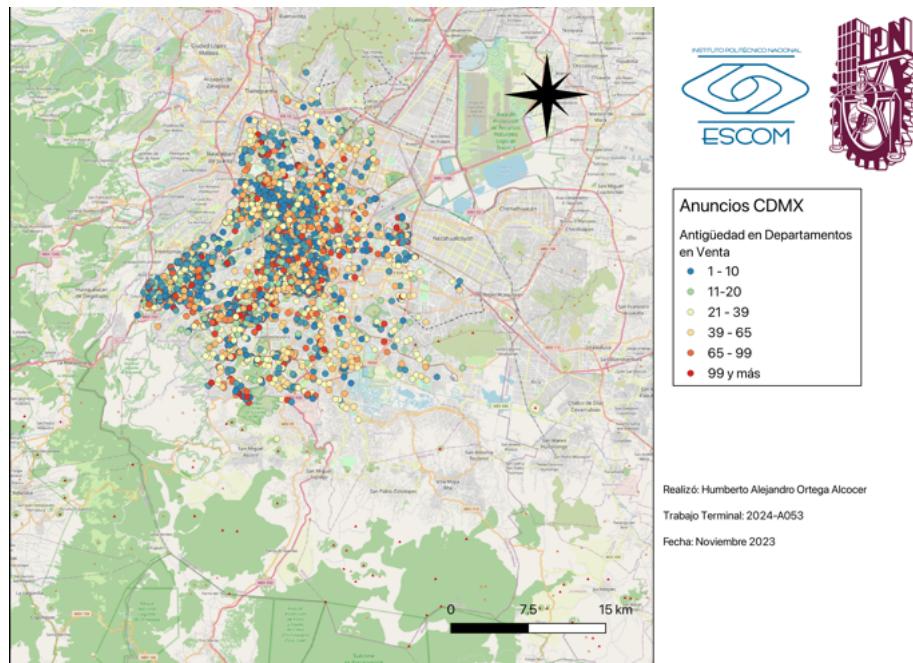


Figura 3.8: Mapa de calor de antigüedad.

De aquí, podemos derivar las siguientes conclusiones:

- La gran mayoría de los departamentos en la Ciudad de México tienen entre 0 y 20 años de antigüedad.
- Los departamentos con menos de 20 años de antigüedad se encuentran en las zonas con mayor densidad de población, como la zona centro de la Ciudad de México. A su vez, estos departamentos tienen un precio más alto.
- Los departamentos con más de 20 años de antigüedad se encuentran en las zonas con mayor precio, como la zona poniente de la Ciudad de México.

Mapa de calor por número de estacionamientos

Los estacionamientos son un factor muy importante en la Ciudad de México, ya que una gran cantidad de personas utilizan automóvil para transportarse.

En la Figura 3.9 se muestra el mapa de calor de número de estacionamientos para el conjunto de datos de bienes raíces.

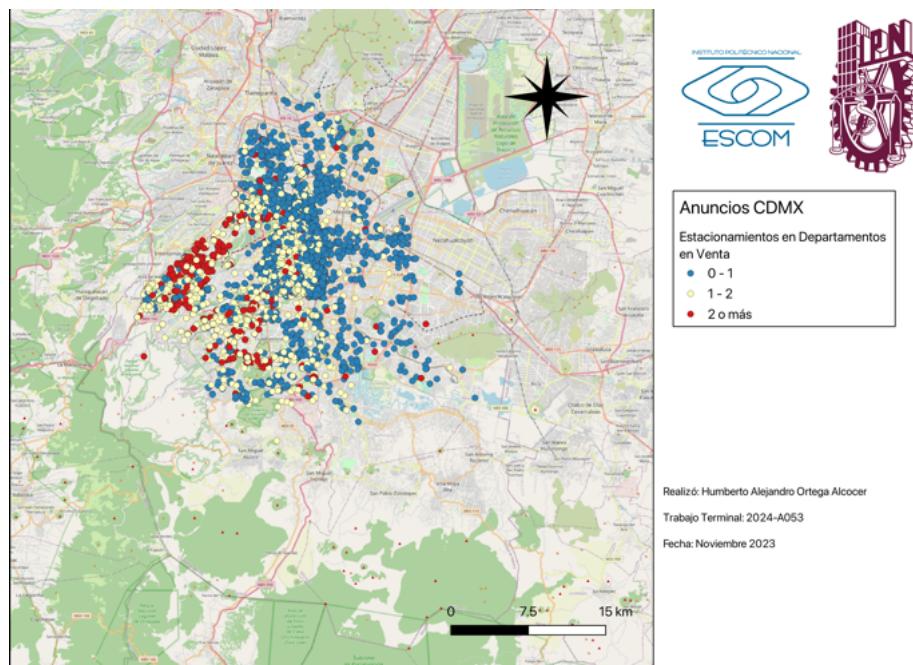


Figura 3.9: Mapa de calor de número de estacionamientos.

De aquí, podemos derivar las siguientes conclusiones:

- La gran mayoría de los departamentos en la Ciudad de México tienen 1 o 2 estacionamientos.
- Los departamentos con 1 estacionamiento se encuentran en las zonas con mayor densidad de población, como la zona centro de la Ciudad de México.
- Los departamentos con 3 o más estacionamientos se encuentran en las zonas con mayor precio, como la zona poniente de la Ciudad de México.

3.4.2. Análisis Exploratorio Econométrico

La econometría es un campo de estudio que utiliza métodos estadísticos y matemáticos para analizar y comprender las relaciones económicas, ayudando a distinguir entre buenas y malas ideas en economía, negocios y políticas públicas. Se enfoca en proporcionar respuestas cuantitativas a preguntas importantes en estos ámbitos, permitiendo vislumbrar cómo individuos, empresas

y gobiernos toman decisiones basadas en dichas relaciones. La econometría busca hacer la teoría inmediatamente relevante para las aplicaciones prácticas, integrando la teoría y las aplicaciones para hacer el aprendizaje más atractivo y pertinente [83].

El análisis exploratorio econométrico consiste en caracterizar, mediante el uso de estadísticas descriptivas, las variables que se utilizarán para el entrenamiento del modelo de aprendizaje automático.

- Análisis descriptivo
- Correlación entre variables
- Análisis de regresión simple y múltiple
- Análisis de componentes principales (PCA)

Análisis Descriptivo

Un análisis descriptivo nos permite caracterizar las variables que se utilizarán para el entrenamiento del modelo de aprendizaje automático. Además, nos permite obtener información rápida sobre nuestros datos a bien de poder determinar si los datos son adecuados para el entrenamiento del modelo.

Para realizar un análisis descriptivo, se utilizó la librería Pandas de Python, la cual permite realizar análisis descriptivos de manera sencilla. En el Código 3.1 se muestra el código utilizado para realizar el análisis descriptivo.

Listing 3.1: Análisis descriptivo.

```
1 import pandas as pd
2
3 # Carga de datos
4 df = pd.read_csv('scrape-20231103-16-27-14.csv')
5
6 # Columnas a analizar
7 numeric_columns = ['Price', 'Size_Terrain', 'Size_Construction',
8                     'Rooms', 'Bathrooms', 'Parking', 'Age']
9
10 # Seleccionar solo las columnas numéricas
11 df_numeric = df[numeric_columns]
12
13 # Análisis descriptivo
14 resultado = df_numeric.describe(include='all')
15
16 # Imprimir resultado
17 print(resultado)
```

En el Cuadro 3.2 se muestran los resultados del análisis descriptivo.

De aquí, podemos derivar las siguientes conclusiones:

Cuadro 3.2: Estadísticas Descriptivas de los Listados de Bienes Raíces

	Precio (MDP)	Superficie	Superficie Cons.	Recámaras	Baños	Estacionamiento	Antigüedad
count	17,88	17 740,00	17 694,00	17 537,00	17 478,00	16 269,00	10 758,00
mean	12,18	181,00	156,00	2,00	2,00	2,00	15,00
std	183,45	1601,00	426,00	1,00	1,00	1,00	15,00
min	0,00	1,00	1,00	1,00	1,00	1,00	1,00
25 %	3,58	75,00	75,00	2,00	2,00	1,00	5,00
50 %	5,65	109,00	107,00	2,00	2,00	2,00	10,00
75 %	9,95	180,00	179,00	3,00	2,00	2,00	20,00
max	175,00	1401,00	3301,00	6,00	5,00	8,00	310,00

- El precio promedio de las propiedades es de 12.18 millones de pesos, con una amplia variación indicada por una desviación estándar de 183.45 millones de pesos.
- La superficie del terreno varía significativamente entre las propiedades, con un promedio de 181 m^2 y una desviación estándar alta de 1601 m^2 .
- La superficie de construcción promedio es de 155 m^2 , con propiedades que van desde 1 m^2 hasta 33015 m^2 .
- El número promedio de habitaciones por propiedad es de aproximadamente 2.34, con la mayoría de las propiedades teniendo entre 2 y 3 habitaciones.
- El número de baños y espacios de estacionamiento también varía, con un promedio cercano a 2 para cada uno.
- La edad media de las propiedades es de aproximadamente 14.78 años, con algunas propiedades teniendo hasta 310 años de antigüedad.
- Los mínimos en varias categorías (como 1 m^2 de superficie de construcción) podrían indicar datos atípicos o errores en los datos.

Correlación entre variables

La correlación entre variables nos permite determinar si existe una relación entre las variables que se utilizarán para el entrenamiento del modelo de aprendizaje automático. Este análisis es importante ya que nos permite determinar los pesos que podrían tener las variables en el entrenamiento del modelo. Por otra parte, entender qué variables tienen más impacto en otras nos permite determinar si es necesario realizar un análisis de regresión simple o múltiple.

Para realizar un análisis de correlación entre variables, se utilizó la librería Pandas de Python, la cual permite realizar análisis de correlación entre variables de manera sencilla. En el Código 3.2 se muestra el código utilizado

para realizar el análisis de correlación entre variables.

Listing 3.2: Correlación entre variables.

```

1 import pandas as pd
2
3 # Carga de datos
4 df = pd.read_csv('scrape-20231103-16-27-14.csv')
5
6 # Columnas a analizar
7 numeric_columns = ['Price', 'Size_Terrain', 'Size_Construction',
8                     'Rooms', 'Bathrooms', 'Parking', 'Age']
9
10 # Seleccionar solo columnas numéricas
11 df_numeric = df[numeric_columns]
12
13 # Correlación de Pearson
14 resultado = df_numeric.corr(method='pearson')
15
16 # Imprimir resultado
17 print(resultado)

```

En el Cuadro 3.3 se muestran los resultados del análisis de correlación entre variables.

Cuadro 3.3: Matriz de Correlación entre Variables

	Precio	Superficie	Superficie Cons.	Recámaras	Baños	Estacionamiento	Antigüedad
Precio	1,00	0,00	0,01	0,02	0,02	0,02	0,01
Superficie	0,00	1,00	0,27	0,04	0,04	0,06	-0,01
Superficie Cons.	0,01	0,27	1,00	0,15	0,14	0,21	0,01
Recámaras	0,02	0,04	0,15	1,00	0,47	0,46	0,13
Baños	0,02	0,04	0,14	0,47	1,00	0,46	-0,05
Estacionamiento	0,02	0,06	0,21	0,46	0,46	1,00	-0,10
Antigüedad	0,01	-0,01	0,01	0,13	-0,05	-0,10	1,00

De aquí, podemos derivar las siguientes conclusiones:

- **Relaciones Débiles:** Los coeficientes de correlación entre el precio y las demás variables son muy bajos (todos inferiores a 0.03), lo que sugiere que no hay una relación lineal fuerte entre el precio y las otras características de las propiedades en el conjunto de datos.
- **Superficie y Construcción:** La correlación más fuerte observada es entre la superficie del terreno y la superficie construida (0.27), lo cual es esperable ya que propiedades con terrenos más grandes tienden a tener mayores construcciones.
- **Recámaras y Baños/Estacionamiento:** Hay una correlación moderada entre el número de recámaras y baños (0.47) y recámaras y estacionamientos (0.46), lo que indica que propiedades con más recámaras tienden a tener más baños y más espacio de estacionamiento.

- **Edad de la Propiedad:** La edad tiene una correlación negativa con el estacionamiento y los baños, aunque es débil. Esto podría sugerir que las propiedades más antiguas tienen menos baños y menos estacionamientos, pero es una relación muy leve.
- **Relevancia para IA:** Para la valuación inmobiliaria con IA utilizando redes neuronales, estos resultados implican que las variables incluidas no proporcionan una indicación clara del precio de manera individual. Por lo tanto, el modelo de red neuronal deberá capturar relaciones no lineales complejas o interacciones entre variables que no son evidentes a partir de los coeficientes de correlación lineal.

Análisis de regresión simple y múltiple

Con el fin de poder determinar la relación entre las variables que se utilizarán para el entrenamiento del modelo de aprendizaje automático, se realizó un análisis de regresión simple y múltiple. Las regresiones simples y múltiples son técnicas estadísticas que permiten determinar la relación entre una variable dependiente y una o más variables independientes. En el caso de la regresión simple, se determina la relación entre una variable dependiente y una variable independiente, mientras que en el caso de la regresión múltiple, se determina la relación entre una variable dependiente y dos o más variables independientes [83].

Para realizar un análisis de regresión simple y múltiple, se utilizó la librería StatsModels de Python, la cual permite realizar análisis de regresión simple y múltiple de manera sencilla. En el Código 3.3 se muestra el código utilizado para realizar el análisis de regresión simple y múltiple.

Listing 3.3: Regresión simple y múltiple.

```

1 import statsmodels.api as sm
2 import pandas as pd
3 import numpy as np
4
5 # Carga de datos
6 df = pd.read_csv('scrape-20231103-16-27-14.csv')
7
8 # Reemplazar infinitos con NaN
9 df.replace([np.inf, -np.inf], np.nan, inplace=True)
10
11 # Eliminar filas con valores NaN
12 df_clean = df.dropna(subset=['Price', 'Size_Terrain', 'Size_Construction',
13                       'Rooms', 'Bathrooms', 'Parking'])
14
15 # Columnas a analizar
16 numeric_columns = ['Price', 'Size_Terrain', 'Size_Construction',
17                     'Rooms', 'Bathrooms', 'Parking', 'Age']
18
19 # Seleccionar solamente columnas numéricas

```

```

20 df_numeric = df_clean[numeric_columns]
21
22 # Regresión simple usando Size_Construction como predictor del Price
23 X_simple = sm.add_constant(df_numeric['Size_Construction']) # Añadir constante
24 modelo_simple = sm.OLS(df_numeric['Price'], X_simple).fit()
25
26 # Regresión múltiple usando Size_Construction, Rooms, Bathrooms y Parking como predictores del Price
27 X_multiple = sm.add_constant(df_numeric[['Size_Terrain', 'Rooms', 'Bathrooms', 'Parking']]) # Añadir constante
28 modelo_multiple = sm.OLS(df_numeric['Price'], X_multiple).fit()
29
30 # Imprimir los resultados
31 print("Regresión simple")
32 print(modelo_simple.summary())
33 print()
34 print("Regresión múltiple")
35 print(modelo_multiple.summary())

```

En el Cuadro 3.4 se muestran los resultados del análisis de regresión simple.

Cuadro 3.4: Resultados de la Regresión Simple

Variable	Coeficiente	Error Estándar
Constante	1,21	1,69
Size_Construction	4917,3102	4216,805

Nota: R-cuadrado: 0.000; F-estadístico: 1.360, P-valor: 0.244. El Omnibus y Jarque-Bera indican una distribución no normal de los residuos.

En el Cuadro 3.5 se muestran los resultados del análisis de regresión múltiple.

Cuadro 3.5: Resultados de la Regresión Múltiple

Variable	Coeficiente	Error Estándar
Constante	-6,191	5,63
Size_Terrain	76,1953	925,533
Rooms	1,772	2,6
Bathrooms	1,304	1,8
Parking	3,122	2,03

Nota: R-cuadrado: 0.001; F-estadístico: 2.021, P-valor: 0.0887. Las notas adicionales mencionan errores estándar que asumen que la matriz de covarianza de los errores está correctamente especificada y un número de condición grande, lo que podría indicar multicolinealidad.

De los resultados obtenidos, podemos derivar las siguientes conclusiones:

- Para la regresión simple, el R-cuadrado es 0, lo que indica que el modelo no explica nada de la variabilidad del precio. Esto está reforzado por el alto P-valor, sugiriendo que la variable Size_Construction no es un predictor estadísticamente significativo del precio.
- Para la regresión múltiple, aunque incluye más variables, el R-cuadrado sigue siendo muy bajo (0.001), indicando que el modelo explica menos del 0.1 % de la variabilidad del precio. Los P-valores también son altos, sugiriendo que ninguna de las variables independientes es un predictor significativo del precio.
- Las altas estadísticas de Omnibus y Jarque-Bera junto con la significativa desviación (skewness) y la kurtosis apuntan a una distribución no normal de los residuos, lo que podría afectar la fiabilidad de los tests estadísticos.
- El número de condición alto en la regresión múltiple sugiere que podría haber problemas de multicolinealidad en el modelo, lo que significa que algunas de las variables independientes están correlacionadas entre sí.

Análisis de componentes principales (PCA)

El análisis de componentes principales es un método estadístico versátil para reducir una tabla de datos de casos por variables a sus características esenciales, llamadas componentes principales. Los componentes principales son unas pocas combinaciones lineales de las variables originales que explican de manera máxima la varianza de todas las variables. En el proceso, el método proporciona una aproximación de la tabla de datos original utilizando solo estos pocos componentes principales [84].

Para poder llevar a cabo este análisis, se utilizó la librería Scikit-Learn de Python, la cual permite realizar análisis de componentes principales de manera sencilla. En el Código 3.4 se muestra el código utilizado para realizar el análisis de componentes principales.

Listing 3.4: Análisis de componentes principales (PCA).

```

1 import pandas as pd
2 import numpy as np
3 from sklearn.decomposition import PCA
4 from sklearn.preprocessing import StandardScaler
5
6 # Carga de datos
7 df = pd.read_csv('scrape-20231103-16-27-14.csv')
8
9 # Reemplazar infinitos con NaN
10 df.replace([np.inf, -np.inf], np.nan, inplace=True)
11

```

```

12 # Eliminar filas con valores NaN
13 df_clean = df.dropna(subset=['Price', 'Size_Terrain', 'Size_Construction',
14                      'Rooms', 'Bathrooms', 'Parking', 'Age'])
15
16 # Columnas a analizar
17 numeric_columns = ['Price', 'Size_Terrain', 'Size_Construction',
18                      'Rooms', 'Bathrooms', 'Parking', 'Age']
19
20 # Seleccionar únicamente las columnas numéricas
21 df_numeric = df_clean[numeric_columns]
22
23 # Estandarizar los datos
24 scaler = StandardScaler()
25 df_scaled = scaler.fit_transform(df_numeric)
26
27 # PCA
28 pca = PCA(n_components=2) # Reducir a 2 componentes principales
29
30 componentes_principales = pca.fit_transform(df_scaled)
31
32 # Imprimir los componentes principales
33 print('Componentes principales')
34 print(componentes_principales)
35
36 # Varianza explicada
37 print('Varianza explicada')
38 print(pca.explained_variance_ratio_)
39
40 # Coeficientes de los componentes
41 print('Coeficientes de los componentes')
42 print(pca.components_)

```

Los resultados del análisis de componentes principales se muestran en el Cuadro 3.6.

Componente	Varianza Explicada	Coeficientes
1	30.61 %	[0.047, 0.083, 0.458, 0.505, 0.485, 0.537, 0.051]
2	15.42 %	[0.185, -0.094, 0.008, 0.255, -0.109, -0.237, 0.908]

Cuadro 3.6: Resultados del Análisis de Componentes Principales

En la figura 3.10 se muestra la visualización de los componentes principales.

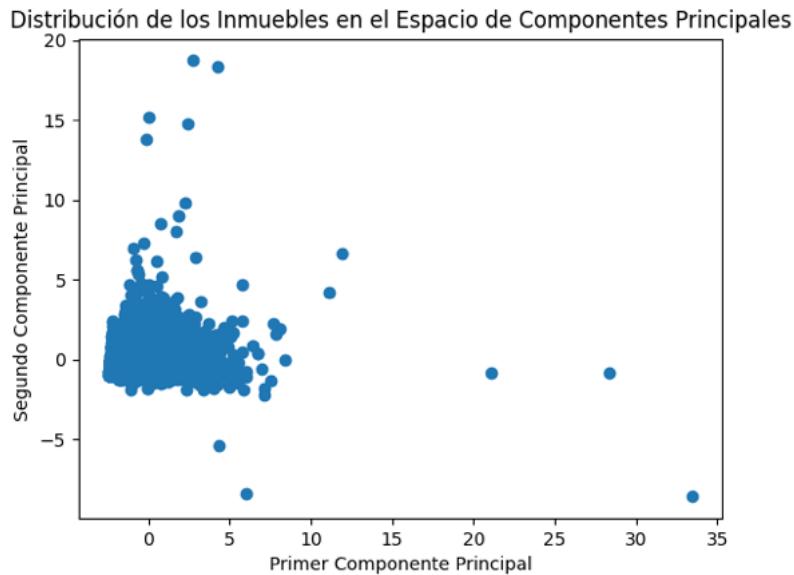


Figura 3.10: Visualización de los componentes principales.

De los resultados obtenidos, podemos derivar las siguientes conclusiones:

1. Las características físicas de los inmuebles, como el tamaño de la construcción, el número de habitaciones, baños y estacionamientos, son las que más varían y tienen mayor influencia en el primer componente principal.
2. La edad del inmueble es una dimensión importante de variabilidad y es el factor predominante en el segundo componente principal.
3. Estos dos componentes capturan casi la mitad de la variabilidad total de los datos, lo que indica una distribución equilibrada de la influencia de las diferentes variables.
4. Se recomienda visualizar estos componentes principales en un gráfico bidimensional para identificar patrones o agrupaciones interesantes en los datos.
5. Dependiendo de los objetivos del análisis, podría ser útil considerar aumentar el número de componentes principales para capturar más variabilidad.

3.5. Algoritmos de Aprendizaje Automático

Para la realización de este proyecto se deberán considerar múltiples algoritmos de aprendizaje automático, los cuales se utilizarán para el entrenamiento de los modelos de valuación inmobiliaria. Estos algoritmos serán utilizados para realizar una comparativa entre el rendimiento y ajuste de cada uno con el objetivo de poder determinar, una vez finalizada la creación del conjunto de datos final, cuál algoritmo emplear para la creación del modelo de valuación inmobiliaria final.

3.5.1. Máquinas de Soporte Vectorial

Método efectivo en espacios de alta dimensión. Utiliza hiperplanos para separar clases de datos. Los hiperplanos se eligen maximizando el margen entre las clases. Las SVMs pueden usar diferentes tipos de kernels, como lineal, polinomial y radial, para adaptarse a la naturaleza no lineal de algunos datos [2].

3.5.2. Random Forest

El algoritmo de bosques aleatorios, propuesto por L. Breiman en 2001, ha sido extremadamente exitoso como un método de clasificación y regresión de propósito general. El enfoque, que combina varios árboles de decisión aleatorizados y agrega sus predicciones mediante promedios, ha mostrado un excelente rendimiento en entornos donde el número de variables es mucho mayor que el número de observaciones. Además, es lo suficientemente versátil como para ser aplicado a problemas a gran escala, se adapta fácilmente a varias tareas de aprendizaje ad hoc y devuelve medidas de la importancia de las variables [85].

3.5.3. Redes Neuronales Artificiales

Las redes neuronales son un modelo de aprendizaje automático inspirado en el funcionamiento del cerebro humano. Las redes neuronales se componen de neuronas artificiales que se conectan entre sí para formar una red. Cada neurona artificial tiene una o más entradas y una salida. Las entradas de una neurona artificial están conectadas a otras neuronas artificiales a través de conexiones llamadas pesos. Cada conexión tiene un peso asociado con él. El peso de una conexión determina la influencia de una neurona en otra. Las neuronas artificiales utilizan una función de activación para determinar su

salida en función de sus entradas y pesos. Las redes neuronales se utilizan para resolver problemas de clasificación y regresión [86].

3.6. Estudio de Factibilidad

Para poder determinar si el proyecto es factible, se realizó un estudio de factibilidad, el cual se divide en tres partes: factibilidad técnica, factibilidad económica y factibilidad operativa.

El objetivo general será determinar si el planteamiento del proyecto es viable y si se puede llevar a cabo con los recursos disponibles.

3.6.1. Factibilidad Técnica

En el apartado de factibilidad técnica se analizaron los recursos técnicos necesarios para poder llevar a cabo el proyecto planteado en el presente documento. En el Cuadro 3.7 se muestra el análisis de factibilidad técnica.

Recurso	Disponibilidad	Viabilidad
Equipo de cómputo	Sí	Sí
Software	Sí	Sí
Conexión a internet	Sí	Sí
Plataforma en la nube	Sí	Sí

Cuadro 3.7: Factibilidad Técnica

Dado que todos los recursos necesarios para llevar a cabo el proyecto se encuentran disponibles, se puede concluir que el proyecto es factible desde el punto de vista técnico.

3.6.2. Factibilidad Económica

Para llevar a cabo el proyecto será necesario contar con un presupuesto, el cual se utilizará para la adquisición de recursos necesarios para el desarrollo del mismo. En el Cuadro 3.8 se muestra el análisis de factibilidad económica.

Recurso	Costo	Viabilidad
AWS	\$ 10 USD/mes	Sí
Equipo de cómputo	\$ 20,000.00	Sí
Recursos Humanos	\$ 15,000.00/mes	Sí

Cuadro 3.8: Factibilidad Económica

*Nota: AWS ofrece un plan gratuito que incluye 750 horas de uso de instancias EC2 t2.micro, 5 GB de almacenamiento EBS, 40 GB de almacenamiento EFS, 750 horas de uso de RDS, 25 horas de uso de DynamoDB, 15 GB de ancho de banda de salida, 1 millón de solicitudes de Lambda y 1 millón de solicitudes de API Gateway al mes durante un año [87]. A pesar de esto, se incluyó el costo de AWS en el análisis de factibilidad económica para poder tener un presupuesto más realista e inclusive realizar las proyecciones pertinentes en caso de que se requiera utilizar recursos adicionales a los ofrecidos en el plan gratuito.

Debido a que se estiman 6 meses de trabajo, los costos finales del proyecto se observan en el Cuadro 3.9.

Recurso	Costo
AWS	\$ 60 USD (\$1,029.00 MN)
Equipo de cómputo	\$20,000.00 MN
Recursos Humanos	\$90,000.00 MN
Costo Total	\$ 111,060.00 MN

Cuadro 3.9: Costos del Proyecto (6 meses) en Moneda Nacional

Dado que el costo total del proyecto es de \$111,060.00 MN y que se cuenta con un presupuesto de \$150,000.00 MN, se puede concluir que el proyecto es factible desde el punto de vista económico y contempla un margen de \$38,940.00 MN para gastos adicionales.

3.6.3. Factibilidad Operativa

Debido a que el proyecto se realizará en un periodo de 6 meses, se debe contar con el tiempo suficiente para llevar a cabo cada una de las tareas, al igual que el equipo de trabajo deberá conocer las herramientas a utilizar para el desarrollo del proyecto. En el Cuadro 3.10 se muestra el análisis de factibilidad operativa.

Recurso	Disponibilidad	Viabilidad
Tiempo	Sí	Sí
Equipo de trabajo	Sí	Sí
Conocimiento	Sí	Sí

Cuadro 3.10: Factibilidad Operativa

Dado que se cuenta con el tiempo necesario para llevar a cabo el proyecto, se cuenta con el equipo de trabajo y dicho equipo cuenta con el conocimiento para desarrollar el proyecto, se puede concluir que el proyecto es factible desde el punto de vista operativo.

3.6.4. Factibilidad General

Dado que el proyecto es factible desde el punto de vista técnico, económico y operativo, podemos concluir que el proyecto es *factible* y se puede llevar a cabo con los recursos disponibles.

3.6.5. Consideraciones Legales

Los datos que se utilizarán para el desarrollo del proyecto son datos obtenidos de distintas plataformas online de bienes raíces, por lo que se debe tener en cuenta las políticas de privacidad de cada una de las plataformas para poder utilizar los datos de manera adecuada.

En México, la Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP) es la ley que regula el tratamiento de los datos personales en posesión de los particulares, con el fin de regular su uso, conservación y transferencia [88].

Dado que no se utilizarán datos personales para el desarrollo del proyecto, no se deberá tener ningún problema con la LFPDPPP. Sin embargo, y debido a que los datos iniciales (sin preprocesamiento) contienen los campos de “Descripción” y “Título” del anuncio (los cuales son campos abiertos y podrían contener información sensible), se debe de tener presente esta ley para descartar el uso de dichos campos en el entrenamiento del modelo.

3.7. Análisis de Riesgos

Todo proyecto conlleva riesgos, los cuales pueden afectar el desarrollo del mismo. A bien de poder realizar una planeación apropiada del proyecto a

llevar a cabo, se realizó un análisis de riesgos, en el cual se identificaron los riesgos más importantes que pueden afectar el desarrollo del proyecto. Para cada uno de los riesgos identificados, se utilizó una matriz de riesgos como se puede observar en la Figura 3.11.

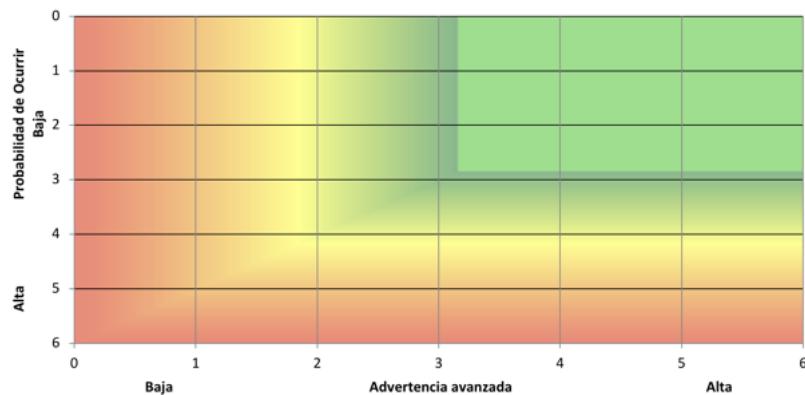


Figura 3.11: Matriz de riesgos inicial.

En esta matriz se pueden visualizar tres parámetros importantes para cada riesgo:

- **Impacto:** El impacto que tendrá el riesgo en el proyecto, el cual se encuentra en una escala de 1 a 5, donde 1 es el impacto más bajo y 5 es el impacto más alto.
- **Probabilidad:** La probabilidad de que el riesgo ocurra, la cual se encuentra en una escala de 1 a 5, donde 1 es la probabilidad más baja y 5 es la probabilidad más alta.
- **Advertencia Avanzada:** El tiempo de advertencia que se tiene antes de que el riesgo ocurra, la cual se encuentra en una escala de 1 a 5, dónde 1 representa un riesgo totalmente desconocido y 5 representa un riesgo que se puede predecir con mucha anticipación.

Con estos tres parámetros se puede calcular el nivel de riesgo de cada riesgo identificado, el cual, por su ubicación en la matriz, se puede clasificar en bajo, medio o alto. Con esta información se puede realizar una planeación apropiada para cada riesgo, con el fin de mitigar el impacto que este pueda tener en el proyecto.

3.7.1. Riesgo de Cronograma

Si se realizó una mala estimación sobre el tiempo que tomará la realización del presente proyecto o si surgió una afectación a algún recurso, sea o no humano, se puede presentar un retraso en la entrega del proyecto. En el Cuadro 3.11 se muestra el riesgo de cronograma.

Impacto	Advertencia Avanzada	Probabilidad
Medio (3)	Conocido - informado, documentado (5)	Baja (1)

Cuadro 3.11: Riesgo de Cronograma

En la Figura 3.12 se muestra la ubicación del riesgo de cronograma en la matriz de riesgos.

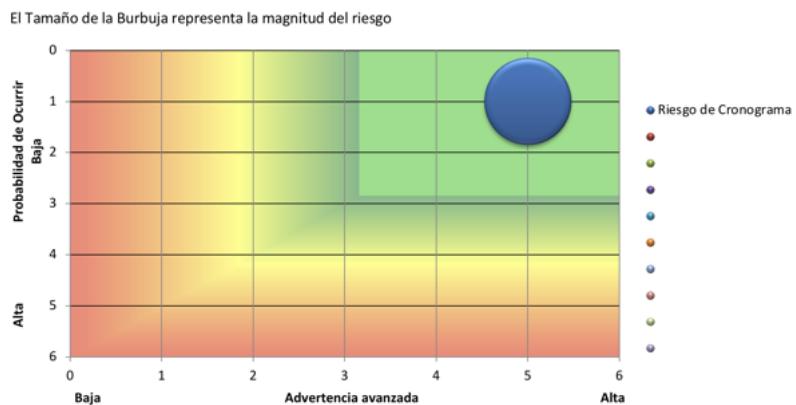


Figura 3.12: Matriz de riesgo de cronograma.

Mitigación de los riesgos de cronograma

Para mitigar el riesgo de cronograma, se realizará una planeación adecuada de las tareas a realizar, con el fin de poder realizar una estimación adecuada del tiempo que tomará cada una de las tareas. Por otra parte, se realizará un seguimiento constante del avance del proyecto, con el fin de poder detectar cualquier retraso en el cronograma y poder tomar las acciones necesarias para poder cumplir con los tiempos establecidos.

A su vez, se asignarán las tareas con tiempo de sobra para poder cumplir con los tiempos establecidos, con el fin de poder tener un margen de error en caso de que se presente algún retraso en el cronograma.

3.7.2. Riesgo de Costos

Si se llegara a rebasar el límite de presupuesto asignado para el proyecto, se puede presentar un retraso en la entrega del proyecto. En el Cuadro 3.12 se muestra el riesgo de costos.

Impacto	Advertencia Avanzada	Probabilidad
Medio Alto (4)	Conocido - no informado, no documentado (2)	Media (2)

Cuadro 3.12: Riesgo de Costos

En la Figura 3.13 se muestra la ubicación del riesgo de costos en la matriz de riesgos.

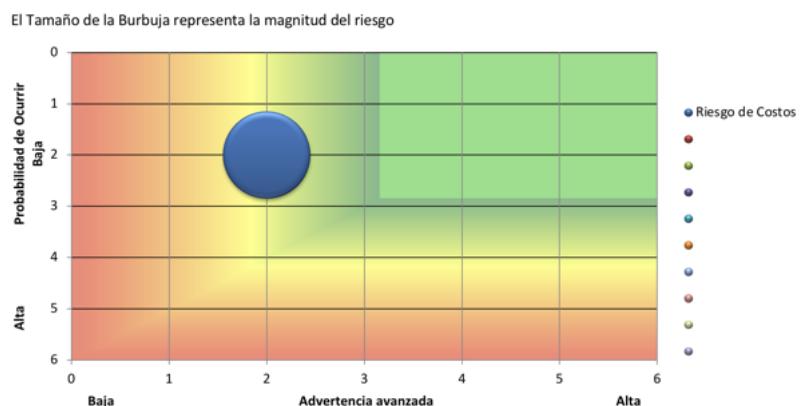


Figura 3.13: Matriz de riesgo de costos.

Mitigación de los riesgos de costos

Para prevenir que excedamos el presupuesto del proyecto, o que se presenten costos no contemplados en el presupuesto, se realizará una planeación adecuada de los recursos necesarios para el desarrollo del proyecto, con el fin de poder contar con un presupuesto adecuado para el desarrollo del proyecto. Igualmente, se realizará la planeación considerando un margen de error, con el fin de poder disponer de recursos adicionales en caso de que se presenten costos no contemplados.

3.7.3. Riesgo Tecnológico

En caso de que se presenten cambios en las tecnologías seleccionadas para el proyecto, o si hubiese un cambio en las herramientas de desarrollo, o en los datos disponibles para el entrenamiento del modelo, se puede presentar un retraso en la entrega del proyecto. En el Cuadro 3.13 se muestra el riesgo tecnológico.

Impacto	Advertencia Avanzada	Probabilidad
Medio Alto (4)	Conocido - no informado, documentado (3)	Alta (3)

Cuadro 3.13: Riesgo Tecnológico

En la Figura 3.14 se muestra la ubicación del riesgo tecnológico en la matriz de riesgos.

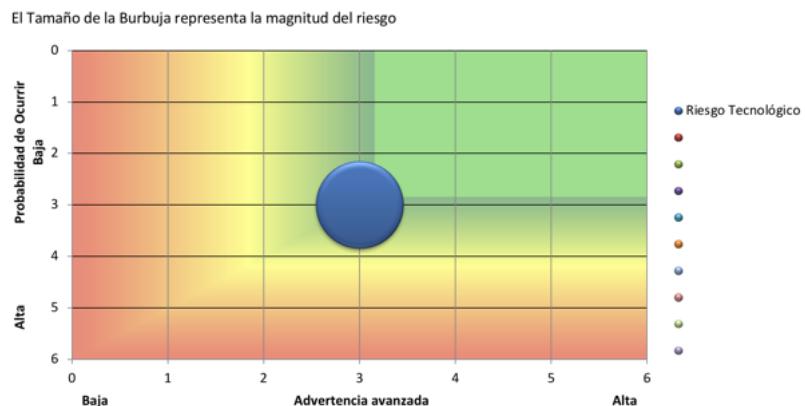


Figura 3.14: Matriz de riesgo tecnológico.

Mitigación de los riesgos tecnológicos

Para mitigar el riesgo tecnológico, se realizará una investigación adecuada de las tecnologías a utilizar, con el fin de poder seleccionar las tecnologías más adecuadas para el desarrollo del proyecto. Se considerarán las aptitudes y conocimientos del equipo de trabajo, así como la documentación disponible para cada una de las tecnologías a utilizar.

3.7.4. Riesgo Operacional

Al ser un proyecto que puede prestarse a muchos criterios sobre las conclusiones que pudiesen derivarse sobre los datos empleados para el mismo, se puede presentar un riesgo operacional. Esto puede presentarse tanto en falta de aptitudes del equipo de trabajo, falta de conocimiento sobre el tema, o falta de conocimiento sobre las herramientas empleadas. En el Cuadro 3.14 se muestra el riesgo operacional.

Impacto	Advertencia Avanzada	Probabilidad
Medio (3)	Conocido - informado, no documentado (4)	Media (2)

Cuadro 3.14: Riesgo Operacional

En la Figura 3.15 se muestra la ubicación del riesgo operacional en la matriz de riesgos.

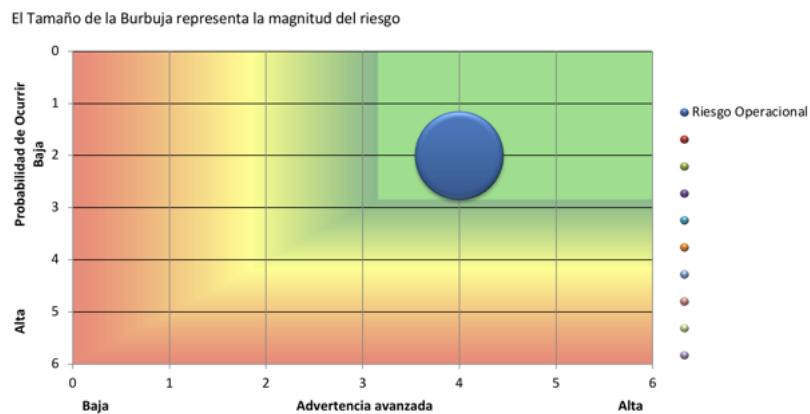


Figura 3.15: Matriz de riesgo operacional.

Mitigación de los riesgos operacionales

Para mitigar el riesgo operacional, se realizará una investigación adecuada de las conclusiones que se puedan derivar de los datos, con el fin de poder delimitar y documentar aquellas que resulten más relevantes para el proyecto.

3.7.5. Riesgos Externos

Si las plataformas en la nube descontinuaran algún servicio utilizado para el alojamiento, procesamiento o análisis de datos, se puede presentar un retraso

en la entrega del proyecto. En el Cuadro 3.15 se muestra el riesgo externo.

Impacto	Advertencia Avanzada	Probabilidad
Medio Alto (4)	Conocido - informado, no documentado (4)	Baja (1)

Cuadro 3.15: Riesgo Externo

En la Figura 3.16 se muestra la ubicación del riesgo externo en la matriz de riesgos.

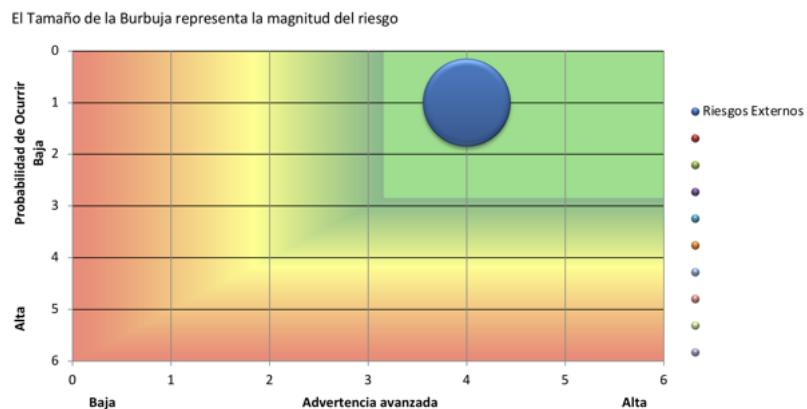


Figura 3.16: Matriz de riesgo externo.

Mitigación de los riesgos externos

Para prevenir que algún actor o factor externo pudiese afectar en el desarrollo del proyecto, se realizará una investigación adecuada de las plataformas en la nube a utilizar, con el fin de poder seleccionar aquellas que sean más estables así como los servicios que ofrezcan, con el fin de poder evitar que se presenten cambios en los servicios que puedan afectar el desarrollo del proyecto.

3.7.6. Riesgos Legales

Si se llegara a presentar un cambio en las leyes que regulan el uso de los datos empleados para el entrenamiento del modelo, se puede presentar un retraso en la entrega del proyecto. En el Cuadro 3.16 se muestra el riesgo legal.

Impacto	Advertencia Avanzada	Probabilidad
Bajo (2)	Conocido - informado, documentado (5)	Baja (1)

Cuadro 3.16: Riesgo Legal

En la Figura 3.17 se muestra la ubicación del riesgo legal en la matriz de riesgos.

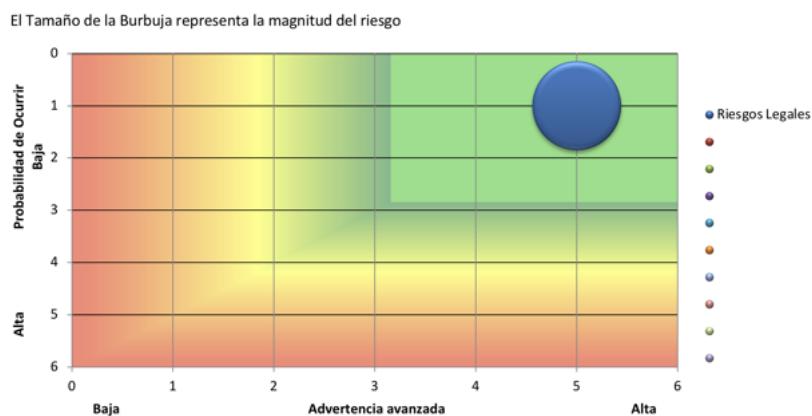


Figura 3.17: Matriz de riesgo legal.

Mitigación de los riesgos legales

Gracias a la realización del estudio de factibilidad, se pudo determinar que no se utilizarán datos personales para el desarrollo del proyecto, por lo que no se deberá tener ningún problema con la Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP). Sin embargo, y debido a que los datos iniciales (sin preprocesamiento) contienen los campos de “Descripción” y “Título” del anuncio (los cuales son campos abiertos y podrían contener información sensible), se debe de tener presente esta ley para descartar el uso de dichos campos en el entrenamiento del modelo.

3.7.7. Riesgos generalizados

En la Figura 3.18 se muestra la ubicación de cada uno de los riesgos descritos anteriormente a bien de poder visualizarlos en conjunto.

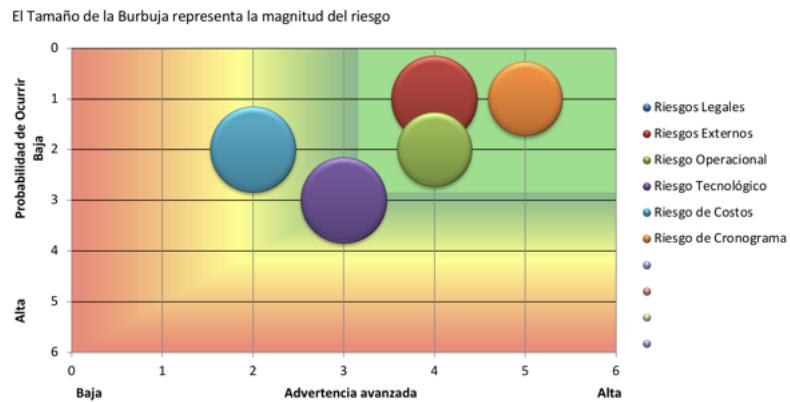


Figura 3.18: Matriz de riesgos generalizados.

3.7.8. Exposición al Riesgo

En la Figura 3.19 se muestra la exposición al riesgo de cada uno de los riesgos descritos anteriormente a bien de poder tener una idea de con cuales riesgos se deberá tener más precaución durante el desarrollo del proyecto.

Descripción del Riesgo	Impacto	Probabilidad de Ocurrir	Advertencia Avanzada	Exposición al Riesgo	
				Impacto	Probabilidad de Ocurrir
Riesgos Legales	2	1	5	2	2
Riesgos Externos	4	1	4	4	4
Riesgo Operacional	3	2	4	6	6
Riesgo Tecnológico	4	3	3	12	12
Riesgo de Costos	4	2	2	8	8
Riesgo de Cronograma	3	1	5	3	3

Figura 3.19: Exposición al riesgo.

Debido a que el proyecto se encuentra mayoritariamente expuesto al riesgo tecnológico de costos, se tendrá especial cuidado con los aspectos de selección de herramientas tecnológicas y de planeación de costos. Por otra parte, se preferirá utilizar herramientas de código abierto para evitar costos de licencias y cambios abruptos en las herramientas utilizadas.

Capítulo 4

Diseño

4.1. Arquitectura del Sistema

Para el desarrollo del sistema que permitirá a los distintos actores del sector inmobiliario realizar estimaciones de precios de departamentos en la Ciudad de México, se propone un sistema web basado en inteligencia artificial usando redes neuronales que permita a los usuarios ingresar la información del departamento y obtener una estimación del precio del mismo.

Para el desarrollo del proyecto se utilizará una arquitectura básica de aplicación web. Siendo así sus elementos un cliente web escrito en React, una API HTTP con el modelo pre-entrenado en el conjunto de datos extraído, y configuraciones propias del entorno web para proveer de una buena experiencia y eficiencia en el uso de la aplicación. En la Figura 4.1 se muestra la arquitectura del sistema.

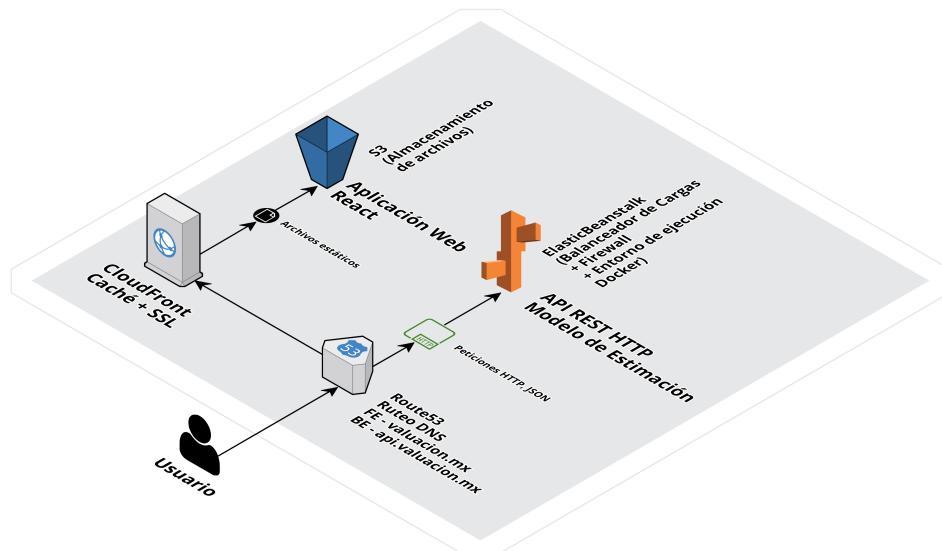


Figura 4.1: Arquitectura del sistema

4.1.1. Descripción de la arquitectura

A continuación se describe cada uno de los elementos que componen la arquitectura del sistema.

1. **Cliente web:** El cliente web es la interfaz de usuario que utilizarán los usuarios para interactuar con el sistema. El cliente web está escrito en React, un framework de JavaScript para el desarrollo de interfaces de usuario.
2. **API REST HTTP:** La API HTTP es el servicio web que provee el modelo de inteligencia artificial para la estimación de precios de departamentos. La API HTTP está escrita en Python utilizando el framework Flask e implementa el modelo de inteligencia artificial utilizando la librería TensorFlow.
3. **Route53:** Route53 es un servicio de Amazon Web Services que permite la administración de los nombres de dominio y la resolución de DNS, aquí direcccionaremos el dominio de la aplicación web y de la API HTTP.
4. **CloudFront:** CloudFront es un servicio de Amazon Web Services que permite la distribución de contenido de manera eficiente y segura, aquí se almacenarán los archivos estáticos del cliente web. Además, provee de un certificado SSL para la comunicación segura entre el cliente web y la API HTTP.

5. **S3:** S3 es un servicio de Amazon Web Services que permite el almacenamiento de archivos de manera segura y escalable, aquí se guardarán los archivos estáticos del cliente web.
6. **Elastic Beanstalk:** Elastic Beanstalk es un servicio de Amazon Web Services que permite el despliegue de aplicaciones web de manera rápida y sencilla, aquí se desplegará la API HTTP. Además, provee de un certificado SSL para la comunicación segura entre el cliente web y la API HTTP.

4.1.2. Caso de Uso

Un caso de uso se define como *una secuencia de acciones, incluyendo variaciones, que el sistema puede ejecutar y que produce un resultado observable de valor para un actor que interactúa con el sistema* [89].

Dado que el proyecto se compone de una única interacción entre el usuario y el sistema, se cuenta con un único caso de uso, dónde el usuario envía la información pertinente al sistema y este le devuelve la estimación de precio para el departamento. En la Figura 4.2 se muestra el diagrama de caso de uso.

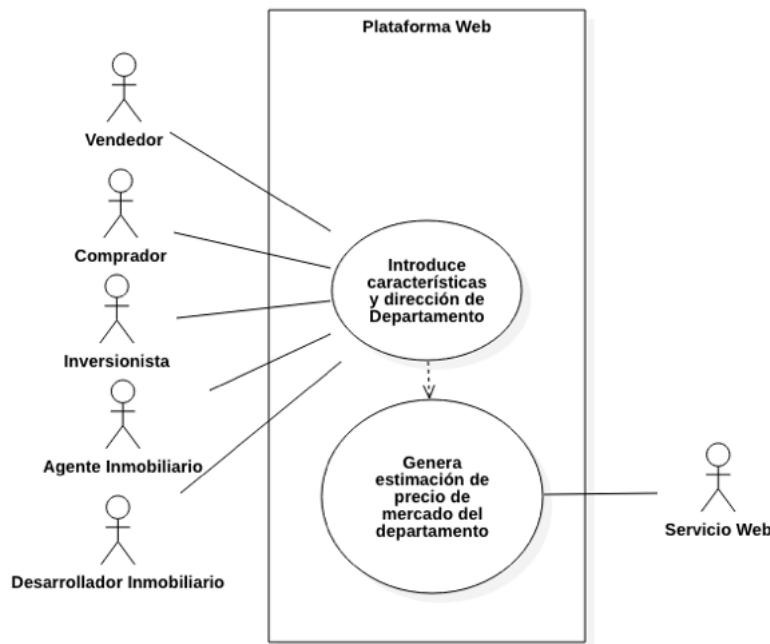


Figura 4.2: Diagrama de caso de uso

En el Cuadro 4.1 se muestra la descripción del caso de uso.

Caso de Uso	Estimar precio de un departamento
Descripción	El usuario envía la información del departamento al sistema y este le devuelve la estimación de precio.
Actores	Usuario
Precondiciones	El usuario debe tener la información del departamento.
Postcondiciones	El usuario recibe la estimación de precio del departamento.
Escenario Principal	<ol style="list-style-type: none">1. El usuario envía la información del departamento al sistema.2. El sistema devuelve la estimación de precio del departamento.

Cuadro 4.1: Descripción del caso de uso

4.1.3. Reglas de Negocio

Las reglas de negocio permiten definir las condiciones para establecer un negocio o precisar de qué forma se controlará el comportamiento de los eventos dentro de este. Muy importante resulta establecer los pasos que se deben seguir para alcanzar las metas y objetivos del mismo. Las reglas de negocio son requisitos que definen cómo el negocio debe operar [90].

A continuación se presentan las reglas de negocio del sistema:

1. **Validez de la Dirección:** Cada dirección ingresada debe ser verificada contra una base de datos actualizada de direcciones válidas de la Ciudad de México para asegurar la precisión de la ubicación y la relevancia de la estimación de precios proporcionada.
2. **Integridad de los Datos:** Los usuarios deben proporcionar información completa y precisa para cada departamento. El sistema rechazará entradas que contengan datos faltantes, ambiguos o claramente erróneos, como valores negativos para áreas de superficie.
3. **Privacidad de los Datos:** La plataforma se compromete a proteger la privacidad de los usuarios. Los datos personales recopilados durante el proceso de estimación se manejarán de acuerdo con las normativas

de protección de datos vigentes, asegurando la confidencialidad y el derecho a la privacidad de los usuarios.

4. **Limitaciones de la Estimación:** Las estimaciones generadas son aproximaciones basadas en modelos matemáticos y no pueden capturar todas las variables únicas de cada propiedad. Se debe comunicar claramente a los usuarios que el precio de mercado real puede variar.
5. **Actualización de Datos:** Para mantener la precisión de las estimaciones, el modelo subyacente se recalibrará periódicamente incorporando nuevos datos del mercado, reflejando así los cambios y tendencias actuales en los precios de los departamentos.
6. **Tratamiento de Outliers:** Durante la fase de análisis y antes del entrenamiento del modelo predictivo, se identificarán y excluirán los valores atípicos para evitar distorsiones en la estimación de precios.
7. **Selección de Características:** El modelo se construirá utilizando únicamente aquellas características que tengan una correlación estadísticamente significativa con el precio del departamento, optimizando así la precisión de las predicciones.
8. **Manejo de “En Construcción”:** Los departamentos en fase de construcción se manejarán con una etiqueta especial dentro del sistema. Se asignará un valor de antigüedad convencional o se desarrollará un tratamiento estadístico alternativo para estos casos.
9. **Acceso al Servicio:** La plataforma implementará políticas de uso justo para garantizar la disponibilidad y la fiabilidad del servicio, evitando la sobrecarga del sistema y asegurando un acceso equitativo para todos los usuarios.
10. **Respuesta del Sistema:** El sistema está diseñado para proporcionar una estimación de precios de manera eficiente y dentro de un marco de tiempo razonable, buscando maximizar la experiencia del usuario y la utilidad del servicio.

4.1.4. Historias de Usuario

Una historia de usuario describe alguna funcionalidad que debe ser valiosa a un usuario o comprador de un sistema o software [91]. Las historias de usuario se componen de tres aspectos fundamentales:

1. **Descripción:** Una descripción corta de la historia de usuario.

2. **Criterios de Aceptación:** Una lista de criterios que deben ser cumplidos para que la historia de usuario sea aceptada.
3. **Conversaciones:** Una lista de conversaciones que se han tenido con el usuario para entender mejor la historia de usuario.

Al igual que en la sección anterior, dado que el proyecto se compone de una única interacción entre el usuario y el sistema, se cuenta con una única historia de usuario, dónde el usuario envía la información pertinente al sistema y este le devuelve la estimación de precio para el departamento. En el Cuadro 4.2 se muestran las historias de usuario del sistema.

Historia de Usuario	Estimar precio de un departamento
Descripción	El usuario envía la información del departamento al sistema y este le devuelve la estimación de precio.
Criterios de Aceptación	<ol style="list-style-type: none"> 1. El usuario debe tener la información del departamento. 2. El usuario debe recibir la estimación de precio del departamento.

Cuadro 4.2: Historia de usuario

4.1.5. Diagrama de Secuencia

Los diagramas de secuencia se utilizan para presentar el comportamiento dinámico del diseño de un sistema, mientras que los diagramas de clases representan la estructura estática del sistema. Como uno de los dos tipos de diagramas de interacción UML, un diagrama de secuencia muestra las interacciones entre objetos organizados en una secuencia temporal [92].

En la Figura 4.3 se muestra el diagrama de secuencia correspondiente a la interacción entre el usuario y el sistema.

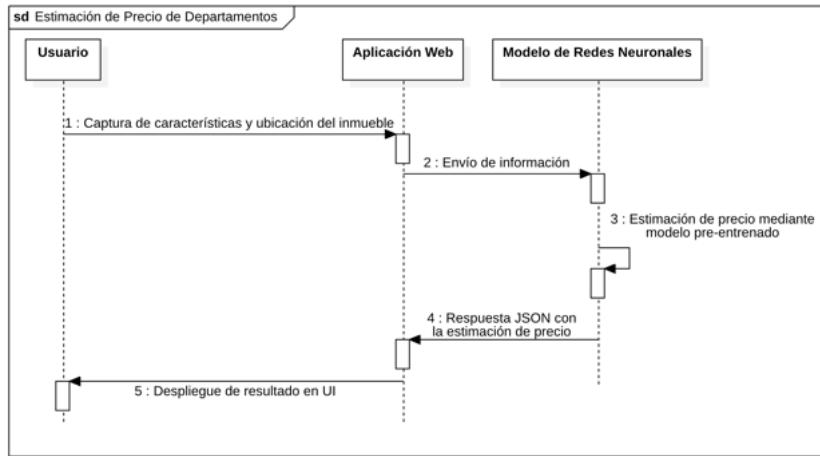


Figura 4.3: Diagrama de secuencia

4.1.6. Requerimientos Funcionales

Los requerimientos funcionales se refieren a las funcionalidades que el sistema debe proveer al usuario [93]. A continuación se presenta un listado con los requerimientos funcionales del sistema. Aunque.

1. **Ingreso de información del departamento:** Permitir al usuario ingresar detalles de un departamento (ubicación, superficie, cantidad de dormitorios, cantidad de baños, antigüedad, etc.).
2. **Estimación de precio del departamento:** Permitir al usuario obtener una estimación del precio del departamento ingresado.
3. **Visualización de información del departamento:** Permitir al usuario visualizar la información ingresada del departamento.
4. **Interfaz de usuario intuitiva:** La interfaz de usuario debe ser intuitiva y fácil de usar.

4.1.7. Requerimientos No Funcionales

Los requerimientos no funcionales sirven para especificar los criterios que se deben cumplir para que el sistema sea aceptado por el usuario [93]. A continuación se presenta un listado con los requerimientos no funcionales del sistema.

1. **Tiempo de respuesta:** El sistema debe responder en un tiempo menor a 1 segundo.

2. **Precisión:** El sistema debe tener una precisión mayor al 80 %.
3. **Usabilidad:** El sistema debe ser fácil de usar.
4. **Escalabilidad:** El sistema debe ser escalable.
5. **Disponibilidad:** El sistema debe estar disponible las 24 horas del día.
6. **Seguridad:** El sistema debe ser seguro.

4.2. Conjunto de Datos

En este apartado se abordará el conjunto de datos utilizado para el desarrollo del proyecto. Se explicará la descripción del conjunto de datos, la obtención del conjunto de datos y el preprocesamiento del conjunto de datos.

4.2.1. Descripción del Conjunto de Datos

Dado que el proyecto se basa en la estimación de precios de departamentos en la Ciudad de México, se utilizó un conjunto de datos de departamentos con características como ubicación, superficie, cantidad de dormitorios, cantidad de baños, antigüedad, etc. y su precio. El conjunto de datos fue obtenido de distintas fuentes, como por ejemplo, el portal de bienes raíces inmuebles24.com [94].

El conjunto de datos cuenta con 40,000 registros, de los cuales 18,000 son únicos. Debido a que los portales inmobiliarios pueden o no incluir múltiples otros datos como el número de estacionamientos, el número de pisos, etc., se decidió utilizar únicamente los datos que se encuentran en la mayoría de los registros, es decir, los datos que se encuentran en más del 50 % de los registros. En la Tabla 4.3 se muestran los datos que se utilizaron para el desarrollo del proyecto.

Datos	Descripción
Ubicación	Alcaldía, colonia, latitud y longitud.
Superficie	Superficie total y superficie construida.
Dormitorios	Cantidad de dormitorios.
Baños	Cantidad de baños.
Antigüedad	Antigüedad del departamento.
Precio	Precio del departamento.

Cuadro 4.3: Datos utilizados para el desarrollo del proyecto

Adicional, se retiene el ID original del anuncio, para poder identificar los registros duplicados y eliminarlos.

4.2.2. Obtención del Conjunto de Datos

Para obtener el conjunto de datos, se construyó un *web scraper* que extrae la información de los departamentos de la Ciudad de México del portal Inmuebles24 [94]. El *web scraper* fue desarrollado en Bash utilizando la herramienta *curl* para realizar las peticiones HTTP y la herramienta *jq* para procesar los datos en formato JSON.

Dicho *web scraper* se ejecutó durante 7 horas, y los datos obtenidos se guardaron en un archivo JSON, con un peso aproximado de 800MB. En la Figura 4.4 se muestra el diagrama de flujo del *web scraper*.

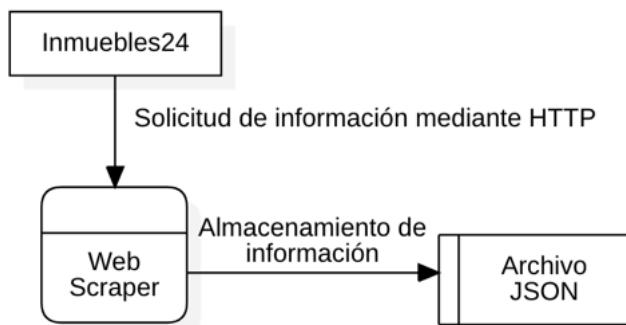


Figura 4.4: Diagrama de flujo del *web scraper*

En el listado 4.1 se muestra el código fuente del *web scraper*.

Listing 4.1: Código fuente del *web scraper*

```
1 #!/bin/bash
2 # Este es un pequeño script de bash que usa cURL para simular correctamente
3 # el ambiente del navegador para hacer una petición al API de Inmuebles24 y
4 # así poder obtener la lista de departamentos en venta en la Ciudad de Mé
5 # xico.
6 # Verificamos que jq esté instalado.
7 if ! command -v jq &> /dev/null
8 then
9     echo "No se encontró jq, por favor instálalo antes de continuar."
10    exit
11 fi
```

```

12 # Generamos un nombre de archivo único para esta ejecución.
13 filename=scrape-$(date +"%Y%m%d-%H-%M-%S").json
14
15 # Inicializamos un arreglo JSON vacío en el archivo.
16 echo "[" > "$filename"
17
18 # Página inicial.
19 pagina=1
20
21 # Ciclo para recorrer los primeros 2015 resultados.
22 # (Nota: el valor se deberá cambiar manualmente dependiendo de los anuncios
23 # disponibles en la plataforma)
24 while (( pagina <= 2015)); do
25     # Imprimos en qué página vamos.
26     echo "Obteniendo página #$pagina..."
27
28     # Realizamos el request a inmuebles24.com
29     response=$(curl -s 'https://www.inmuebles24.com/rplis-api/postings' \
30         -H 'authority: www.inmuebles24.com' \
31         -H 'accept: */*' \
32         -H 'accept-language: en' \
33         -H 'content-type: application/json' \
34         -H 'dnt: 1' \
35         -H 'origin: https://www.inmuebles24.com' \
36         -H 'referer: https://www.inmuebles24.com/departamentos-en-venta-en-\
37             ciudad-de-mexico.html' \
38         -H 'sec-ch-ua: "Chromium";v="118", "Google Chrome";v="118", "Not=A?Brand\
39             ";v="99"' \
40         -H 'sec-ch-ua-mobile: ?0' \
41         -H 'sec-ch-ua-platform: "macOS"' \
42         -H 'sec-fetch-dest: empty' \
43         -H 'sec-fetch-mode: cors' \
44         -H 'sec-fetch-site: same-origin' \
45         -H 'user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) \
46             AppleWebKit/537.36 (KHTML, like Gecko) Chrome/118.0.0.0 Safari/537.36' \
47         -H 'x-requested-with: XMLHttpRequest' \
48         --data-raw '{"q":null,"direccion":null,"moneda":null,"preciomin":null,"\
49             preciomax":null,"services":"","general":"","searchbykeyword":"","\
50             amenidades":"","caracteristicasprop":null,"comodidades":"","disposicion\
51             ":null,"roomType":"","outside":"","areaPrivativa":"","areaComun":"","\
52             multipleRets":"","tipoDePropiedad":"2","subtipoDePropiedad":null,"\
53             tipoDeOperacion":"1","garages":null,"antiguedad":null,"expensasminimo":\
54             null,"expensamaximo":null,"habitacionesminimo":0,"habitacionesmaximo\
55             ":0,"ambientesminimo":0,"ambientesmaximo":0,"banos":null,"\
56             superficieCubierta":1,"idunidadaddemida":1,"metroscuadrado":null,"\
57             metroscuadrado":null,"tipoAnunciante":"ALL","grupoTipoDeMultimedia\
58             ":"","publicacion":null,"sort":"more_recent","etapaDeDesarrollo":"","\
59             auctions":null,"polygonApplied":null,"idInmobiliaria":null,"\
60             excludePostingContacted":"","banks":"","places":"","condominio":"","\
61             pagina":'$pagina',"city":null,"province":69,"zone":null,"valueZone":\
62             null,"subZone":null,"coordenates":null}' \
63         --compressed)
64
65     # Obtenemos el arreglo de anuncios de la respuesta.
66     list_postings=$(echo "$response" | jq '.listPostings')
67
68     # Imprimimos la cantidad de anuncios en la página.
69     echo "Cantidad de anuncios: $(echo "$list_postings" | jq '. | length')"
70
71     # Obtenemos el arreglo de anuncios de la respuesta y removemos los
72     # corchetes iniciales y finales.

```

```

56 list_postings2=${list_postings:1:${#list_postings}-2}
57
58 # Agregamos los datos de list_postings al archivo, seguido de una coma
59 # para separarlo del siguiente set de datos,
60 # solo si hay datos para agregar.
61 if [[ -n $list_postings2 ]]; then
62   echo "$list_postings2," >>"$filename"
63 fi
64
65 # Incrementar el número de página para la siguiente iteración
66 ((pagina++))
67 done
68
69 # Cuando terminamos de iterar, removemos la última coma y agregamos un
70 # corchete final.
71 sed -i '' -e '$ s/,/$//g' -e '$ s/$/]/' "$filename"
72
73 # Imprimimos el nombre del archivo donde se guardaron los datos.
74 echo "Datos guardados en: $filename"

```

4.2.3. Preprocesamiento del Conjunto de Datos

Debido a que los datos obtenidos son súmamente extensos (cada anuncio puede incluir inclusive información personal del anunciante como nombre, teléfono,) y suponen riesgos de privacidad, se optó por la construcción de un *Convertidor* el cual se encarga de procesar el archivo JSON obtenido por el *web scraper* y generar un nuevo archivo CSV con los datos que se utilizarán para el desarrollo del proyecto.

En la Figura 4.5 se muestra el diagrama de flujo del *Convertidor* propuesto.

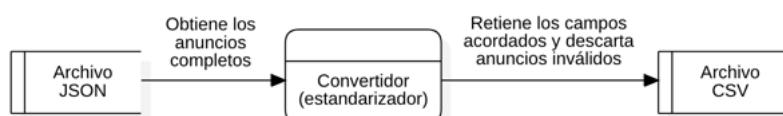


Figura 4.5: Diagrama de flujo del *Convertidor*

4.3. Limpieza del Conjunto de Datos

Tal y cómo se mostró en la sección del análisis exploratorio econométrico, el dataset final que se utilizará para el entrenamiento del modelo de estimación de precios estará sujeto a una serie de transformaciones y limpiezas que permitirán obtener un dataset más limpio y con mejores características para el

entrenamiento del modelo. En esta sección se explicarán las transformaciones y limpiezas que se realizarán al dataset.

4.3.1. Detección de Outliers

Para la detección de outliers se utilizará el método de los cuartiles, el cual consiste en calcular el rango intercuartil (IQR) y luego multiplicarlo por 1.5. Los valores que se encuentren por encima de este valor serán considerados outliers.

4.3.2. Detección de Valores Faltantes

Para la detección de valores faltantes se utilizará el método de la imputación por la media, el cual consiste en reemplazar los valores faltantes por la media de los valores de la columna correspondiente.

4.3.3. Detección de Valores Atípicos

Para la detección de valores atípicos se utilizará el método de la imputación por la mediana, el cual consiste en reemplazar los valores atípicos por la mediana de los valores de la columna correspondiente.

4.3.4. Detección de Valores Negativos

Para la detección de valores negativos se utilizará el método de la imputación por la mediana, el cual consiste en reemplazar los valores negativos por la mediana de los valores de la columna correspondiente.

4.3.5. Detección de Valores Categóricos

Para la detección de valores categóricos se utilizará el método de la codificación one-hot, el cual consiste en crear una columna por cada valor categórico y asignar un 1 si el valor categórico está presente en el registro y un 0 si no lo está.

4.3.6. Detección de Valores Numéricos

Para la detección de valores numéricos se utilizará el método de la normalización, el cual consiste en normalizar los valores numéricos para que estén en el conjunto de números reales.

4.4. Modelo de Estimación de Precios

En esta sección, describiremos el diseño e implementación de un modelo de estimación de precios para una plataforma web de valuación inmobiliaria con Inteligencia Artificial. La estrategia principal consiste en utilizar múltiples modelos para generar diversas estimaciones de precios, seleccionando posteriormente la más adecuada. Esta aproximación permite aprovechar las fortalezas de diferentes técnicas de aprendizaje automático, optimizando así la precisión y fiabilidad de las estimaciones.

4.4.1. Arquitectura del Modelo

La arquitectura del modelo se compone de tres enfoques principales: redes neuronales, máquinas de soporte vectorial (SVM) y árboles aleatorios. Cada uno de estos métodos aporta características únicas en la estimación de precios, permitiendo abarcar distintos aspectos del problema. En la Figura 4.6 se muestra la arquitectura del modelo.

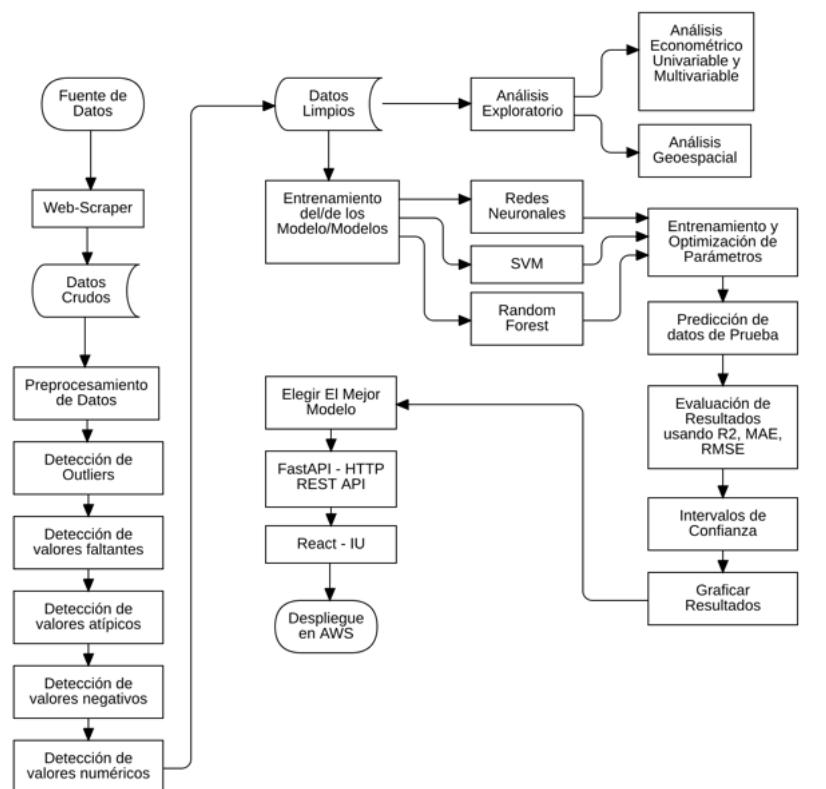


Figura 4.6: Arquitectura del modelo

Red Neuronal

Las redes neuronales se utilizarán para captar patrones complejos en los datos. Su capacidad para aprender representaciones no lineales las hace especialmente útiles en el contexto de los datos inmobiliarios, que a menudo contienen relaciones complejas entre las características.

Máquina de Soporte Vectorial

Las máquinas de soporte vectorial (SVM) se aplicarán para clasificar y predecir los precios de los inmuebles. Su eficacia en espacios de alta dimensión y con margen de separación claro las hace valiosas para este tipo de estimaciones.

Árboles Aleatorios

Los árboles aleatorios, o bosques aleatorios, se usarán por su habilidad para manejar grandes conjuntos de datos y su resistencia a overfitting. Son útiles para capturar relaciones no lineales sin necesidad de un preprocesamiento extenso de los datos.

4.4.2. Entrenamiento del Modelo

El entrenamiento de los modelos se realizará utilizando librerías populares en Python, como Scikit-learn, TensorFlow y PyTorch. Estas herramientas proporcionan una amplia gama de funcionalidades para manejar distintos aspectos del entrenamiento de modelos de aprendizaje automático, desde la preprocesamiento de datos hasta la optimización de hiperparámetros.

4.4.3. Evaluación del Modelo

La evaluación de los modelos implicará el uso de métricas estándar como el error cuadrático medio (MSE) y el coeficiente de determinación (R^2). Estas métricas permitirán determinar la precisión de las estimaciones de precios y comparar la efectividad de los diferentes modelos implementados.

4.4.4. Despliegue del Modelo

Finalmente, el despliegue del modelo se llevará a cabo localmente. Los modelos entrenados se integrarán en un contenedor Docker, junto con un servicio desarrollado en FastAPI. Esta configuración permitirá realizar consultas al modelo a través de una API, donde el usuario podrá enviar la ubicación y

características del inmueble, recibiendo a cambio las estimaciones de precios proporcionadas por los distintos modelos.

4.5. Aplicación Web

El producto final del proyecto es una aplicación web que permita a los usuarios realizar estimaciones de precios de departamentos en la Ciudad de México. En esta sección se explicará la arquitectura de la aplicación web, la interfaz de usuario y el funcionamiento de la aplicación web.

4.5.1. Wireframes

Para el desarrollo de la interfaz de usuario se utilizaron wireframes, los cuales son una representación visual de la estructura de una página web. En la Figura 4.7 se muestra el wireframe inicial de la aplicación web, es decir, lo que el usuario verá al ingresar a la aplicación web. Consiste en un formulario con los campos necesarios para ingresar la información del departamento y un botón para enviar la información al sistema. Se decidió utilizar un formulario debido a que es la manera más sencilla de ingresar información en una aplicación web.

The wireframe shows a web browser window with the following details:

- Header:** "Plataforma de Estimación de Precios de Departamentos en la CDMX".
- Address Bar:** "https://valuacion.mx".
- Title:** "Estima el valor de mercado de un departamento en la CDMX".
- Input Field:** "Introduce la dirección:" followed by a text input containing "Av. Juan de Dios Bótiz, Nueva Industrial Vallejo, Gustavo A. Madero, 07320 Ciudad de México, CDMX".
- Form Fields:**
 - "Número de Baños": Input field with value "3" and up/down arrows.
 - "Número de Recámaras": Input field with value "3" and up/down arrows.
 - "Lugares de estacionamiento": Input field with value "3" and up/down arrows.
 - "Antigüedad": Input field with value "3" and up/down arrows.
 - "Superficie total (m²)": Input field with value "125".
 - "Superficie construida (m²)": Input field with value "110".
- Button:** "Obtener Estimación de Precio".

Figura 4.7: Wireframe con el estado inicial de la aplicación web

Una vez que el usuario haya hecho click en el botón con el cual se envía la información capturada en el formulario, se mostrará una pantalla de carga, mientras el sistema procesa la información y devuelve la estimación de precio del departamento. En la Figura 4.8 se muestra el wireframe correspondiente al resultado final del sistema, es decir, la estimación de precio del departamento.

The wireframe shows a web browser interface for a price estimation platform. The title bar reads "Plataforma de Estimación de Precios de Departamentos en la CDMX". The address bar shows the URL "https://valuacion.mx". The main content area has a heading "Estima el valor de mercado de un departamento en la CDMX". It includes a text input field for "Introduce la dirección:" containing the placeholder "Av. Juan de Dios Bátiz, Nuevo Industrial Vallejo, Gustavo A. Madero, 07320 Ciudad de México, CDMX". Below this are several input fields with dropdown menus for selecting values: "Número de Baños" (3), "Número de Recámaras" (3), "Lugares de estacionamiento" (3), "Antigüedad" (3), "Superficie total (m²)" (125), and "Superficie construida (m²)" (110). A large button labeled "Obtener Estimación de Precio" is centered below these inputs. Below the button is a small square icon labeled "Ícono de la Aplicación". At the bottom, the estimated price is displayed as "Tu departamento se estima en: \$5,750,000.00 MXN Ⓜ". A note below states: "Este precio fue obtenido mediante un modelo de aprendizaje automático entrenado con 18,000 anuncios correspondientes a Agosto del 2023. Esta información es meramente informativa y carece de cualquier validez jurídica." The footer includes the text "Escuela Superior de Cómputo del IPN" and "TT - 2024-A053 - Humberto Alejandro Ortega Alcocer".

Figura 4.8: Wireframe con la estimación de precio del departamento

4.5.2. Dependencias externas

Debido a que la aplicación web deberá ofrecer una funcionalidad de geolocalización, es decir, obtener las coordenadas geográficas a partir de una dirección, se requiere de la integración de un servicio externo que permita realizar dicha funcionalidad. Esto ya que la implementación de un servicio

propio de geo-localización es una tarea compleja y que requiere de una gran cantidad de recursos. Sobre todo porque la información geoespacial de la Ciudad de México es muy extensa y compleja, además de estar sujeta a una constante actualización y modificación.

Google Maps

Google Maps es un servicio de Google que permite la visualización de mapas geográficos. Además, ofrece una API que permite la integración de Google Maps en aplicaciones web. Dicha API ofrece una funcionalidad de geo-localización, la cual permite obtener las coordenadas geográficas a partir de una dirección. En la Figura 4.9 se muestra un ejemplo de la funcionalidad de geo-localización de Google Maps [76].

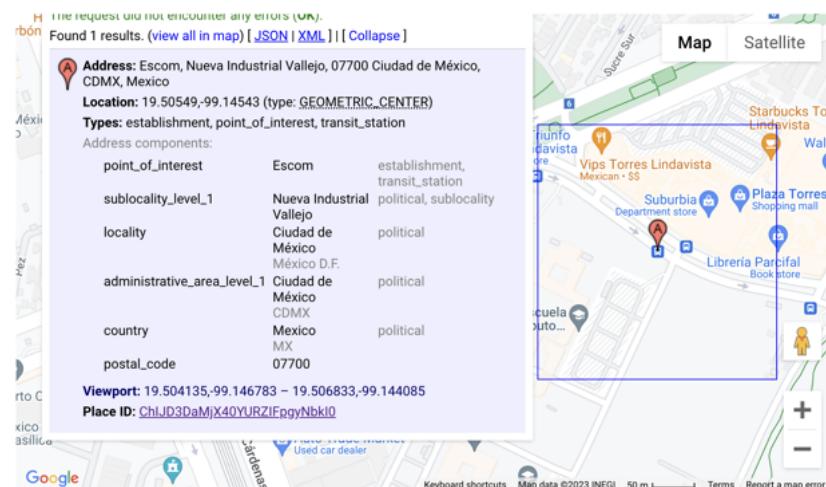


Figura 4.9: Funcionalidad de geo-localización de Google Maps

Capítulo 5

Implementación

5.1. Arquitectura del sistema

La arquitectura final de implementación no difiere en gran medida de la arquitectura discutida en el capítulo de diseño, sin embargo, se realizaron ajustes en función de algunos retos encontrados durante la implementación. La arquitectura final del sistema se muestra en la figura 5.1.

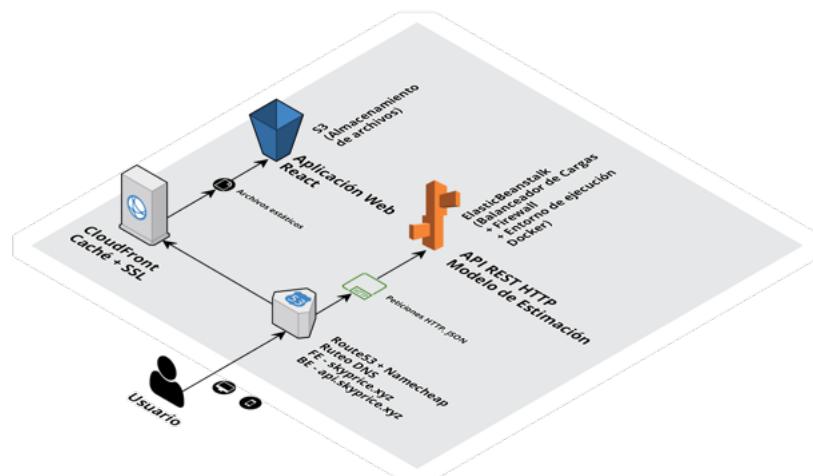


Figura 5.1: Arquitectura final del sistema.

El único aspecto a destacar es la integración de Namecheap como proveedor

de dominios ya que Route53 de AWS no permite la compra de dominios con terminación .xyz, por lo que se optó por utilizar Namecheap para la compra del dominio `skyprice.xyz`.

5.2. Desarrollo de los modelos

En esta sección se abordarán todos los pasos empleados para concluir con el desarrollo de los modelos de aprendizaje automático, empleando los datos descritos previamente y afinando los hiperparámetros de los modelos.

5.2.1. Conjunto de datos

En la figura 5.2 se muestra un vistazo de las características del archivo CSV que se utilizó para el entrenamiento de los modelos. Este archivo contiene la información sobre los departamentos en la Ciudad de México que será utilizada para predecir el precio de las propiedades.

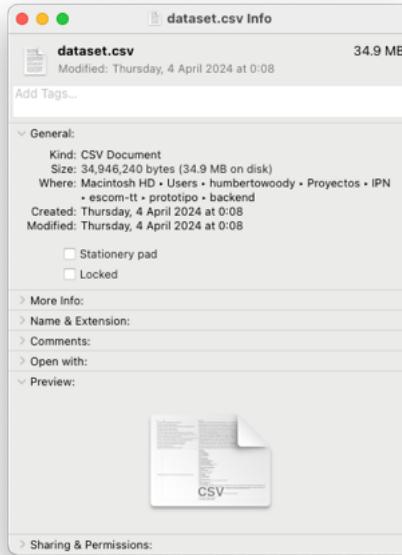


Figura 5.2: Vistazo al conjunto de datos utilizado para el entrenamiento de los modelos.

5.2.2. Preprocesamiento de los datos

El preprocesamiento de los datos incluyó la limpieza de los mismos para eliminar valores atípicos, ajustar los valores faltantes y normalizar las características. Posteriormente, se comienza con la separación de los conjuntos de datos de entrenamiento (75 % de los datos) y de prueba (25 % de los datos) y se crean los escaladores para normalizar las características, utilizando un `StandardScaler` para las características numéricas y un `OneHotEncoder` para las características categóricas.

En el listado 5.1 se muestra el código relevante para realizar el preprocesamiento de los datos usando Python, con las librerías Pandas y Scikit-learn.

Listing 5.1: Preprocesamiento de los datos

```
1 # Dividir el dataset en conjuntos de entrenamiento y prueba
2 X = df[['Municipality', 'Size_Terrain', 'Size_Construction', 'Rooms', 'Bathrooms', 'Parking', 'Age', 'Lat', 'Lng']]
3 y = df['Price']
4 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=TEST_SIZE, random_state=RANDOM_STATE)
5 print(f'Dimensiones de los conjuntos de entrenamiento y prueba: {X_train.shape}, {X_test.shape}')
6
7 # Crear un procesador para las columnas numéricas
8 numeric_transformer = Pipeline(steps=[
9     ('scaler', StandardScaler())
10])
11
12 # Obtener las columnas numéricas
13 numeric_features=columnas_numericas[1:]
14
15 # Crear un procesador para las columnas categóricas
16 preprocessor = ColumnTransformer(
17     transformers=[
18         ('cat', OneHotEncoder(categories=[municipalities]), ['Municipality']),
19         ('num', numeric_transformer, numeric_features),
20     ])
21
```

5.2.3. Entrenamiento de los modelos

Cada algoritmo de aprendizaje automático se entrenó utilizando el conjunto de datos de entrenamiento y se evaluó utilizando el conjunto de datos de prueba. De esta manera, se pudo determinar la precisión de cada modelo y seleccionar los hiperparámetros adecuados para cada uno.

Dado que las combinaciones de hiperparámetros son numerosas, se utilizó la técnica de búsqueda en cuadrícula (`GridSearchCV`) para encontrar la combinación óptima de hiperparámetros para cada modelo, el cual permite especificar una cuadrícula de hiperparámetros y evaluar todas las combinaciones

posibles [95].

El tiempo de entrenamiento promedio para los tres modelos fue de aproximadamente una hora con treinta minutos, esto debido a las permutaciones de hiperparámetros resultantes de la búsqueda en cuadrícula, con el objetivo de realizar distintas pruebas, se decidió por proveer de un modo reducido de hiperparámetros para disminuir el tiempo de entrenamiento y poder realizar pruebas más rápidas, es por esto que en varios fragmentos de código se puede observar la variable `FAST_TEST` que se encarga de seleccionar el modo reducido de hiperparámetros.

Redes neuronales

Las redes neuronales se implementaron usando TensorFlow y Keras, y se experimentó con diferentes configuraciones de capas, neuronas y funciones de activación para mejorar la precisión del modelo. A continuación se muestran los hiperparámetros utilizados en la búsqueda en cuadrícula para las redes neuronales.

- Capas y neuronas por capa: $[(10, 10), (10, 10, 10), (30, 30)]$
- Dropout: $[0.5, 0.7]$
- Tamaño del conjunto de entrenamiento: $[50, 100]$
- Épocas: $[50, 100, 200, 500]$
- Tasa de aprendizaje: $[0.1, 0.01, 0.001]$
- Funciones de activación: $['relu', 'tanh', 'sigmoid']$

Los parámetros cuyos valores no cambiaron son los siguientes:

- Optimizador: Adam
- Función de pérdida: Mean Squared Error

Una vez finalizado el entrenamiento y selección del mejor modelo, se procedió a exportar el modelo a un archivo `.h5` para su posterior importación en el servicio web.

El código que se muestra en el listado 5.2 corresponde al entrenamiento de las redes neuronales, donde se utilizó la técnica de búsqueda en cuadrícula para encontrar los mejores hiperparámetros.

Listing 5.2: Entrenamiento de redes neuronales

```
1 # Preprocesar los datos para la red neuronal
2 X_train_preprocessed = preprocessor.fit_transform(X_train)
```

```

3 # Función para obtener el modelo de red neuronal
4 def get_reg(meta, hidden_layer_sizes, dropout):
5     n_features_in_ = meta["n_features_in_"]
6     model = keras.models.Sequential()
7     model.add(keras.layers.Input(shape=(n_features_in_,)))
8     for hidden_layer_size in hidden_layer_sizes:
9         model.add(keras.layers.Dense(hidden_layer_size, activation="relu"))
10        model.add(keras.layers.Dropout(dropout))
11    model.add(keras.layers.Dense(1))
12    return model
13
14
15 # Crear el modelo de red neuronal
16 reg = KerasRegressor(
17     model=get_reg,
18     loss=keras.losses.MeanSquaredError,
19     optimizer=keras.optimizers.Adam,
20     metrics=[keras.metrics.R2Score],
21     hidden_layer_sizes=(100,),
22     dropout=0.5,
23 )
24
25 # Parámetros a buscar
26 param_grid_nn_fast = {
27     "hidden_layer_sizes": [(10, 10)],
28     "dropout": [0.5],
29     "batch_size": [50],
30     "epochs": [50],
31     "optimizer": [keras.optimizers.Adam],
32     "optimizer__learning_rate": [0.1],
33     "loss": [keras.losses.MeanSquaredError],
34     "metrics": [keras.metrics.R2Score],
35 }
36 param_grid_nn_slow = {
37     "hidden_layer_sizes": [(10, 10), (10, 10, 10), (30, 30)],
38     "dropout": [0.5, 0.7],
39     "batch_size": [50, 100],
40     "epochs": [50, 100, 200, 500],
41     "optimizer": [keras.optimizers.Adam],
42     "optimizer__learning_rate": [0.1, 0.01, 0.001],
43     "loss": [keras.losses.MeanSquaredError],
44     "metrics": [keras.metrics.R2Score],
45 }
46 param_grid_nn = FAST_TEST and param_grid_nn_fast or param_grid_nn_slow
47
48 # Crear el objeto GridSearchCV
49 grid_search_nn = GridSearchCV(
50     estimator=reg,
51     param_grid=param_grid_nn,
52     refit=False,
53     cv=5,
54     verbose=2,
55     n_jobs=-1,
56 )
57
58 # Debido a que el preprocesamiento ya está hecho, podemos usar
      X_train_preprocessed directamente aquí
59 grid_search_nn.fit(X_train_preprocessed, y_train)
60
61 # Obtener los mejores parámetros
62 mejores_parametros = grid_search_nn.best_params_

```

Random Forest

El modelo de Random Forest se entrenó de igual manera que las redes neuronales, utilizando la técnica de búsqueda en cuadrícula para encontrar los mejores hiperparámetros como lo son el número de estimadores, la profundidad máxima y el mínimo de muestras para bifurcar un nodo. A continuación se muestran los hiperparámetros utilizados en la búsqueda en cuadrícula para el modelo de Random Forest:

- Número de estimadores: [100, 500, 1000]
- Profundidad máxima: [None, 10, 20, 30]
- Mínimo de muestras para dividir: [2, 5, 10]

Una vez finalizado el entrenamiento y selección del mejor modelo, se procedió a exportar el modelo a un archivo `.joblib` para su posterior importación en el servicio web.

En el listado 5.3 se muestra el código relevante para entrenar el modelo de Random Forest, utilizando la técnica de búsqueda en cuadrícula.

Listing 5.3: Entrenamiento de Random Forest

```
1 # Crear el pipeline
2 rf_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
3                             ('regressor', RandomForestRegressor(
4                                 random_state=42, oob_score=True, n_jobs=-1))])
5
6 # Parámetros a buscar
7 param_grid_rf_fast = {
8     'regressor__n_estimators': [100],
9     'regressor__max_depth': [None],
10    'regressor__min_samples_split': [2],
11 }
12 param_grid_rf_slow = {
13     'regressor__n_estimators': [100, 500, 1000],
14     'regressor__max_depth': [None, 10, 20, 30],
15     'regressor__min_samples_split': [2, 5, 10]
16 }
17 param_grid_rf = FAST_TEST and param_grid_rf_fast or param_grid_rf_slow
18
19 # Crear el objeto GridSearchCV
20 grid_search_rf = GridSearchCV(rf_pipeline, param_grid=param_grid_rf, cv=5,
21                               verbose=2, n_jobs=-1)
22
23 # Ejecutar la búsqueda
24 grid_search_rf.fit(X_train, y_train)
25
26 # Obtener el mejor modelo
27 rf_pipeline = grid_search_rf.best_estimator_
```

Máquinas de soporte vectorial

El modelo de Máquinas de Soporte Vectorial (SVM) se entrenó utilizando la técnica de búsqueda en cuadrícula para encontrar los mejores hiperparámetros, como lo son el parámetro de regularización C, el parámetro gamma y el parámetro epsilon. A continuación se muestran los hiperparámetros utilizados en la búsqueda en cuadrícula para el modelo de SVM:

- Parámetro de regularización C: [0.1, 1, 100, 1000]
- Parámetro gamma: ['scale', 'auto', 0.01, 0.001]
- Parámetro epsilon: [0.01, 0.1, 1, 10]
- Kernel: ['rbf', 'linear', 'poly', 'sigmoid']

Una vez finalizado el entrenamiento y selección del mejor modelo, se procedió a exportar el modelo a un archivo `.joblib` para su posterior importación en el servicio web.

En el listado 5.4 se muestra el código relevante para entrenar el modelo de Máquinas de Soporte Vectorial, utilizando la técnica de búsqueda en cuadrícula.

Listing 5.4: Entrenamiento de Máquinas de Soporte Vectorial

```
1 # Crear el pipeline
2 svm_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
3                               ('svr', SVR())])
4
5 # Parámetros a buscar
6 param_grid_svm_fast = {
7     'svr__C': [0.1],
8     'svr__gamma': ['scale'],
9     'svr__epsilon': [0.01],
10    'svr__kernel': ['rbf']}
11 }
12 param_grid_svm_slow = {
13     'svr__C': [0.1, 1, 100, 1000],
14     'svr__gamma': ['scale', 'auto', 0.01, 0.001],
15     'svr__epsilon': [0.01, 0.1, 1, 10],
16     'svr__kernel': ['rbf', 'linear', 'poly', 'sigmoid']}
17 }
18 param_grid_svm = FAST_TEST and param_grid_svm_fast or param_grid_svm_slow
19
20 # Crear el objeto GridSearchCV
21 grid_search_svm = GridSearchCV(svm_pipeline, param_grid=param_grid_svm, cv=5, verbose=2, n_jobs=-1)
22
23 # Ejecutar la búsqueda
24 grid_search_svm.fit(X_train, y_train)
25
26 # Obtener el mejor modelo
27 svm_pipeline = grid_search_svm.best_estimator_
```

5.3. Servicio web

Una vez finalizado el entrenamiento de los modelos, se procedió a implementar el servicio web con FastAPI, el cual se encargará de importar los modelos, evaluar su desempeño, generar gráficas de evaluación y exponer una API REST para realizar predicciones de precio y consultar información sobre los modelos.

5.3.1. Importación de los modelos

En el listado 5.5 se muestra el código relevante para inicializar las librerías requeridas por el servicio, posteriormente se procede a importar los modelos de los archivos .h5 y .joblib generados en el entrenamiento de las redes neuronales, Random Forest y Máquinas de Soporte Vectorial. Además, se importan los conjuntos de datos de entrenamiento y prueba para evaluar el desempeño de los modelos.

Listing 5.5: Importación de los modelos

```
1 from fastapi.middleware.cors import CORSMiddleware # Para configurar CORS
2 import datetime # Para obtener la fecha y hora actual
3 from joblib import load # Para cargar los modelos de aprendizaje automático
4 from sklearn.metrics import mean_absolute_error, mean_squared_error,
   root_mean_squared_error # Para calcular el error absoluto medio
5 from pydantic import BaseModel, Field # Para definir la clase de la
   propiedad y sus campos
6 import pandas as pd # Para trabajar con los datos de la propiedad
7 from fastapi import FastAPI # Para crear la aplicación de FastAPI
8 from fastapi.responses import FileResponse # Para servir archivos estáticos
9 import matplotlib # Para configurar el renderizador
10 import matplotlib.pyplot as plt # Para generar gráficas
11 import logging # Para mostrar mensajes de depuración
12 from constants import *
13
14 # Mensaje de depuración
15 logging.basicConfig(level=logging.INFO)
16 logging.info("Cargando modelos de aprendizaje automático...")
17
18 # Elegimos usar el renderizador "Agg" que no requiere usar el entorno grá
   fico
19 # de nuestro Sistema Operativo.
20 matplotlib.use('Agg')
21
22 # Carga de modelos
23 rf_model = load(ARCHIVO_MODELO_RF)
24 svm_model = load(ARCHIVO_MODELO_SVM)
25 nn_model = load(ARCHIVO_MODELO_RN)
26 nn_model_history = load(ARCHIVO_MODELO_RN_HISTORIA)
27 preprocessor = load(ARCHIVO_PREPROCESADOR)
28
29 logging.info("Modelos de aprendizaje automático cargados con éxito")
30 logging.info("Cargando datos de entrenamiento y prueba...")
31
32 # Carga de datos de entrenamiento y prueba
```

```

33 X_train = pd.read_csv(ARCHIVO_X_TRAIN)
34 X_test = pd.read_csv(ARCHIVO_X_TEST)
35 y_train = pd.read_csv(ARCHIVO_Y_TRAIN)
36 y_test = pd.read_csv(ARCHIVO_Y_TEST)
37
38 # Carga del dataset original
39 df = pd.read_csv(ARCHIVO_DATASET)

```

Cargar el conjunto de datos original es útil para obtener información adicional del mismo y poder exponerla dinámicamente a través de la API REST y así permitir a la interfaz gráfica mostrar información relevante sobre las propiedades de entretenimiento.

Se puede observar el uso de constantes como ARCHIVO_MODELO_RF, las cuales se encuentran definidas en un archivo `constants.py` que se encarga de almacenar todas las constantes utilizadas en el servicio web a manera de facilitar su modificación y mantenimiento.

5.3.2. Evaluación de los modelos

Evaluar los modelos al iniciar el servicio web es una parte fundamental para asegurar que el servicio sea independiente a los modelos en sí, y que estos sean capaces de realizar predicciones de manera correcta. En el listado 5.6 se muestra el código relevante para evaluar los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales.

Listing 5.6: Evaluación de los modelos

```

1 # Predecir con cada modelo
2 rf_preds = rf_model.predict(X_test)
3 svm_preds = svm_model.predict(X_test)
4 nn_preds = nn_model.predict(preprocessor.transform(X_test)).flatten()
5
6 # Calcular el error cuadrático medio de cada modelo
7 rf_mse = mean_squared_error(y_test, rf_preds)
8 svm_mse = mean_squared_error(y_test, svm_preds)
9 nn_mse = mean_squared_error(y_test, nn_preds)
10
11 # Calcular el RMSE (raíz del error cuadrático medio) de cada modelo
12 rf_rmse = root_mean_squared_error(y_test, rf_preds)
13 svm_rmse = root_mean_squared_error(y_test, svm_preds)
14 nn_rmse = root_mean_squared_error(y_test, nn_preds)
15
16 # Calcular los intervalos de confianza de cada modelo
17 rf_ci = (rf_preds - y_test.squeeze()).mean() - 1.96 * (rf_preds - y_test.
    squeeze()).std(), (rf_preds - y_test.squeeze()).mean() + 1.96 * (
    rf_preds - y_test.squeeze()).std()
18 svm_ci = (svm_preds - y_test.squeeze()).mean() - 1.96 * (svm_preds - y_test.
    squeeze()).std(), (svm_preds - y_test.squeeze()).mean() + 1.96 * (
    svm_preds - y_test.squeeze()).std()
19 nn_ci = (nn_preds - y_test.squeeze()).mean() - 1.96 * (nn_preds - y_test.
    squeeze()).std(), (nn_preds - y_test.squeeze()).mean() + 1.96 * (
    nn_preds - y_test.squeeze()).std()

```

```

20
21 # Calculamos el error absoluto medio de cada modelo
22 rf_mae = mean_absolute_error(y_test, rf_preds)
23 svm_mae = mean_absolute_error(y_test, svm_preds)
24 nn_mae = mean_absolute_error(y_test, nn_preds)
25
26 # Calculamos el coeficiente de determinación de cada modelo
27 rf_r2 = rf_model.score(X_test, y_test)
28 svm_r2 = svm_model.score(X_test, y_test)
29 nn_r2 = nn_model_history.history['r2_score'][-1]

```

Métricas de evaluación

Las métricas de evaluación utilizadas para evaluar los modelos fueron el error cuadrático medio (MSE), el error absoluto medio (MAE), el coeficiente de determinación (R2) y el intervalo de confianza. Estas métricas permiten comparar el rendimiento de los modelos y determinar el comportamiento de cada uno en diferentes situaciones.

Gráficas de evaluación

Las gráficas de evaluación se generaron empleando la librería Matplotlib, permitiendo visualizar el rendimiento de los modelos de manera gráfica. Un aspecto a considerar cuando se generan gráficas dinámicamente en un servicio web es configurar el renderizador de Matplotlib para que no requiera el entorno gráfico del Sistema Operativo, esto se logra con la línea `matplotlib.use('Agg')`.

Por otro lado, no es óptimo generar gráficas en tiempo real cada vez que se realiza una petición, por lo que se optó por generar las gráficas de evaluación al iniciar el servicio web y servirlas estáticamente a través de una ruta específica. En el listado 5.7 se muestra el código relevante para generar las gráficas de evaluación.

Listing 5.7: Generación de gráficas de evaluación

```

1 # Generación de gráficas
2 models = {'Random Forest': rf_preds, 'SVM': svm_preds, 'Neural Network':
3           nn_preds}
4 axs = plt.subplots(1, 3, figsize=(15, 5))[1]
5
6 # Graficar las predicciones vs valores reales
7 for ax, (model_name, preds) in zip(axs, models.items()):
8     ax.scatter(y_test, preds, alpha=0.5)
9     ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
10            lw=2)
11    ax.set_xlabel('Valores Reales')
12    ax.set_ylabel('Predicciones')
13    ax.set_title(model_name)

```

```

13 # Ajustar el espacio entre las gráficas
14 plt.tight_layout()
15
16 # Servimos la gráfica sin tocar el disco
17 plt.savefig('plots.png')
18
19 # Cerramos la gráfica
20 plt.close('all')

```

En la figura 5.3 se muestra una de las gráficas de evaluación generadas al iniciar el servicio web, la cual permite visualizar las predicciones de los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales en comparación con los valores reales.

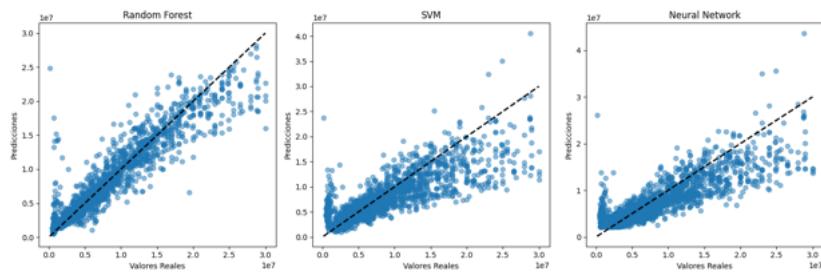


Figura 5.3: Gráfica de evaluación de los modelos de aprendizaje automático.

5.3.3. API REST

El servicio web exponer cuatro rutas principales, las cuales permiten a la interfaz gráfica interactuar con los modelos de aprendizaje automático y obtener información relevante sobre los mismos. Además, FastAPI permite documentar la API de manera automática, lo que facilita la comprensión y el uso de la misma.

En el listado 5.8 se muestra el código relevante para la inicialización del servicio, la configuración de CORS y la definición de los modelos que se utilizarán en la API para que FastAPI pueda generar la documentación de la API de manera automática.

Listing 5.8: API REST

```

1 # Inicialización de la aplicación
2 app = FastAPI(title="SkyPrice API",
3                 description="Hola, bienvenido a la API de **SkyPrice**. Aquí
4                 puedes predecir el precio de una propiedad utilizando tres modelos de
5                 aprendizaje automático: Random Forest, SVM y Redes Neuronales. Para
6                 predecir el precio de una propiedad, envía una solicitud POST a la ruta
7                 /predict con los datos de la propiedad. También puedes obtener información
8                 sobre los modelos y sus características en la ruta /models. ¡Diviértete!",
9

```

```

4         version="1.0.0",
5         docs_url="/openapi",
6         openapi_url="/openapi.json",
7         redoc_url="/redoc",
8         summary="API para predecir el precio de un departamento en
9         la CDMX",
10        contact={
11            "name": "Humberto Alejandro Ortega Alcocer",
12            "email": "hortegaa1500@alumno.ipn.mx",
13            "url": "https://humbertowoodly.xyz",
14        },
15        license_info={
16            "name": "MIT",
17            "url": "https://opensource.org/licenses/MIT",
18        },
19        terms_of_service="https://opensource.org/licenses/MIT",
20        servers=[
21            {
22                "url": f"{HOSTNAME}",
23                "description": "URL de la API"
24            }
25        ]
26    )
27
28 # Configuración de CORS
29 app.add_middleware(
30     CORSMiddleware,
31     allow_credentials=True,
32     allow_origins=["*"],
33     allow_methods=["*"],
34     allow_headers=["*"],
35 )
36
37 # Definición de una propiedad
38 class Property(BaseModel):
39     Size_Terrain: float = Field(..., examples=[120.5], description="El tamaño del terreno en metros cuadrados")
40     Size_Construction: float = Field(..., examples=[250.0], description="El tamaño de la construcción en metros cuadrados")
41     Rooms: int = Field(..., examples=[3], description="El número de habitaciones")
42     Bathrooms: float = Field(..., examples=[2.5], description="El número de baños")
43     Parking: int = Field(..., examples=[1], description="El número de espacios de estacionamiento disponibles")
44     Age: int = Field(..., examples=[5], description="La edad de la propiedad en años")
45     Lat: float = Field(..., examples=[19.432608], description="La latitud de la propiedad")
46     Lng: float = Field(..., examples=[-99.133209], description="La longitud de la propiedad")
47     Municipality: str = Field(..., examples=["Benito Juárez"], description="El municipio donde se encuentra la propiedad")
48
49 # Definición de la respuesta del endpoint principal
50 class PrincipalResponse(BaseModel):
51     message: str = Field(..., examples=["Bienvenido a la API de predicción inmobiliaria"], description="Mensaje de bienvenida")
52     time: datetime.datetime = Field(..., examples=[datetime.datetime.now()], description="La hora actual del servidor")
53     version: str = Field(..., examples=["0.1"], description="La versión de la API")

```

```

53     description: str = Field(..., examples=["API para predecir el precio de
54         una propiedad"], description="Descripción de la API")
55     openapi: str = Field(..., examples=[f"{HOSTNAME}/openapi"], description=
56         "Enlace a la documentación de la API")
57     redoc: str = Field(..., examples=[f"{HOSTNAME}/redoc"], description="Enlace a la documentación de la API en formato ReDoc")
58
59 # Definición de la respuesta del endpoint de predicciones
60 class PredictResponse(BaseModel):
61     random_forest: float = Field(..., examples=[2500000.0], description="La
62         predicción del precio de la propiedad con el modelo Random Forest")
63     svm: float = Field(..., examples=[2700000.0], description="La predicción
64         del precio de la propiedad con el modelo SVM")
65     neural_network: float = Field(..., examples=[2600000.0], description="La
66         predicción del precio de la propiedad con el modelo de Redes Neuronales
67         ")
68
69 # Definición de la respuesta del endpoint de modelos
70 class ModelsResponse(BaseModel):
71     dataset: dict = Field(..., examples=[{"original": (1000,8), "training": {
72         "X": (1000, 8), "y": (1000, 1)}, "testing": {"X": (250, 8), "y": (250,
73         1)}}, description="Información sobre los datos de entrenamiento y
74         prueba")
75     models: dict = Field(..., examples=[{"random_forest": {"mse": 10000000000.0, "ci": (900000000.0, 1100000000.0), "mae": 30000.0, "r2": 0.9, "feature_importances": [0.1, 0.2, 0.3, 0.4, 0.0, 0.0, 0.0, 0.0], "max_features": "auto", "max_depth": 10, "n_estimators": 100, "oob_score": True}, "svm": {"mse": 15000000000.0, "ci": (14000000000.0, 16000000000.0), "mae": 40000.0, "r2": 0.8, "kernel": "rbf", "C": 1.0, "epsilon": 0.1}, "neural_network": {"mse": 20000000000.0, "ci": (19000000000.0, 21000000000.0), "mae": 50000.0, "r2": 0.7, "learning_rate": 0.001, "beta_1": 0.9, "beta_2": 0.999, "epsilon": 1e-07}}], description="Información sobre los modelos y sus características")

```

Ruta principal

La ruta principal de la API REST proporciona información general sobre el servicio, incluyendo la versión, la descripción, la hora actual del servidor y enlaces a la documentación de la API en formato OpenAPI y ReDoc. En el listado 5.9 se muestra el código relevante para la ruta principal.

Listing 5.9: Ruta principal

```

1 # Ruta principal
2 @app.get("/", summary="Ruta principal", description="Ruta principal de la
3     API de predicción inmobiliaria.", tags=["Principal"],
4     response_description="Mensaje de bienvenida", response_model=
5     PrincipalResponse)
6
7 async def principal():
8     # Regresamos un mensaje de bienvenida, la hora actual del servidor, la
9     # fecha, la versión del API, la descripción y un link a los docs de
10    # swagger.
11    return {"message": "Bienvenido a la API de SkyPrice",
12        "time": datetime.datetime.now(),
13        "version": "1.0.0",
14        "description": "API para predecir el precio de un departamento
15        en la Ciudad de México, si deseas ver la documentación de la API, visita
16        el enlace de OpenAPI (Swagger UI) o ReDoc:",
17

```

```

9         "openapi": f"{HOSTNAME}/openapi",
10        "redoc": f"{HOSTNAME}/redoc"

```

En la figura 5.4 se muestra la documentación de la API generada para la ruta principal utilizando Swagger UI:

Figura 5.4: Documentación de la API para la ruta principal.

Ruta de predicción

La ruta de predicción permite a los usuarios enviar datos sobre una propiedad y obtener una estimación del precio utilizando los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales. En el listado 5.10 se muestra el código relevante para la ruta de predicción.

Listing 5.10: Ruta de predicción

```

1 # Ruta para predecir el precio de una propiedad
2 @app.post("/predict", summary="Predecir el precio de una propiedad",
            description="Predecir el precio de una propiedad utilizando tres modelos
                        de aprendizaje automático: Random Forest, SVM y Redes Neuronales.",
            tags=["Predicciones"], response_description="Predicciones del precio de
                        la propiedad", response_model=PredictResponse)
3 async def predict(property: Property):
4     # Convertir entrada a DataFrame para preprocessamiento
5     input_data = [property.Municipality, property.Size_Terrain, property.
                    Size_Construction, property.Rooms, property.Bathrooms, property.Parking,
                    property.Age, property.Lat, property.Lng]
6     input_df = pd.DataFrame([input_data], columns=['Municipality', ,
                    'Size_Terrain', 'Size_Construction', 'Rooms', 'Bathrooms', 'Parking',
                    'Age', 'Lat', 'Lng'])
7

```

```

8     # Predecir con cada modelo
9     rf_pred,      = rf_model.predict(input_df)
10    svm_pred = svm_model.predict(input_df)[0]
11    # Preprocesar para la red neuronal y luego predecir
12    nn_input = preprocessor.transform(input_df)
13    nn_pred = nn_model.predict(nn_input)[0][0]
14
15    # Devolver las predicciones
16    return {
17        "random_forest": float(rf_pred),
18        "svm": float(svm_pred),
19        "neural_network": float(nn_pred)
20    }

```

En la figura 5.5 se muestra la documentación de la API generada para la ruta de predicción utilizando Swagger UI:

The screenshot shows the Swagger UI interface for a 'Predicciones' endpoint. At the top, there's a green button labeled 'POST /predict'. Below it, a brief description reads: 'Predecir el precio de una propiedad utilizando tres modelos de aprendizaje automático: Random Forest, SVM y Redes Neuronales.' Under the 'Parameters' section, it says 'No parameters'. In the 'Request body' section, the 'required' field is highlighted in red, and the media type is set to 'application/json'. A large black box obscures the example value. The 'Responses' section contains two entries: a 200 status code response with a description 'Predicciones del precio de la propiedad', a media type of 'application/json', and a link section labeled 'Links' (which is empty). Another black box obscures the schema. A 422 status code response is also listed under 'Responses' with a similar structure.

Figura 5.5: Documentación de la API para la ruta de predicción.

Ruta de gráficas de evaluación

La ruta de gráficas de evaluación permite a los usuarios visualizar las predicciones de los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales en comparación con los valores reales. En el listado 5.11 se muestra el código relevante para la ruta de gráficas de evaluación, destacando el uso de la función `FileResponse` para servir archivos estáticos.

Listing 5.11: Ruta de gráficas de evaluación

```
1 # Ruta para obtener las gráficas de predicciones vs valores reales
2 @app.get("/plots", response_class=FileResponse, summary="Obtener gráficas de
    predicciones vs valores reales", description="Obtener las gráficas de
    predicciones vs valores reales de los modelos Random Forest, SVM y Redes
    Neuronales.", tags=["Gráficas"], response_description="Gráficas de
    predicciones vs valores reales")
3 async def plots():
4     return FileResponse('plots.png')
```

En la figura 5.6 se muestra la documentación de la API generada para la ruta de gráficas de evaluación utilizando Swagger UI:

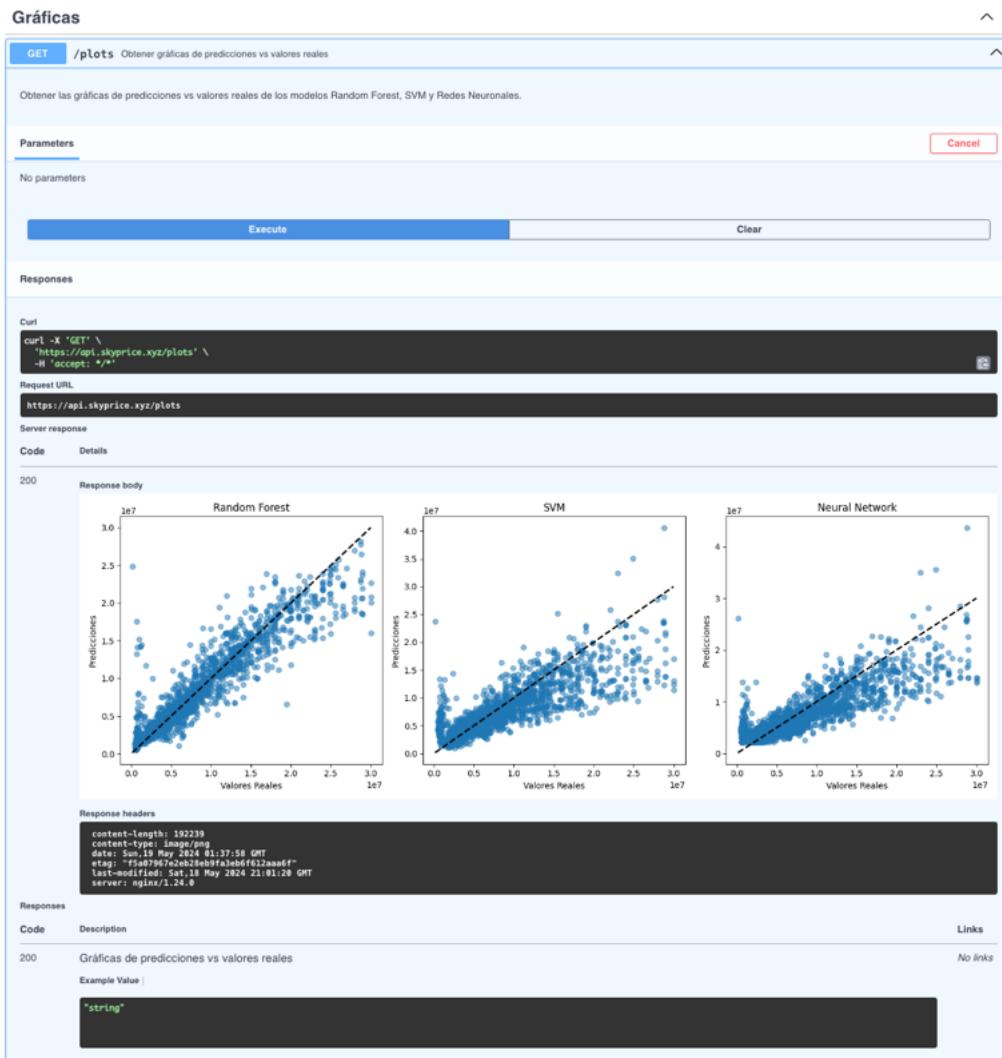


Figura 5.6: Documentación de la API para la ruta de gráficas de evaluación.

Ruta de información de modelos

La ruta de información de modelos permite a los usuarios obtener información detallada sobre los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales, incluyendo métricas de evaluación, hiperparámetros y características importantes. En el listado 5.12 se muestra el código relevante para la ruta de información de modelos.

Listing 5.12: Ruta de información de modelos

```
1 # Ruta para listar los modelos y sus características (ajuste de datos de
   entrenamiento, hiperparámetros, etc.)
```

```

2  @app.get("/models", summary="Obtener información sobre los modelos y sus
   características", description="Obtener información sobre los modelos de
   aprendizaje automático utilizados en la API, incluyendo sus caracterí-
   sticas, ajuste de datos de entrenamiento, hiperparámetros, etc.", tags=[

3  "Modelos"], response_description="Información sobre los modelos y sus
   características", response_model=ModelsResponse)
4
5  async def models_info():
6      # Regresar información sobre los modelos y sus características
7      return {
8          "dataset": {
9              "original": df.shape,
10             "training": {
11                 "X": X_train.shape,
12                 "y": y_train.shape
13             },
14             "testing": {
15                 "X": X_test.shape,
16                 "y": y_test.shape
17             }
18         },
19         "models": {
20             "random_forest": {
21                 "mse": rf_mse,
22                 "rmse": rf_rmse,
23                 "ci": rf_ci,
24                 "mae": rf_mae,
25                 "r2": rf_r2 ,
26                 "feature_importances": rf_model['regressor'].
27                     feature_importances_.tolist(),
28                     "max_features": rf_model['regressor'].max_features,
29                     "max_depth": rf_model['regressor'].max_depth,
30                     "n_estimators": rf_model['regressor'].n_estimators,
31                     "oob_score": rf_model['regressor'].oob_score,
32             },
33             "svm": {
34                 "mse": svm_mse,
35                 "rmse": svm_rmse,
36                 "ci": svm_ci,
37                 "mae": svm_mae,
38                 "r2": svm_r2 ,
39                 "kernel": svm_model['svr'].kernel,
40                 "C": svm_model['svr'].C,
41                 "epsilon": svm_model['svr'].epsilon
42             },
43             "neural_network": {
44                 "mse": nn_mse,
45                 "rmse": nn_rmse,
46                 "ci": nn_ci,
47                 "mae": nn_mae,
48                 "r2": nn_r2 ,
49                 "learning_rate": float(nn_model.optimizer.learning_rate.
50                     numpy()),
51                 "beta_1": nn_model.optimizer.beta_1,
52                 "beta_2": nn_model.optimizer.beta_2,
53                 "epsilon": nn_model.optimizer.epsilon
54             }
55         }
56     }
57 }
```

En la figura 5.7 se muestra la documentación de la API generada para la

ruta de información de modelos utilizando Swagger UI:

The screenshot shows the Swagger UI interface for a 'Models' endpoint. At the top, there is a blue header bar with the text 'GET /models'. Below this, a main content area has a title 'Obtener información sobre los modelos de aprendizaje automático utilizados en la API, incluyendo sus características, ajuste de datos de entrenamiento, hiperparámetros, etc.' A 'Try it out' button is located in the top right corner of this section. The interface is divided into sections: 'Parameters' (which says 'No parameters'), 'Responses', and a detailed table for the 200 status code. The table columns are 'Code', 'Description', and 'Links'. The 'Description' column for code 200 states 'Información sobre los modelos y sus características'. The 'Media type' dropdown is set to 'application/json'. The 'Example Value' section shows a JSON schema for a dataset, which is partially visible as a large black rectangular redaction box.

Figura 5.7: Documentación de la API para la ruta de información de modelos.

5.3.4. Documentación

FastAPI permite documentar la API de manera automática utilizando la información proporcionada en las rutas y modelos definidos. La documentación se genera en formato OpenAPI (Swagger) y ReDoc, lo que facilita la comprensión y el uso de la API.

Esta documentación no solo es útil para el desarrollo de la interfaz gráfica en el actual trabajo terminal, sino que también es útil para otros desarrolladores que deseen integrar la API en sus propios proyectos.

OpenAPI (Swagger)

La documentación de la API en formato OpenAPI (Swagger) permite a los usuarios explorar los endpoints, los modelos y los parámetros de la API de manera interactiva. En la figura 5.8 se muestra la documentación de la API en formato OpenAPI, la cual incluye información detallada sobre los endpoints y los modelos utilizados en la API.

Figura 5.8: Documentación de la API en formato OpenAPI (Swagger).

ReDoc

La documentación de la API en formato ReDoc permite a los usuarios explorar los endpoints, los modelos y los parámetros de la API de manera interactiva y visual, además de proveer una interfaz de documentación más editorial y amigable. En la figura 5.9 se muestra la documentación de la API en formato ReDoc.

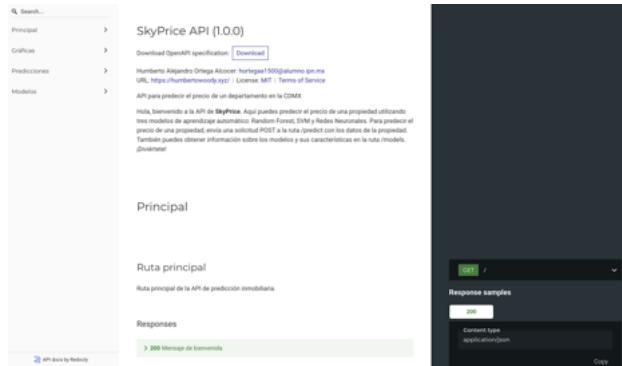


Figura 5.9: Documentación de la API en formato ReDoc.

5.4. Interfaz gráfica

La interfaz gráfica se implementó utilizando React con NextJS como marco de trabajo y Material-UI como biblioteca de componentes. La interfaz gráfica permite a los usuarios interactuar con los modelos de aprendizaje automático, obtener información sobre los modelos y realizar predicciones de precio para propiedades en la Ciudad de México, además se incluye un mapa interactivo con la ubicación de las propiedades y otras capas de datos relevantes.

5.4.1. Estructura del proyecto

En la figura 5.10 se muestra la estructura del proyecto de la interfaz gráfica, la cual se divide en diferentes carpetas y archivos para facilitar el desarrollo y el mantenimiento del código.

```
> pwd  
/Users/humbertowoodly/Proyectos/IPN/escom-tt/prototipo/frontend  
> tree .  
. |  
+-- README.md  
+-- next-env.d.ts  
+-- next.config.mjs  
+-- package-lock.json  
+-- package.json  
+-- public  
|   +-- aerial-cdmx.jpg  
|   +-- android-chrome-192x192.png  
|   +-- android-chrome-384x384.png  
|   +-- browserconfig.xml  
|   +-- departamento-isometrico.svg  
|   +-- kepler-bg.png  
|   +-- kepler.html  
|   +-- logo-escom.png  
|   +-- logo-ipn.png  
|   +-- mstile-150x150.png  
|   +-- redoc-logo.png  
|   +-- reforma-sm.jpg  
|   +-- reforma.jpg  
|   +-- safari-pinned-tab.svg  
|   +-- santafe.jpg  
|   +-- skyline-cdmx-2.jpeg  
|   +-- skyline-cdmx-sm.jpg  
|   +-- skyline-cdmx-xs.jpg  
|   +-- swagger-logo.png  
+-- src  
    +-- app  
    |   +-- acerca-de  
    |   |   +-- page.tsx  
    |   |   +-- apple-icon.png  
    |   |   +-- favicon.ico  
    |   |   +-- i18nContext.tsx  
    |   |   +-- icon1.png  
    |   |   +-- icon2.png  
    |   |   +-- layout.tsx  
    |   |   +-- manifest.json  
    |   +-- mapa  
    |   |   +-- page.tsx  
    |   |   +-- page.tsx  
    |   |   +-- robots.txt  
    |   |   +-- sitemap.xml  
    +-- components  
    |   +-- Creditos.tsx  
    |   +-- CurrencyConverter.tsx  
    |   +-- Datos.tsx  
    |   +-- Footer.tsx  
    |   +-- FormularioPrediccion.tsx  
    |   +-- LanguageSelector.tsx  
    |   +-- ModelComparison.tsx  
    |   +-- Modelos.tsx  
    |   +-- NavBar.tsx  
    +-- locales  
    |   +-- en.json  
    |   +-- es.json  
    |   +-- fr.json  
    |   +-- pt.json  
    |   +-- theme.ts  
    +-- types  
        +-- estadisticas.ts 142  
    tsconfig.json  
  
9 directories, 52 files
```

Figura 5.10: Estructura del proyecto de la interfaz gráfica.

5.4.2. Navegación y traducciones

La interfaz gráfica incluye un menú de navegación con enlaces a las diferentes secciones de la página, como la sección de estimación de precio, la sección de acerca de y el mapa interactivo. Además, se incluyó soporte para traducciones en inglés, español, francés y portugués utilizando la biblioteca i18next. En la estructura mostrada en la figura 5.10 se puede observar la carpeta `src/locales` que contiene los archivos de traducción para cada idioma.

En la figura 5.11 se muestra la interfaz gráfica en francés, la cual incluye traducciones para los textos y los mensajes de la página.

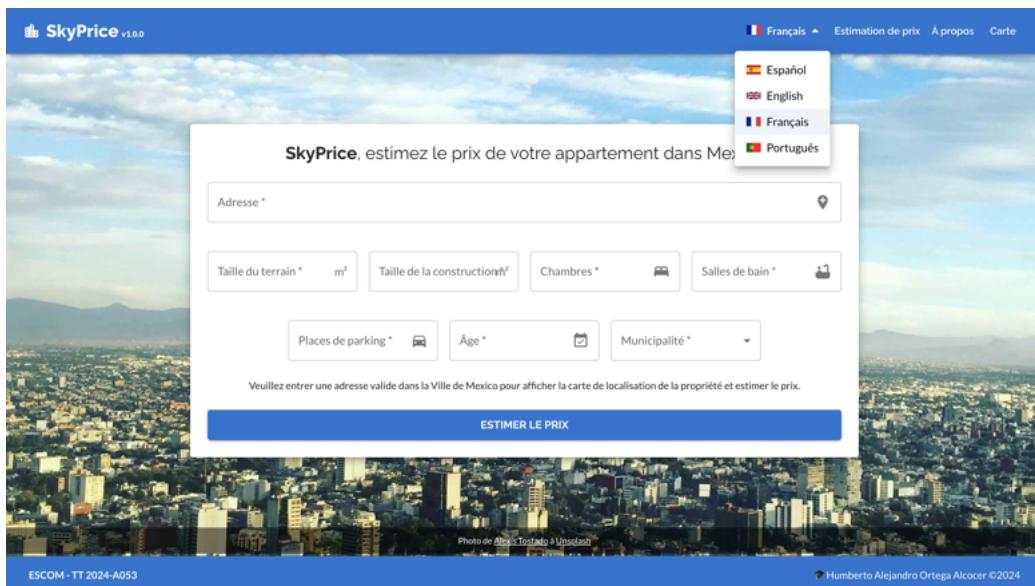


Figura 5.11: Interfaz gráfica en francés.

Traducción de textos

La traducción de textos se implementó utilizando la biblioteca i18next, la cual permite definir archivos de traducción para cada idioma y utilizarlos en la interfaz gráfica. En el listado 5.13 se muestra el código relevante para la traducción de textos en la interfaz gráfica en una de las secciones.

Listing 5.13: Traducción de textos

```
1 const [lang, setLang] = useState<string>(defLang);
2 const [translations, setTranslations] = useState({ ...defTranslations });
3
4 useEffect(() => {
5   import(`@/locales/${lang}.json`)
6     .then((module) => {
```

```

7     setTranslations(module.default);
8     localStorage.setItem('i18nextLng', lang);
9     console.info(`Traducciones cargadas para el idioma ${lang}`);
10    })
11   .catch(() => {
12     console.error(
13       `Archivo de traducciones no encontrado para el idioma ${lang}`,
14     );
15   });
16 }, [lang]);

```

Un ejemplo de archivo de traducción en inglés se muestra en el listado 5.14, el cual muestra un fragmento de un archivo de traducción en inglés con los textos utilizados en la interfaz gráfica.

Listing 5.14: Archivo de traducción en inglés

```

1  {
2   "navbar": {
3     "home": "Home",
4     "prediction": "Prediction",
5     "about": "About",
6     "map": "Map"
7   },
8   "predictionForm": {
9     "title": "Property Price Prediction",
10    "subtitle": "Enter the property details to get an estimate of the price"
11   ,
12   "sizeTerrain": "Terrain Size (m2)",
13   "sizeConstruction": "Construction Size (m2)",
14   "rooms": "Rooms",
15   "bathrooms": "Bathrooms",
16   "parking": "Parking",
17   "age": "Age",
18   "lat": "Latitude",
19   "lng": "Longitude",
20   "municipality": "Municipality",
21   "submit": "Predict"
22 }

```

5.4.3. Estimación de precio

La página principal de la interfaz gráfica incluye una sección donde los usuarios pueden ingresar datos sobre una propiedad y obtener una estimación del precio. Esta funcionalidad consiste en un formulario con campos para los datos de la propiedad, validación para dichos campos, y un botón para realizar la predicción. En la figura 5.12 se muestra la sección de estimación de precio de la interfaz gráfica.

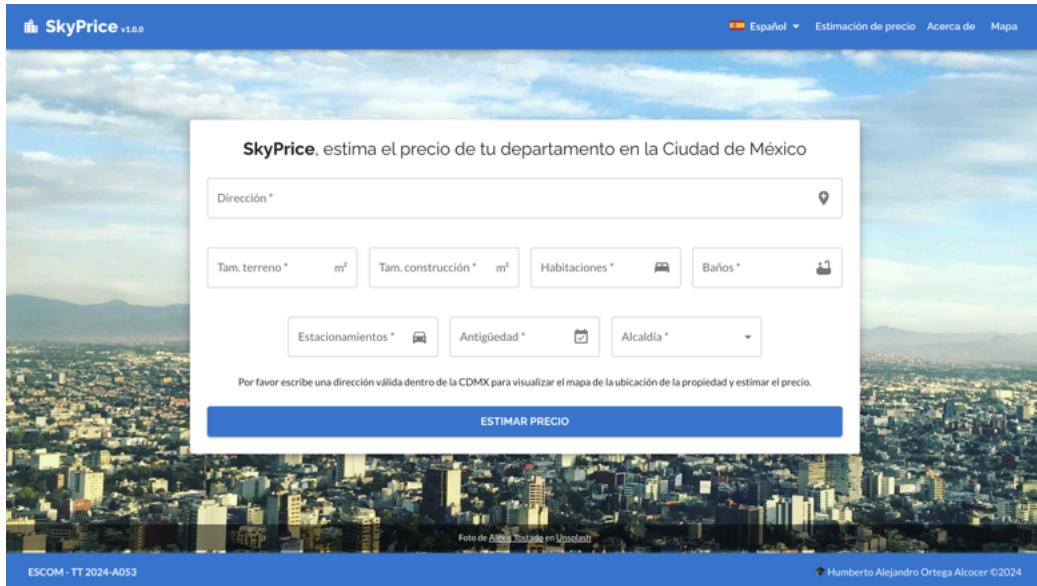


Figura 5.12: Sección de estimación de precio de la interfaz gráfica.

Validación de campos

La validación de campos se implementó utilizando la biblioteca Yup, la cual permite definir esquemas de validación para los campos del formulario. En el listado 5.15 se muestra el código relevante para la validación de campos en la sección de estimación de precio.

Listing 5.15: Validación de campos

```

1 // Esquema de validación para el formulario
2 const validationSchema = yup.object({
3     Size_Terrain: yup
4         .number()
5         .required(t('predictionForm.validations.sizeTerrainRequired'))
6         .min(10, t('predictionForm.validations.sizeTerrainMin', { min: 10 }))
7         .max(1000, t('predictionForm.validations.sizeTerrainMax', { max: 1000
})),
8     Size_Construction: yup
9         .number()
10        .required(t('predictionForm.validations.sizeConstructionRequired'))
11        .min(10, t('predictionForm.validations.sizeConstructionMin', { min: 10
}))
12        .max(
13            1000,
14            t('predictionForm.validations.sizeConstructionMax', { max: 1000 }), )
15        ),
16     Rooms: yup
17         .number()
18         .required(t('predictionForm.validations.roomsRequired'))
19         .integer()
20         .min(0, t('predictionForm.validations.roomsMin', { min: 0 })) )
21         .max(10, t('predictionForm.validations.roomsMax', { max: 10 })),
```

```

22     // Validar que los baños sean un número en pasos de 0.5 y mayor a 1 (mínimo 1 baño)
23     Bathrooms: yup
24         .number()
25         .required(t('predictionForm.validations.bathroomsRequired'))
26         .min(1, t('predictionForm.validations.bathroomsMin', { min: 1 }))
27         .max(10, t('predictionForm.validations.bathroomsMax', { max: 10 }))
28         .test(
29             'is-half',
30             t('predictionForm.validations.bathroomsHalf'),
31             (value) => {
32                 if (value) {
33                     return value % 0.5 === 0;
34                 }
35                 return true;
36             },
37         ),
38     Parking: yup
39         .number()
40         .required(t('predictionForm.validations.parkingRequired'))
41         .integer()
42         .min(0, t('predictionForm.validations.parkingMin', { min: 0 }))
43         .max(5, t('predictionForm.validations.parkingMax', { max: 5 })),
44     Age: yup
45         .number()
46         .required(t('predictionForm.validations.ageRequired'))
47         .integer()
48         .min(0, t('predictionForm.validations.ageMin', { min: 0 }))
49         .max(100, t('predictionForm.validations.ageMax', { max: 100 })),
50     Lat: yup
51         .number()
52         .required(t('predictionForm.validations.latRequired'))
53         .min(19.1, t('predictionForm.validations.outOfBounds'))
54         .max(19.7, t('predictionForm.validations.outOfBounds')),
55     Lng: yup
56         .number()
57         .required(t('predictionForm.validations.lngRequired'))
58         .min(-99.4, t('predictionForm.validations.outOfBounds'))
59         .max(-98.9, t('predictionForm.validations.outOfBounds')),
60     Municipality: yup
61         .string()
62         .required(t('predictionForm.validations.municipalityRequired'))
63         .oneOf(alcaldias, t('predictionForm.validations.municipalityInvalid')),
64     );

```

Envío de datos

El envío de datos se implementó utilizando `fetch` para realizar una petición POST a la API REST con los datos de la propiedad. En el listado 5.16 se muestra el código relevante para el envío de datos en la sección de estimación de precio.

Listing 5.16: Envío de datos

```

1 // Función para enviar los datos de la propiedad a la API
2 const handleSubmit = async (values) => {
3     try {

```

```

4   // Realizar la petición POST a la API REST
5   const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/predict
6   ', {
7     method: 'POST',
8     headers: {
9       'Content-Type': 'application/json',
10    },
11    body: JSON.stringify(values),
12  });
13
14  // Verificar si la petición fue exitosa
15  if (response.ok) {
16    // Obtener los datos de la respuesta
17    const data = await response.json();
18    // Actualizar el estado con los datos de la predicción
19    setPrediction(data);
20  } else {
21    // Mostrar un mensaje de error si la petición falla
22    throw new Error('Error al realizar la predicción');
23  }
24  // Mostrar un mensaje de error si la petición falla
25  catch (error) {
26    // Mostrar un mensaje de error si la petición falla
27    console.error(error);
28  }
29};

```

Geocodificación

Para realizar la geocodificación de la dirección proporcionada por el usuario, se utilizó Google Maps API para obtener las coordenadas geográficas (latitud y longitud) de la dirección. En el listado 5.17 se muestra el código relevante para la geocodificación en la sección de estimación de precio.

Listing 5.17: Geocodificación

```

1  const onPlaceChanged = () => {
2    if (autocomplete !== null) {
3      // @ts-ignore
4      const place = autocomplete.getPlace();
5      if (!place) {
6        console.warn('No se pudo obtener la información del lugar');
7        return;
8      }
9
10     // Extraer latitud y longitud del lugar
11     const lat = place?.geometry?.location?.lat();
12     const lng = place?.geometry?.location?.lng();
13
14     // Extrae el municipio del componente de dirección de place
15     const municipality = encontrarMunicipio(
16       place.address_components,
17       place.formatted_address,
18     );
19
20     // Actualiza el estado del formulario con estos valores
21     formik.setFieldValue('Lat', lat);
22     formik.setFieldValue('Lng', lng);
23     formik.setFieldValue('Municipality', municipality);

```

```

24
25     // Actualizar el estado de la Dirección
26     setAddress(place.formatted_address);
27 } else {
28     console.log('Autocomplete aún no está cargado');
29 }
30 };

```

Una vez que el usuario ha introducido una dirección y se ha realizado la geocodificación, se extrae el municipio de la dirección proporcionada (alcaldía) y se muestra un mapa interactivo con la ubicación de la propiedad. En la figura 5.13 se muestra el formulario con datos de una propiedad así como el mapa resultante de la geocodificación.

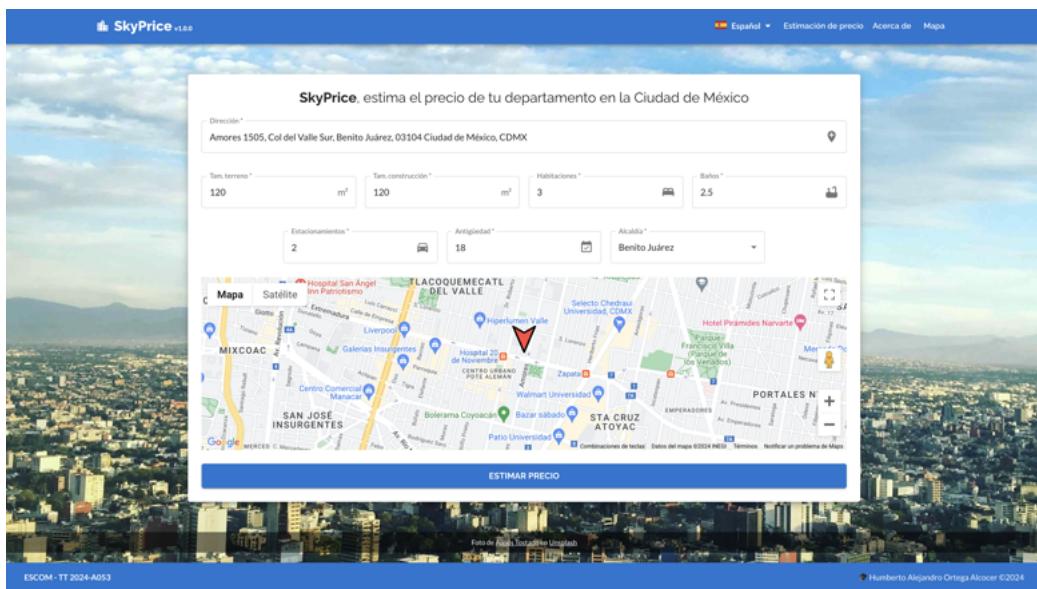


Figura 5.13: Formulario con datos de una propiedad y mapa resultante de la geocodificación.

Resultados de la predicción

Los resultados de la predicción se muestran en una sección debajo del formulario de estimación de precio, donde los usuarios pueden ver las predicciones de los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales. En la figura 5.14 se muestra la sección de resultados de la predicción de la interfaz gráfica.



Figura 5.14: Sección de resultados de la predicción de la interfaz gráfica.

Como se puede apreciar en la figura 5.14, los resultados de la predicción incluyen una estimación de precio de venta por modelo, así como estimaciones de precios de renta mensual calculada al 4 % - 6 % del precio de venta.

Adicionalmente, cada estimación de precios incluye un convertidor de moneda que permite al usuario ver el precio en dólares estadounidenses (USD), euros (EUR), dólares canadienses (CAD) entre otros más. En la figura 5.15 se muestran las opciones de conversión de moneda disponibles.

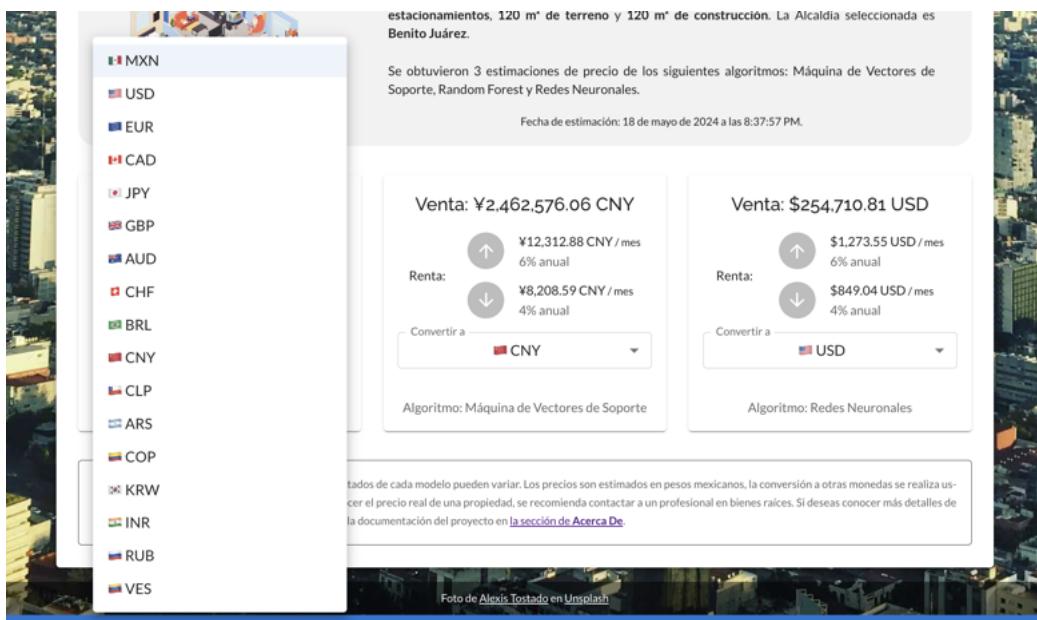


Figura 5.15: Opciones de conversión de moneda en la interfaz gráfica.

5.4.4. Acerca de

La página de Acerca de incluye información sobre el proyecto, los modelos de aprendizaje automático utilizados, información sobre los datos y enlaces a la documentación de la API.

Proyecto en general

La primer sección de la página de Acerca de incluye información general sobre el proyecto, incluyendo el título, la descripción, el autor y la fecha de creación. En la figura 5.16 se muestra la sección de información general de la página de Acerca de.

SkyPrice v1.0.0 Espanol ▾ Estimación de precio Acerca de Mapa

SkyPrice, la plataforma de estimación de precios de departamentos en la Ciudad de México

Paseo de la Reforma, Ciudad de México
Foto de [carlos aranda](#) en Unsplash

¿Qué es **SkyPrice**?
SkyPrice es una plataforma diseñada para analizar y estimar los precios de propiedades en la Ciudad de México, basándose en un detallado conjunto de datos locales. Utilizando técnicas de aprendizaje automático como Random Forest, Máquina de Vectores de Soporte (SVM) y Redes Neuronales, ofrece estimaciones precisas del valor de mercado de departamentos según sus características específicas. Adicionalmente, cuenta con una API pública que facilita la integración de sus funcionalidades en otras aplicaciones, ampliando así su aplicabilidad y alcance.

Estima el precio de tu departamento ahora!

Figura 5.16: Sección de información general de la página de Acerca de.

Datos

La segunda sección de la página de Acerca de incluye información sobre los datos utilizados en el proyecto, incluyendo el tamaño del conjunto de datos, la cantidad de características y la división en conjuntos de entrenamiento y prueba. En la figura 5.17 se muestra la sección de información sobre los datos de la página de Acerca de.

SkyPrice v1.0.0 Espanol ▾ Estimación de precio Acerca de Mapa

Datos

Para el entrenamiento de los modelos se utilizaron conjuntos de datos obtenidos de plataformas de venta de propiedades en la Ciudad de México. Los datos fueron preprocesados y limpiados para obtener un conjunto de datos homogéneo y sin valores nulos.

Los datos se dividieron en dos conjuntos: uno de entrenamiento y otro de prueba. El conjunto de entrenamiento se utilizó para entrenar los modelos, mientras que el conjunto de prueba se utilizó para evaluar el desempeño de los modelos.

Vista aérea de la Ciudad de México
Foto de [Andrea Leopoldi](#) en Unsplash

	Datos "limpios"	Datos de entrenamiento	Datos de prueba
	14893 registros	11169 registros	3724 registros
Esto equivale al 45.80% del total de datos disponibles (32520 registros).			
Esto equivale al 74.99% de los datos (entrenamiento y prueba).			
Esto equivale al 25.01% de los datos (entrenamiento y prueba).			

Figura 5.17: Sección de información sobre los datos de la página de Acerca de.

También se incluye una pequeña sección sobre el mapa interactivo, que permite a los usuarios visualizar los datos en cuestión geoespacialmente usando Kepler.gl. En la figura 5.18 se muestra la sección de información sobre el mapa interactivo de la página de Acerca de.



Figura 5.18: Sección de información sobre el mapa interactivo de la página de Acerca de.

Modelos de aprendizaje automático

La siguiente sección de la página de Acerca de incluye información sobre los modelos de aprendizaje automático utilizados en el proyecto, incluyendo Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales. En la figura 5.19 se muestra la sección de información sobre los modelos de aprendizaje automático, que incluye métricas de evaluación, hiperparámetros y características importantes.

SkyPrice v1.0.0

Español ▾ Estimación de precio Acerca de Mapa

Modelos

Ahora hablaremos de los algoritmos de aprendizaje automático que utiliza SkyPrice. Aquí, se presentan estadísticas de cada modelo utilizado para la estimación de precios de departamentos en la Ciudad de México. Los modelos, que incluyen Random Forest, SVM y Redes Neuronales, han sido meticulosamente configurados y entrenados con una variedad de hiperparámetros para optimizar su rendimiento en la predicción de precios.

Esta sección desglosa los resultados obtenidos, incluyendo el Error Absoluto Medio (MAE), el coeficiente de determinación (R^2) y el Error Cuadrático Medio (MSAE), junto con gráficas de ajuste que comparan las predicciones con los datos de entrenamiento, proporcionando una visión clara de la eficacia y la precisión de cada modelo.



Parque La Mexicana, Santa Fe, Ciudad de México
Foto de Oscar Revo en Unsplash

Random Forest

Random Forest es un modelo compuesto por múltiples árboles de decisión y es efectivo para prevenir el sobreajuste. Con un RMSE de 1653462.66 y un coeficiente R2 del 90.39%, este modelo destaca por su precisión predictiva. Se configuró con 1000 árboles y una profundidad máxima de 30. La puntuación QOB indica su desempeño sin validación cruzada adicional.

[APRENDER MÁS...](#)

Máquina de Vectores de Soporte

El SVM utiliza un kernel lineal para manejar relaciones no lineales. Con un valor de R2 de 69.72%, el modelo se ha configurado con un parámetro C de 1000 y un epsilon de 0.01, ajustando el margen de error y la frontera de decisión.

[APRENDER MÁS...](#)

Red Neuronal

La Red Neuronal utiliza un ritmo de aprendizaje de 0.10000 y parámetros beta1 y beta2 ajustados a 0.90 y 1.00. El RMSE alcanzado es de 2602830.22, evidenciando su capacidad predictiva en la estimación de precios.

[APRENDER MÁS...](#)

Figura 5.19: Sección de información sobre los modelos de aprendizaje automático de la página de Acerca de.

Además, se incluyen dos gráficas que muestran las predicciones de los modelos de Random Forest, Máquinas de Soporte Vectorial y Redes Neuronales en comparación con los valores reales y una comparativa de las métricas de evaluación de los modelos. En la figura 5.20 se muestra la sección de gráficas de la página de Acerca de.

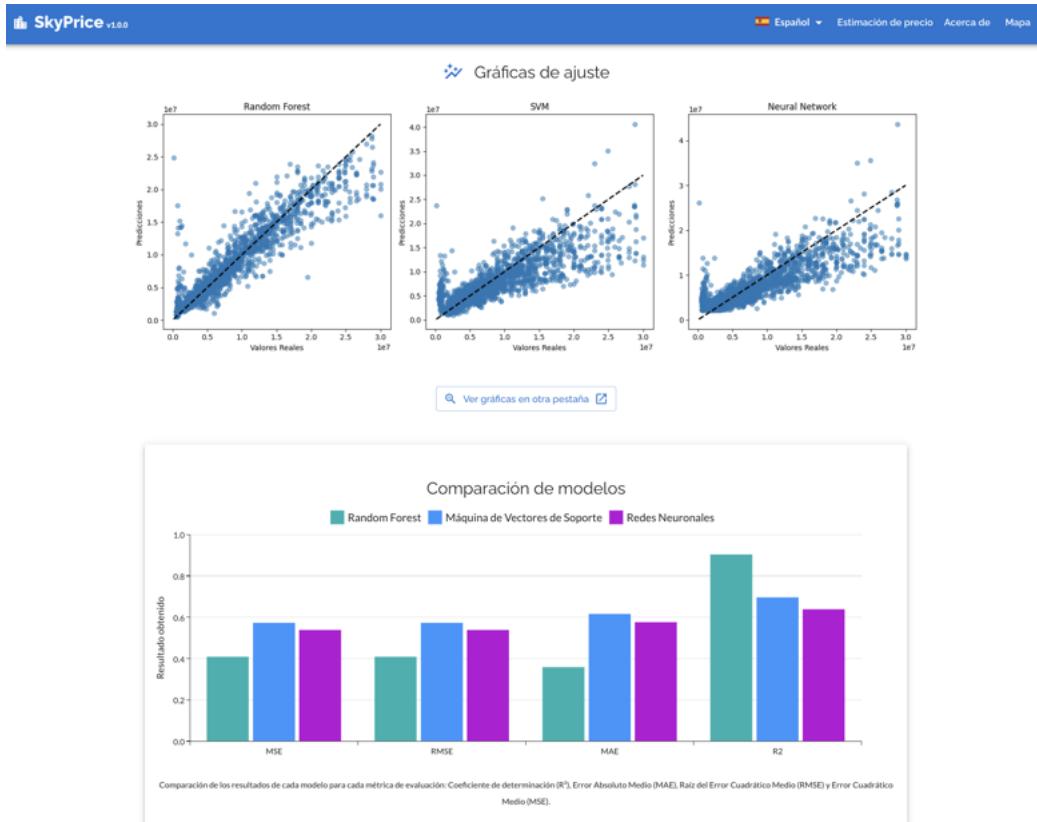


Figura 5.20: Sección de gráficas de la página de Acerca de.

Documentación de la API y créditos

La penúltima sección de la página de Acerca de incluye enlaces a la documentación de la API en formato OpenAPI (Swagger) y ReDoc, lo que permite a los usuarios explorar los endpoints, los modelos y los parámetros de la API de manera interactiva y visual. Además, se incluyen los créditos del proyecto. En la figura 5.21 se muestra la sección de documentación de la página de Acerca de.

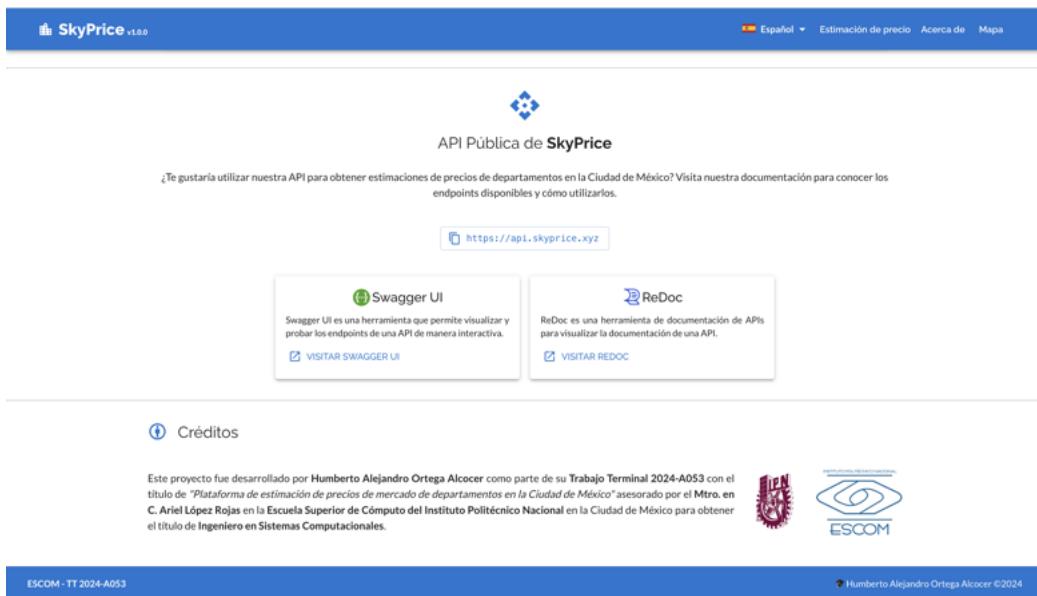


Figura 5.21: Sección de documentación de la página de Acerca de.

5.4.5. Mapa interactivo

La página de Mapa interactivo se desarrolló utilizando Kepler.gl, una biblioteca de código abierto para la visualización de datos geoespaciales. La página permite a los usuarios visualizar los datos de propiedades en la Ciudad de México empleados en el proyecto, además de otras capas de datos relevantes.

Dado que los datos contenidos en las capas puede ser bastante extenso, se incluyó una alerta que informa al usuario que la carga de los datos puede consumir datos móviles, por lo que puede elegir si desea cargar el mapa dinámico o no, en cuyo caso se mostrará una imagen estática del mapa. En la figura 5.22 se muestra la página de Mapa inicialmente con la alerta de carga de datos.

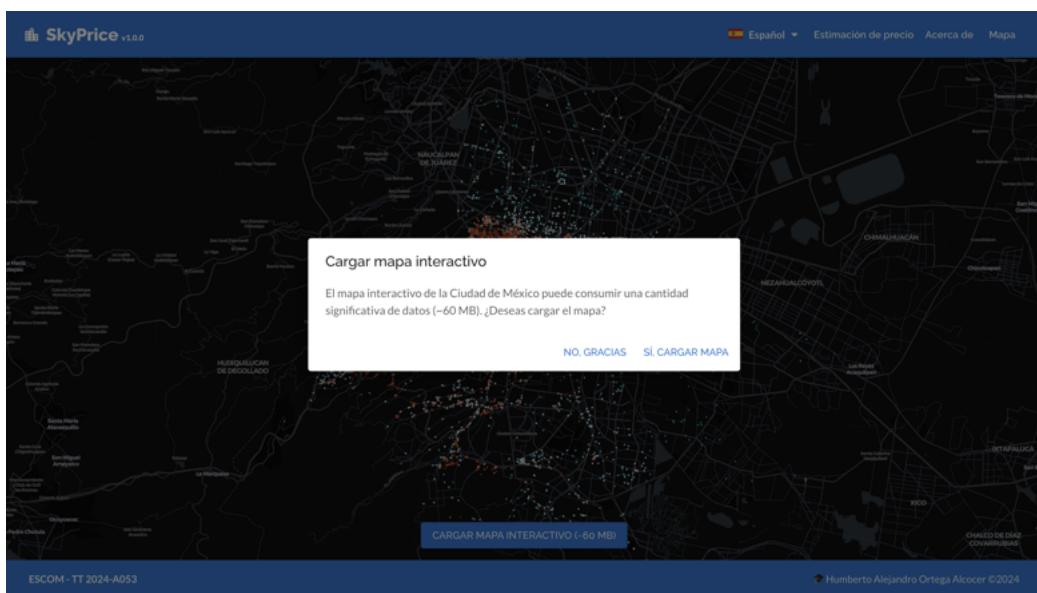


Figura 5.22: Página de Mapa interactivo con alerta de carga de datos.

Si el usuario decide no cargar el mapa dinámico, se muestra la imagen presente en la figura 5.22 y se muestra un botón para cargar el mapa dinámico. En caso de que el usuario decida cargar el mapa dinámico, se muestra el mapa interactivo con las capas de datos correspondientes. En la figura 5.23 se muestra el mapa interactivo con la capa de datos de propiedades en la Ciudad de México.

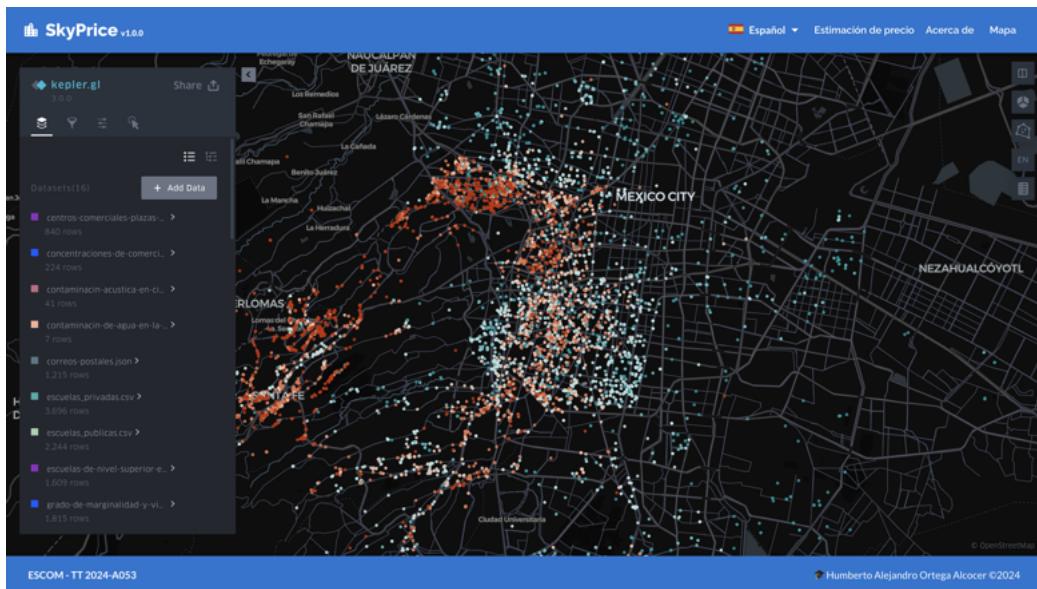


Figura 5.23: Página de Mapa interactivo con capa de datos de propiedades en la Ciudad de México.

Kepler.gl permite a los usuarios interactuar con el mapa, incluyendo la capacidad de hacer zoom, desplazarse, cambiar la perspectiva, activar y desactivar capas de datos, entre otras funcionalidades [74]. La intención de proveer esta herramienta junto con capas de datos relevantes es que los usuarios puedan explorar y visualizar los datos y obtener información valiosa sobre las propiedades en la Ciudad de México.

En el cuadro 5.1 se muestra un resumen de las capas de datos disponibles en el mapa interactivo, incluyendo la fuente de los datos y una breve descripción de cada capa.

Capa de datos	Descripción
Departamentos	Conjunto de datos del proyecto
Centros y plazas comerciales y tiendas de autoservicios [96]	Centros y plazas comerciales y tiendas de autoservicios en la Ciudad de México
Concentraciones comerciantes [97]	Concentraciones de comerciantes en la Ciudad de México
Contaminación acústica [98]	Contaminación acústica en la Ciudad de México
Contaminación del agua [99]	Contaminación del agua en la Ciudad de México
Códigos postales [100]	Códigos postales en la Ciudad de México
Escuelas privadas [101]	Escuelas privadas en la Ciudad de México
Escuelas públicas [102]	Escuelas públicas en la Ciudad de México
Escuelas de nivel superior [103]	Escuelas de nivel superior en la Zona Metropolitana del Valle de México
Grado de marginalidad y violencia [104]	Grado de marginalidad y violencia en la Ciudad de México
Hospitales y centros de salud [105]	Hospitales y centros de salud en la Ciudad de México
Alcaldías [106]	Alcaldías en la Ciudad de México
Mercados [107]	Mercados públicos en la Ciudad de México
Estaciones de Metro [108]	Estaciones de Metro en la Ciudad de México
Epicentros de sismos [109]	Epicentros de sismos en la Ciudad de México
Tianguis [110]	Tianguis en la Ciudad de México

Cuadro 5.1: Capas de datos disponibles en el mapa interactivo.

Estos conjuntos de datos no son definitivos, una de las ventajas de Kepler.gl es que permite a los usuarios cargar sus propios datos y visualizarlos en el mapa interactivo, lo que facilita la exploración y el análisis de datos geoespaciales. A su vez, la capacidad de Kepler.gl para exportar datos y visualizaciones permite a los usuarios compartir y colaborar en proyectos de datos geoespaciales.

Para demostrar una posibilidad de uso, en la figura 5.24 se muestra el mapa interactivo con la capa de datos de propiedades en la Ciudad de México, la capa con las Alcaldías y la capa de contaminación acústica, que podrían

ser útiles para analizar la relación entre el precio de las propiedades y la contaminación acústica en la Ciudad de México.

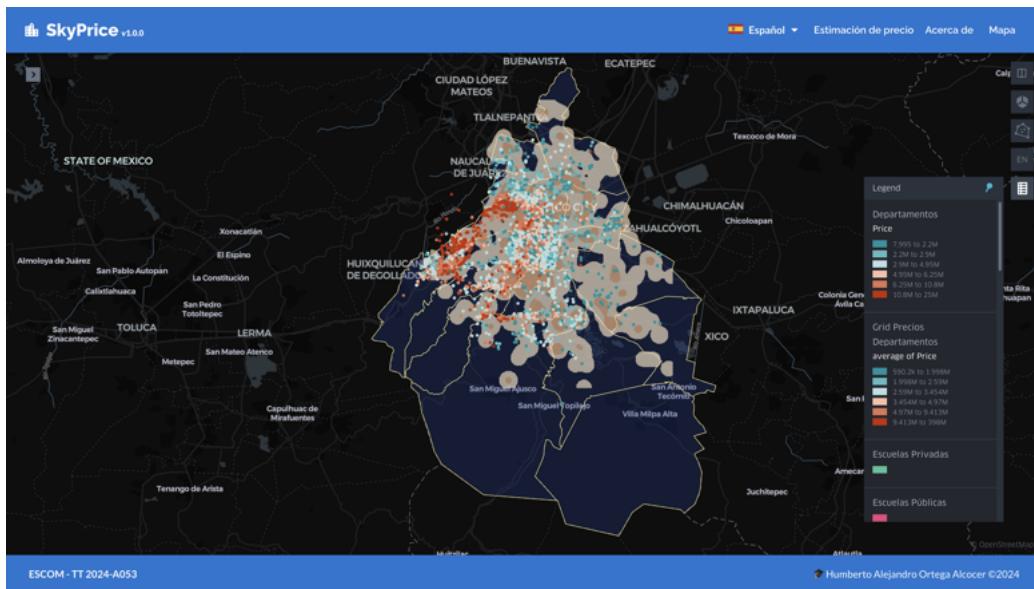


Figura 5.24: Mapa interactivo con capa de datos de propiedades, alcaldías y contaminación acústica.

5.4.6. Diseño responsivo

A bien de garantizar una experiencia de usuario óptima en dispositivos móviles y tabletas, se implementó un diseño responsive en la interfaz gráfica. El diseño responsive se logró utilizando Media Queries de CSS y la biblioteca Material-UI, la cual proporciona componentes y estilos responsivos para aplicaciones web.

En la Figura 5.25 se muestra la pantalla principal de la interfaz gráfica en diseño móvil, con el menú de hamburguesa desplegado mostrando así las opciones de navegación. En la Figura 5.26 se muestra la pantalla de resultados de la interfaz gráfica en diseño móvil, con las predicciones de los modelos de aprendizaje automático.



Figura 5.25: Pantalla principal en diseño móvil



Figura 5.26: Pantalla de resultados en diseño móvil

En la Figura 5.27 se muestra la pantalla de Acerca de en diseño móvil, con la gráfica de comparación de métricas de evaluación de los modelos de aprendizaje automático. En la Figura 5.28 se muestra la pantalla de Mapa interactivo en diseño móvil, con la capa de datos de propiedades en la Ciudad de México.

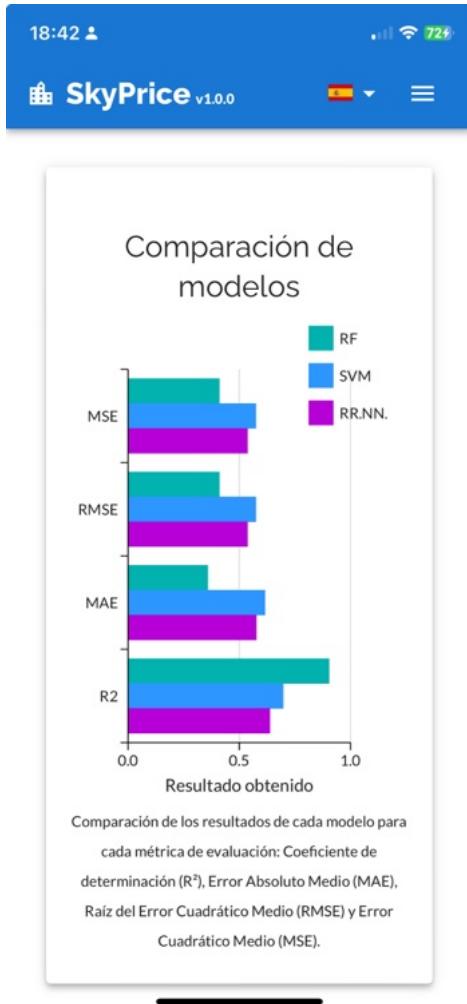


Figura 5.27: Pantalla de Acerca de en diseño móvil

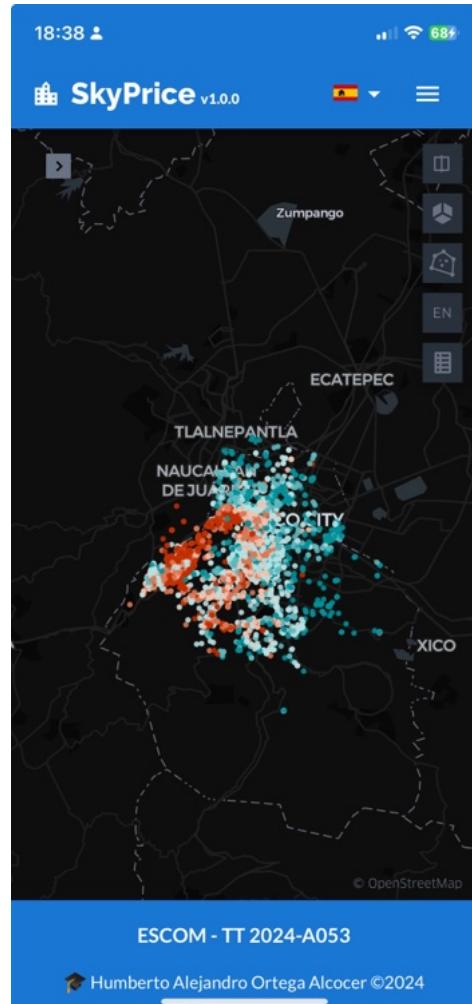


Figura 5.28: Pantalla de Mapa interactivo en diseño móvil

5.5. Despliegue en la nube

El despliegue en la nube del servicio web y la interfaz gráfica se llevó a cabo utilizando Amazon Web Services (AWS), para esto se utilizó una cuenta de AWS Educate que proporciona créditos y recursos para estudiantes y educadores [111].

Lo primero que se realizó fue la adquisición de un dominio para la aplicación, posteriormente se empaquetó y distribuyó el servicio web utilizando contenedores Docker, se desplegó el servicio web en AWS Elastic Beanstalk y la

interfaz gráfica en AWS S3.

5.5.1. Dominio

Se llevó a cabo la adquisición de un dominio para la aplicación. Este dominio se eligió a partir de la disponibilidad, relevancia al tema del presente trabajo terminal y costo. El dominio elegido fue `skyprice.xyz`.

Para llevar a cabo la compra del dominio, se utilizó el servicio de *Namecheap*, el cual ofrece una interfaz amigable y opciones de configuración avanzadas [112]. La compra del dominio se realizó sin problemas y, en la figura 5.29 se muestra la información del dominio en el panel de control de *Namecheap*.

The screenshot shows the Namecheap domain control panel for the domain `skyprice.xyz`. The left sidebar navigation includes: Dashboard, Expiring / Expired, Domain List (selected), Hosting List, Private Email, SSL Certificates, Apps, and Profile. The main content area is titled "Domains --> Details" for `skyprice.xyz`. It displays several tabs: Domain (selected), Products, Sharing & Transfer, and Advanced DNS. Under the Domain tab, sections include: STATUS & VALIDITY (Active, valid until Apr 8, 2025), WHOIS & PRIVACY (Protection enabled, valid until Apr 8, 2025), PremiumDNS (Enable PremiumDNS protection), and NAME SERVERS (Namecheap BasicDNS selected). Below these are sections for REDIRECT DOMAIN (no redirects defined), REDIRECT EMAIL (Alias: Catch-All forwarded to humbertowoody@gmail.com), PRIVATE EMAIL (described as a secure solution for email needs), and DOMAIN CONTACTS (listing Registrant, Administrator, Technical, and Billing contacts, all listed as Humberto Alcocer). A top navigation bar includes: CONTACT US, user profile, a shopping cart icon (0 items), and links for Domains, Hosting, WordPress, Email, Marketing Tools, Security, Transfer to Us, Help Center, and Account.

Figura 5.29: Información del dominio `skyprice.xyz` en Namecheap.

AWS Route 53

Inicialmente se contemplaba realizar la adquisición del dominio directamente en AWS, sin embargo, debido a que Route53 no ofrece dominios con exten-

sión .xyz, se optó por adquirir el dominio en Namecheap y posteriormente configurar los registros DNS en Route53.

En el cuadro 5.2 se muestra un resumen de los registros DNS configurados en AWS Route 53 para el dominio `skyprice.xyz`.

Nombre	Tipo	Valor
<code>api.skyprice.xyz</code>	CNAME	DNS de Elastic Beanstalk
<code>www.skyprice.xyz</code>	CNAME	DNS de CloudFront
<code>skyprice.xyz</code>	CNAME	DNS de CloudFront

Cuadro 5.2: Registros DNS configurados en AWS Route 53 para el dominio `skyprice.xyz`.

5.5.2. Empaque y distribución del servicio web

Uno de los aspectos más importantes de la implementación de cualquier sistema de software es la distribución del mismo. En este caso, se optó por utilizar contenedores Docker para empaquetar y distribuir el servicio web.

En este sentido, se creó un *Dockerfile* que puede observarse en el listado 5.18, el cual define la configuración y las dependencias necesarias (a nivel de sistema operativo y de software) para ejecutar el servicio. Posteriormente, se construyó una imagen Docker a partir de este archivo, la cual se publicó en un repositorio de imágenes, en este caso, Docker Hub.

De esta manera, se asegura que la misma configuración se utilice en cualquier entorno (local o en la nube), lo que facilita el desarrollo, las pruebas y el despliegue del servicio web.

En particular, Amazon Web Services (AWS) Elastic Beanstalk fue el servicio utilizado para desplegar el servicio web en la nube. Elastic Beanstalk permite utilizar imágenes Docker alojadas en Docker Hub, lo que facilita llevar cambios a producción de manera rápida y segura.

Listing 5.18: Dockerfile para el servicio web

```
1 # Usar una imagen oficial de Python como imagen base
2 FROM python:3.12-slim as builder
3
4 # Actualizar el sistema e instalar dependencias necesarias para SciPy,
5 # Scikit-learn, TensorFlow, Keras, etc.
6 RUN apt-get update && apt-get install -y --no-install-recommends \
7     build-essential \
8     libblas-dev \
9     liblapack-dev \
    libatlas-base-dev \
```

```

10      gfortran \
11      libhdf5-dev \
12      libfreetype6-dev \
13      libpng-dev \
14      libzmq3-dev \
15      pkg-config \
16      software-properties-common \
17      swig \
18      curl \
19      git \
20      g++ \
21      gcc \
22      && apt-get clean \
23      && rm -rf /var/lib/apt/lists/*
24
25 # Establecer el directorio de trabajo en el contenedor
26 WORKDIR /app
27
28 # Copiar Pipfile y Pipfile.lock al contenedor
29 COPY Pipfile Pipfile.lock /app/
30
31 # Instalar pipenv y las dependencias del proyecto
32 RUN pip install --upgrade pip pipenv && \
33     pipenv install --system --deploy
34
35 # Copiar el resto del código de la aplicación al contenedor
36 COPY . /app
37
38 # Etapa final
39 FROM python:3.12-slim as run
40
41 # Copiar los archivos de la etapa anterior
42 COPY --from=builder /usr/local/lib/python3.12/site-packages /usr/local/lib/
43     python3.12/site-packages
44 COPY --from=builder /usr/local/bin /usr/local/bin
45 COPY --from=builder /app /app
46
47 # Establecer el directorio de trabajo en el contenedor
48 WORKDIR /app
49
50 # Instalar dependencias adicionales
51 RUN apt-get update && apt-get install -y --no-install-recommends \
52     libhdf5-dev \
53     libfreetype6-dev \
54     libpng-dev \
55     && apt-get clean \
56     && rm -rf /var/lib/apt/lists/*
57
58 # Exponer el puerto 5000
59 EXPOSE 5000
60
61 # Comando para ejecutar la aplicación usando Uvicorn
62 CMD ["uvicorn", "main:app", "--host", "0.0.0.0", "--port", "5000"]

```

Construcción de la imagen

El Dockerfile que se muestra en el listado 5.18 define dos etapas: una para construir las dependencias necesarias para la aplicación y otra para ejecutar la

aplicación. En la primera etapa, se instalan las dependencias necesarias para SciPy, Scikit-learn, TensorFlow, Keras, etc. Posteriormente, se instala pipenv y las dependencias del proyecto. En la segunda etapa, se copian los archivos de la etapa anterior y se instalan las dependencias adicionales necesarias para la aplicación. Finalmente, se expone el puerto 5000 y se ejecuta la aplicación utilizando Uvicorn.

Para construir la imagen Docker, se empleó el script que se muestra en el listado 5.19. Este script construye la imagen base y la imagen de ejecución, y posteriormente sube la imagen a Docker Hub.

Listing 5.19: Script para construir y subir imágenes de Docker

```

1 #!/bin/bash
2 # Script para construir las imágenes de Docker y subirlas a Docker Hub
3
4 # Definir la URL de la imagen en Docker Hub
5 DOCKER_URL="humbertowoodly/skyprice-api"
6 printf "Docker URL: $DOCKER_URL\n"
7
8 # Construir la imagen base
9 docker buildx build --platform linux/arm64,linux/amd64 --cache-from
  $DOCKER_URL-builder:latest --cache-to=type:inline --progress plain -t
  $DOCKER_URL-builder:latest --target builder .
10
11 # Construir la imagen de ejecución
12 docker buildx build --platform linux/arm64,linux/amd64 --cache-from
  $DOCKER_URL-builder:latest --cache-from $DOCKER_URL:latest --cache-to=
    type:inline --progress plain -t $DOCKER_URL:latest --target run .
13
14 # Subir la imagen a Docker Hub
15 docker push $DOCKER_URL:latest

```

Otro aspecto importante a destacar es el uso de la herramienta BuildKit de Docker, la cual permite construir imágenes de manera más eficiente y rápida [113]. En este caso, se utilizó la opción `-platform` para construir imágenes multiplataforma que pueden ejecutarse en arquitecturas ARM y x86 [114]. Esto es útil para garantizar la portabilidad de la aplicación y su ejecución en diferentes entornos, tanto en dispositivos locales como en la nube.

Distribución de la imagen

La imagen Docker se publicó en un repositorio de imágenes, como Docker Hub, permitiendo su distribución y despliegue en diferentes entornos. Esto asegura que la misma configuración se utilice en desarrollo, prueba y producción.

En la figura 5.30 se muestra la imagen publicada en Docker Hub, la cual está disponible para su descarga y despliegue en cualquier entorno que soporte contenedores Docker.

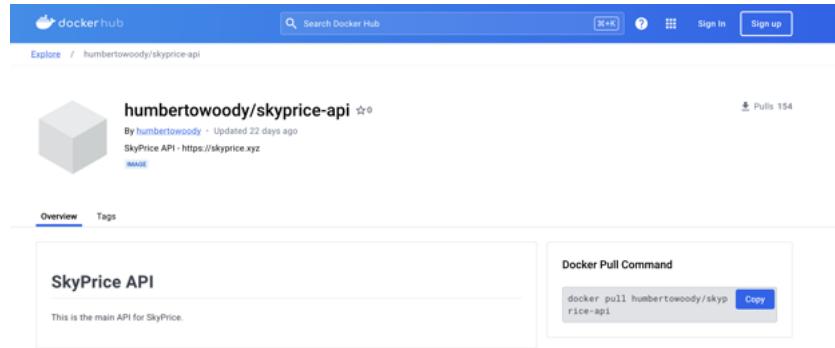


Figura 5.30: Imagen de Docker publicada en Docker Hub [1].

5.5.3. Despliegue del servicio web

Una vez que el servicio web se encuentra en Docker Hub, se inicia su despliegue en AWS utilizando Elastic Beanstalk. Para ello, primeramente se crea un nuevo certificado SSL en AWS Certificate Manager, el cual se asocia al dominio de la aplicación. Posteriormente, se crea un entorno de Elastic Beanstalk para la aplicación, se configuran las variables de entorno y se despliega la imagen de Docker en el entorno.

AWS Certificate Manager

Utilizar un certificado SSL es importante para garantizar la seguridad y la confidencialidad de los datos que se transmiten entre el cliente y el servidor. Más aún, en el caso de aplicaciones web, es fundamental para proteger la información sensible de los usuarios, como contraseñas, datos personales y financieros. Por ello, se creó un certificado SSL en AWS Certificate Manager para el dominio de la aplicación, el cual se muestra en la figura 5.31. Este certificado SSL responde a dos subdominios: `api.skyprice.xyz` y `skyprice.xyz`. El primero se utiliza para el servicio web y el segundo para la interfaz gráfica, lo que permite garantizar la integridad de ambas partes.

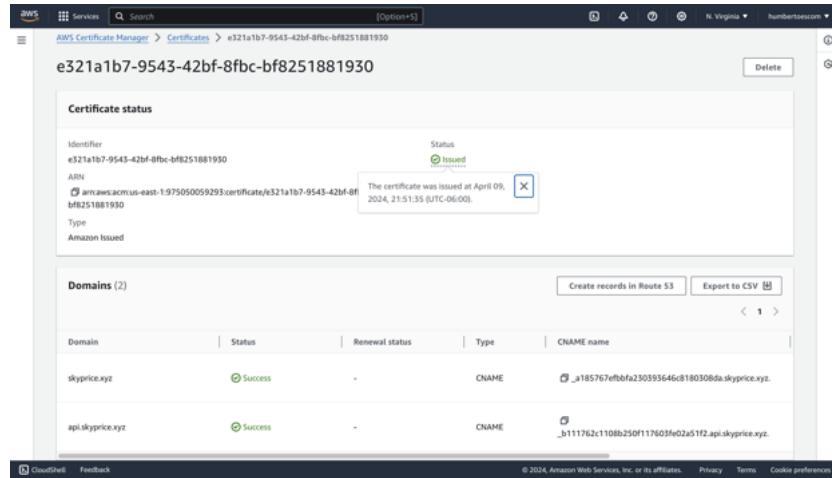


Figura 5.31: Certificado SSL en AWS Certificate Manager.

Aunado a esto, fue necesario configurar el dominio para que integrara los registros requeridos para la validación del certificado SSL. En la figura 5.32 se muestra la configuración de los registros DNS para el dominio de la aplicación.

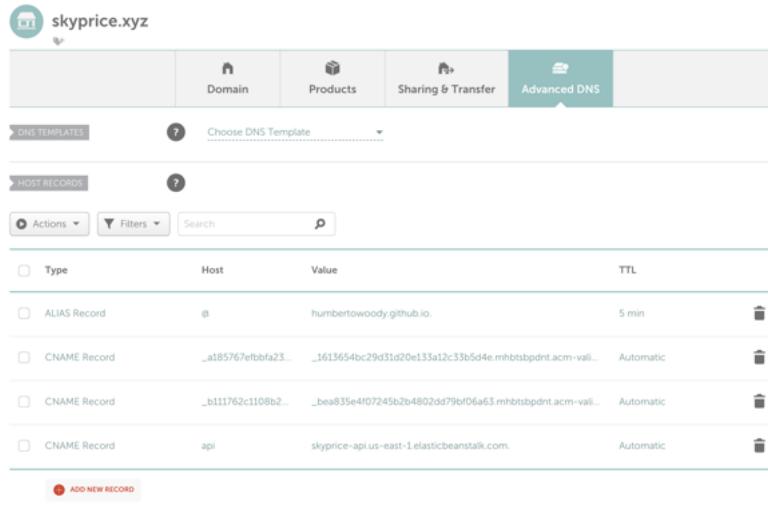


Figura 5.32: Registros DNS para el dominio de la aplicación.

Una vez que AWS Certificate Manager validó el certificado SSL, ahora se puede crear el entorno de Elastic Beanstalk para la aplicación y asociar el certificado SSL al dominio de la aplicación, lo que garantiza una conexión segura y cifrada.

AWS Elastic Beanstalk

Elastic Beanstalk es un servicio de AWS que facilita el despliegue y la administración de aplicaciones web y servicios en la nube. Usarlo requiere crear un ambiente, el cual se llamará `skyprice-api` en este caso, y configurar las variables de entorno necesarias para la aplicación. Cuando se crea un ambiente, debe seleccionarse la plataforma de Docker, ya que la aplicación se ejecuta en un contenedor Docker, y se debe especificar la imagen de Docker que se utilizará para el despliegue, en nuestro caso, la imagen de Docker publicada en Docker Hub.

En el listado 5.20 se muestra el archivo de configuración `Dockerrun.aws.json` que se utiliza para especificar la configuración del contenedor Docker en Elastic Beanstalk. En este archivo se define la imagen de Docker que se utilizará, así como la configuración de red que se aplicará al contenedor. Es importante destacar que se está utilizando la versión 1 de `AWSEBDockerrunVersion` ya que es la versión adecuada para un único contenedor Docker, la versión 2 se utiliza para aplicaciones multicontenedor.

Listing 5.20: Archivo de configuración para Elastic Beanstalk

```
1 {
2     "AWSEBDockerrunVersion": "1",
3     "Image": {
4         "Name": "humbertowoodly/skyprice-api:latest",
5         "Update": "true"
6     },
7     "Ports": [{ "ContainerPort": 5000 }]
8 }
```

Una vez que se ha creado el ambiente y se ha configurado la imagen de Docker, se puede desplegar la aplicación en Elastic Beanstalk. En la figura 5.33 se muestra el panel de control de Elastic Beanstalk con el entorno de la aplicación `skyprice-api` en ejecución.

The screenshot shows the AWS Elastic Beanstalk environment overview for the application 'skyprice-api'. Key details include:

- Health:** Ok
- Environment ID:** e-zii25femm9
- Domain:** skyprice-api.us-east-1.elasticbeanstalk.com
- Application name:** skyprice-api
- Platform:** Docker running on 64bit Amazon Linux 2023/4.3.2
- Running version:** 1.0.0-1
- Platform state:** Supported

Overall health:

Requests / second	2XX responses	3XX responses	4XX responses	5xx responses
0.1	1	-	-	-
P99 latency(ms)	P90 latency(ms)	P75 latency(ms)	P50 latency(ms)	P10 latency(ms)
2	2	2	2	2

Enhanced instance health (1):

Figura 5.33: Entorno de Elastic Beanstalk en ejecución.

Verificación del despliegue

Una vez que la aplicación se ha desplegado en Elastic Beanstalk, es importante verificar que el despliegue se haya realizado correctamente y que la aplicación esté funcionando correctamente. Para ello, se puede acceder a la URL proporcionada por Elastic Beanstalk y verificar que la aplicación responda correctamente. Además, se puede verificar que también se respondan a peticiones al dominio personalizado y que el certificado SSL esté funcionando correctamente. En la figura 5.34 se muestra la respuesta de la aplicación al acceder a la URL <https://api.skyprice.xyz>, así como información del certificado SSL.

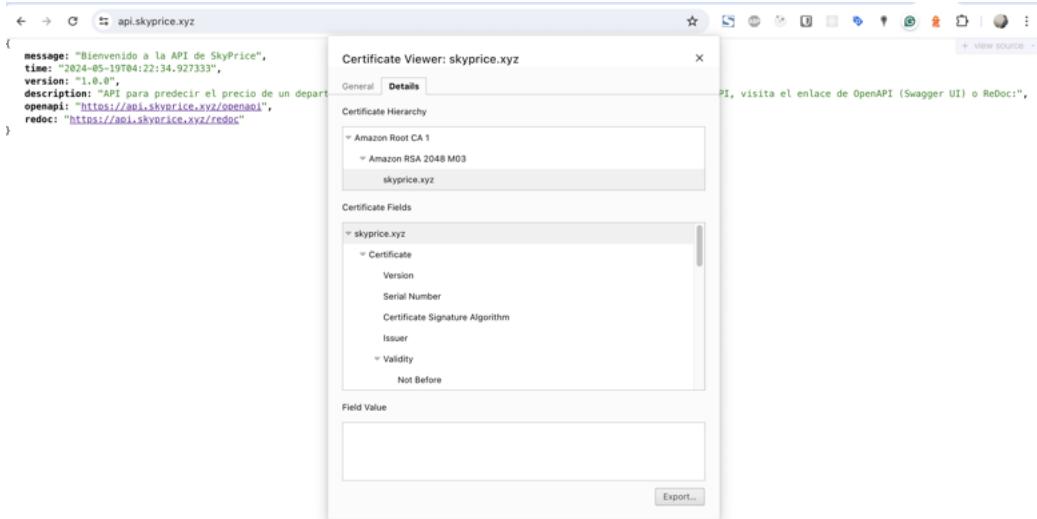


Figura 5.34: Verificación SSL para la URL <https://api.skyprice.xyz>.

5.5.4. Despliegue de la interfaz gráfica

La interfaz gráfica corresponde a una aplicación React utilizando NextJS, la cual se construyó, minificó y empaquetó para su despliegue en un bucket de AWS S3. Posteriormente se configuró un CDN con AWS CloudFront para distribuir el contenido de manera eficiente y segura.

Compilación y empaquetado

Para compilar y empaquetar la interfaz gráfica, se utilizó el comando `npm build` de NextJS, el cual compila la aplicación y la empaqueta en una carpeta `out` que contiene los archivos estáticos y optimizados para producción. Posteriormente, estos archivos serán subidos a un bucket de AWS S3 para su distribución.

En la figura 5.35 se muestra la salida del comando `npm build` en la terminal, donde se puede observar que la compilación se realizó correctamente y que se generaron los archivos estáticos en la carpeta `out`.

```

> npm run build

> skyprice-front@1.0.0 build
> next build

  ▲ Next.js 14.2.3
    - Environments: .env.local

      Creating an optimized production build ...
      ✓ Compiled successfully
      ✓ Linting and checking validity of types
      ✓ Collecting page data
      ✓ Generating static pages (13/13)
      ✓ Collecting build traces
      ✓ Finalizing page optimization

      Route (app)                                Size     First Load JS
      └─ o /                                         66.9 kB      234 kB
          └─ o /_not-found                           871 B       88 kB
          └─ o /acerca-de                            63.1 kB      230 kB
          └─ o /apple-icon.png                         0 B        0 kB
          └─ o /icon1.png                            0 B        0 kB
          └─ o /icon2.png                            0 B        0 kB
          └─ o /manifest.json                          0 B        0 kB
          └─ o /mapa                                 3.53 kB      163 kB
          └─ o /robots.txt                            0 B        0 kB
          └─ o /sitemap.xml                           0 B        0 kB
      + First Load JS shared by all              87.2 kB
          └─ chunks/23-133a31be6d5238d1.js           31.6 kB
          └─ chunks/fd9d1056-4c0823410a8df68a.js      53.6 kB
          └─ other shared chunks (total)                1.98 kB

      o (Static) prerendered as static content

```

Figura 5.35: Compilación de la interfaz gráfica con NextJS.

AWS S3

La interfaz gráfica se desplegó en un bucket de AWS S3, lo que permite servir los archivos estáticos de manera eficiente y segura. Se configuró el bucket para ser accesible públicamente y se implementaron políticas de seguridad adecuadas. En la figura 5.36 se muestra el panel de control de AWS S3 con el bucket `skyprice-front` en el que se alojó la interfaz gráfica.

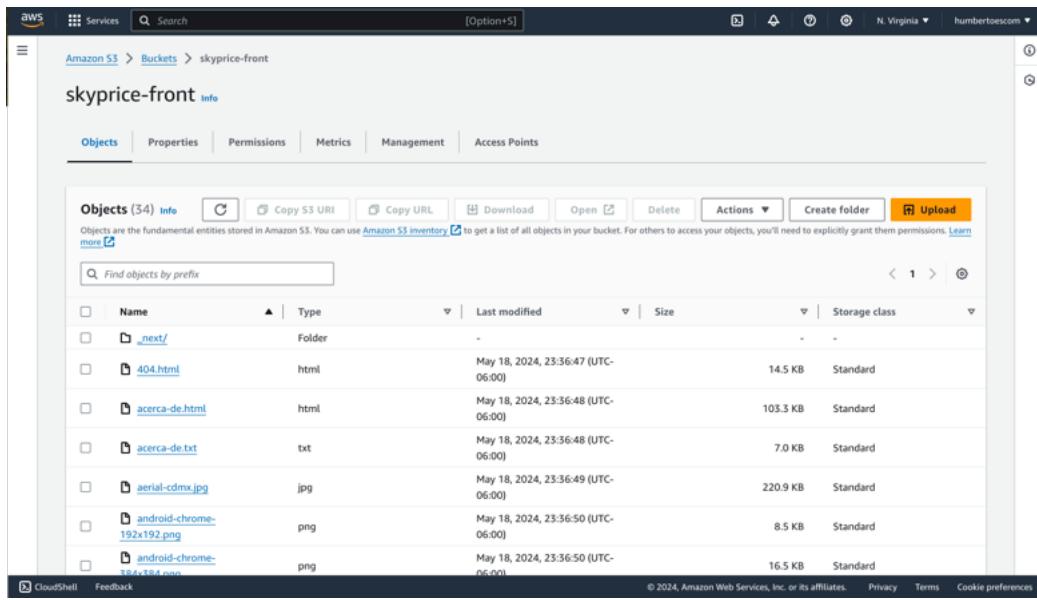


Figura 5.36: Bucket de AWS S3 con la interfaz gráfica.

AWS CloudFront

Se utilizó AWS CloudFront para distribuir el contenido de la interfaz gráfica a través de una red de distribución de contenido (CDN). Esto mejora el rendimiento y reduce la latencia para los usuarios de diferentes regiones geográficas. Además, CloudFront proporciona una capa adicional de seguridad y protección contra ataques Distributed Denial of Service (DDoS) [115].

En la figura 5.37 se muestra el panel de control de AWS CloudFront con la distribución de contenido creada para la interfaz gráfica. Se configuró CloudFront para que se conecte al bucket de S3 y distribuya el contenido de manera eficiente y segura.

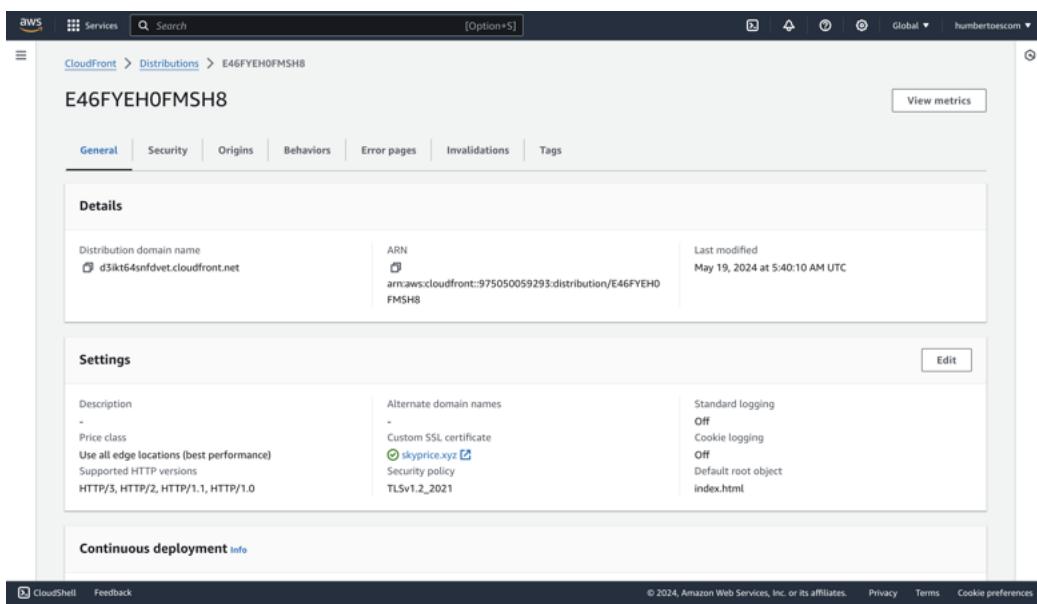


Figura 5.37: Distribución de contenido de AWS CloudFront.

Verificación del despliegue

Una vez que la interfaz gráfica se ha desplegado en AWS S3 y se ha configurado CloudFront para distribuir el contenido, es importante verificar que el despliegue se haya realizado correctamente y que la aplicación esté funcionando correctamente. Para ello, se puede acceder a la URL proporcionada por CloudFront y verificar que la aplicación responda correctamente. En la figura 5.38 se muestra la respuesta de la aplicación al acceder a la URL <https://skyprice.xyz>.

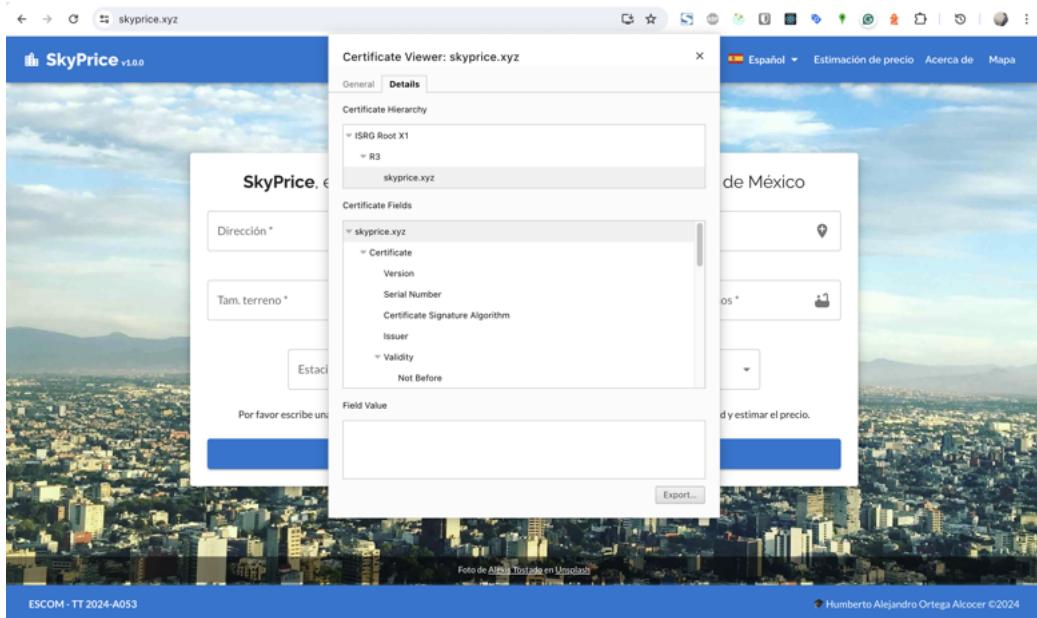


Figura 5.38: Verificación SSL para la URL <https://skyprice.xyz>.

5.5.5. Servicios externos

Para el correcto funcionamiento de la aplicación, se utilizaron servicios externos como Google Maps y ExchangeRate-API. Estos servicios permiten realizar la geocodificación de direcciones y la conversión de monedas, respectivamente, lo que enriquece la experiencia del usuario y agrega valor a la aplicación. En esta sección se describen los servicios externos utilizados y se muestran ejemplos de su integración en la aplicación.

Google Maps

Para llevar a cabo la geocodificación de la dirección de la propiedad a estimar su precio, se utilizó la API de *Places* de Google Maps. Esta API permite realizar búsquedas de direcciones y obtener información detallada sobre las mismas, como coordenadas geográficas, nombres de lugares, tipos de establecimientos, entre otros. En la figura 5.39 se muestra la configuración de la API de Google Maps en la consola de Google Cloud.

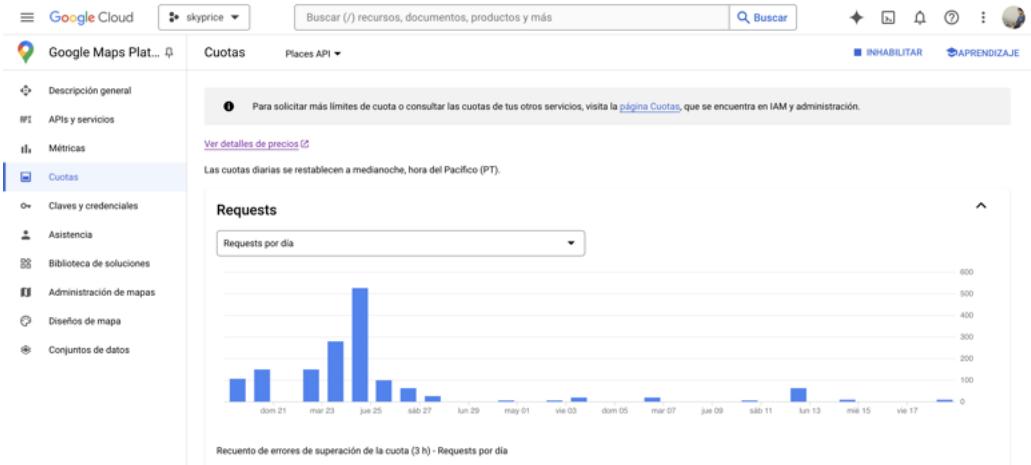


Figura 5.39: Configuración de la API de Google Maps en Google Cloud.

Así, se puede observar que la API se ha configurado exitosamente ya que se muestra el estado *Habilitado* en la consola de Google Cloud. Además, se puede observar el consumo de la API en términos de solicitudes.

ExchangeRate-API

Se implementó un servicio de conversión de monedas utilizando una API externa, permitiendo a los usuarios ver los precios en diferentes monedas. Esto es especialmente útil para usuarios internacionales que deseen ver los precios en su moneda local. Para ello, se utilizó la API de *ExchangeRate-API*, que proporciona tasas de cambio en tiempo real y es fácil de integrar en aplicaciones web y móviles [116].

En la figura 5.40 se muestra la página principal del servicio, dónde se observan las estadísticas de uso de la API, así como la información de la cuenta y las opciones de configuración.

The screenshot shows the ExchangeRate-API dashboard. On the left, there's a sidebar with links like Dashboard, API Keys, Request Usage, Account Details, Manage Plan, Billing, Invoices, Documentation, Docs Overview, Supported Currencies, Standard API Requests, Product Homepage, Email Support, and a support email address. The main content area has three sections: 'API Access' with fields for 'Your API Key' and an example request URL; 'API Request Quota Usage' showing a progress bar for available requests (1333) and stats for today (0), this month (167), last month (0), and monthly quota reset (23rd); and 'Account & Plan Summary' showing a free plan with no payment required since April 23, 2024.

Figura 5.40: Página principal de la API de ExchangeRate-API.

En el listado 5.21 se muestra el código requerido para realizar la conversión de moneda utilizando la API de *ExchangeRate-API*. En este caso, se utiliza `fetch` para realizar la petición HTTP a la API y obtener la tasa de cambio entre la moneda base y la moneda de interés.

Listing 5.21: Conversión de moneda utilizando la API de ExchangeRate-API

```

1  useEffect(() => {
2    if (currency === 'MXN') {
3      setConvertedPrice(price);
4      return;
5    }
6    const fetchData = async () => {
7      try {
8        const response = await fetch(
9          `https://v6.exchangerate-api.com/v6/${
10            process.env.NEXT_PUBLIC_EXCHANGERATE_API_KEY || ''
11          }/pair/MXN/${currency}/${price}`,
12        {
13          method: 'GET',
14          headers: {
15            'Content-Type': 'application/json',
16          },
17          mode: 'cors',
18        },
19      );
20      const jsonData = await response.json();
21      setConvertedPrice(jsonData.conversion_result);
22    } catch (error) {
23      console.error('Error fetching currency data:', error);
24      alert(t('predictionForm.currencyConverter.requestError'));
25    }
  
```

```
26     };
27
28     fetchData();
29 }, [currency, price]);
```

5.6. Pruebas del sistema

Una vez que la aplicación se ha desplegado en la nube, es importante realizar pruebas exhaustivas para garantizar que la aplicación funcione correctamente y cumpla con los requisitos y especificaciones establecidos. En este sentido, se realizaron pruebas de funcionalidad, rendimiento y seguridad para verificar el correcto funcionamiento de la aplicación. Además, se utilizaron herramientas como Google PageSpeed Insights y Google Analytics para evaluar el rendimiento y la experiencia del usuario.

5.6.1. Google Page Speed Insights

Google PageSpeed Insights es una herramienta que analiza el rendimiento de una página web tanto en dispositivos móviles como en escritorio. Utiliza Lighthouse para generar informes detallados y ofrece sugerencias para mejorar la velocidad y la experiencia del usuario. Lighthouse es una herramienta automatizada de código abierto que realiza auditorías en las páginas web, evaluando aspectos como la accesibilidad, el rendimiento, las mejores prácticas y la SEO [117]. A continuación se presentan los resultados de las pruebas de Google PageSpeed Insights en la interfaz gráfica de la aplicación tanto en un entorno de escritorio como en un entorno móvil.

Escritorio

En la figura 5.41 se muestra el resultado de las pruebas de Lighthouse en la interfaz gráfica en un entorno de escritorio. Las puntuaciones obtenidas muestran un rendimiento, accesibilidad y buenas prácticas sobresalientes. El SEO obtuvo una puntuación menor debido a la especificación de metadatos que previenen la indexación de la página.

Informe desde el 18 may 2024, 10:27:23 p.m.

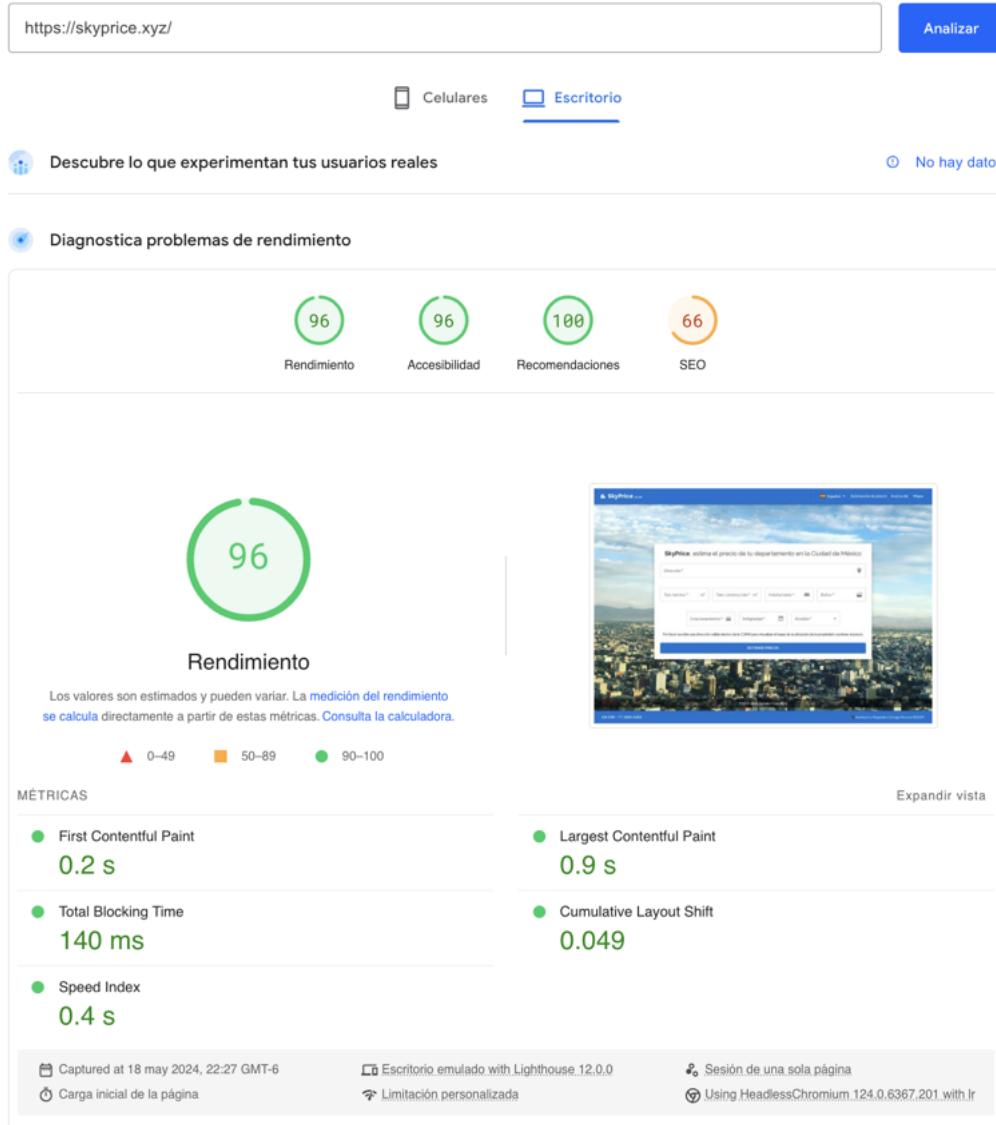


Figura 5.41: Resultados de las pruebas de Lighthouse en un entorno de escritorio.

Móvil

En la figura 5.42 se muestra el resultado de las pruebas de Lighthouse en la interfaz gráfica en un entorno móvil. Las puntuaciones obtenidas muestran un rendimiento malo debido a la falta de compresión de imágenes, la accesibilidad y las buenas prácticas son sobresalientes, y el SEO obtuvo una

puntuación menor debido a la especificación de metadatos que previenen la indexación de la página.

Informe desde el 18 may 2024, 10:35:56 p.m.

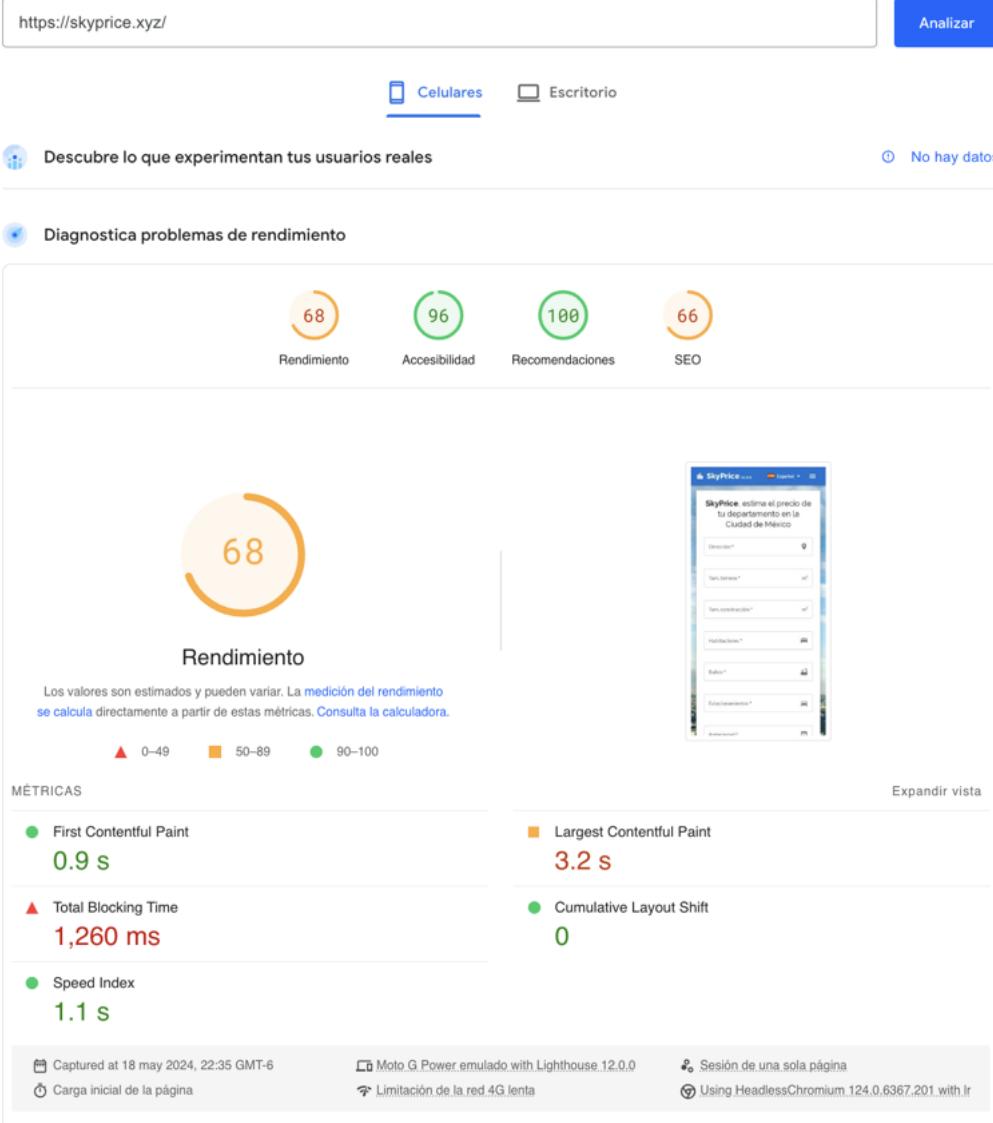


Figura 5.42: Resultados de las pruebas de Lighthouse en un entorno móvil.

Es importante destacar que las puntuaciones obtenidas en las pruebas anteriores no son definitivas y pueden variar dependiendo de factores ajenos al desarrollo de la aplicación, como la velocidad de la conexión a Internet, la

ubicación geográfica del usuario, el dispositivo utilizado, entre otros.

5.6.2. Google Analytics

Google Analytics es una plataforma de análisis web que ofrece herramientas para medir el tráfico y el rendimiento del sitio web. Proporciona informes detallados sobre cómo los usuarios interactúan con el sitio, permitiendo a los administradores web y a los desarrolladores comprender mejor el comportamiento de los usuarios. Con Google Analytics, es posible rastrear diversas métricas, como la cantidad de visitantes, las páginas más vistas, las tasas de conversión y la efectividad de campañas publicitarias. Esta información es crucial para optimizar la experiencia del usuario y tomar decisiones informadas basadas en datos [118].

En la figura 5.43 se muestra la página principal de Google Analytics con información sobre el tráfico del sitio web. En este caso, se puede observar el número de usuarios activos, las páginas más visitadas, la tasa de rebote y la duración promedio de la sesión. Esta información es valiosa para evaluar el rendimiento del sitio web y realizar ajustes en función de los datos obtenidos.

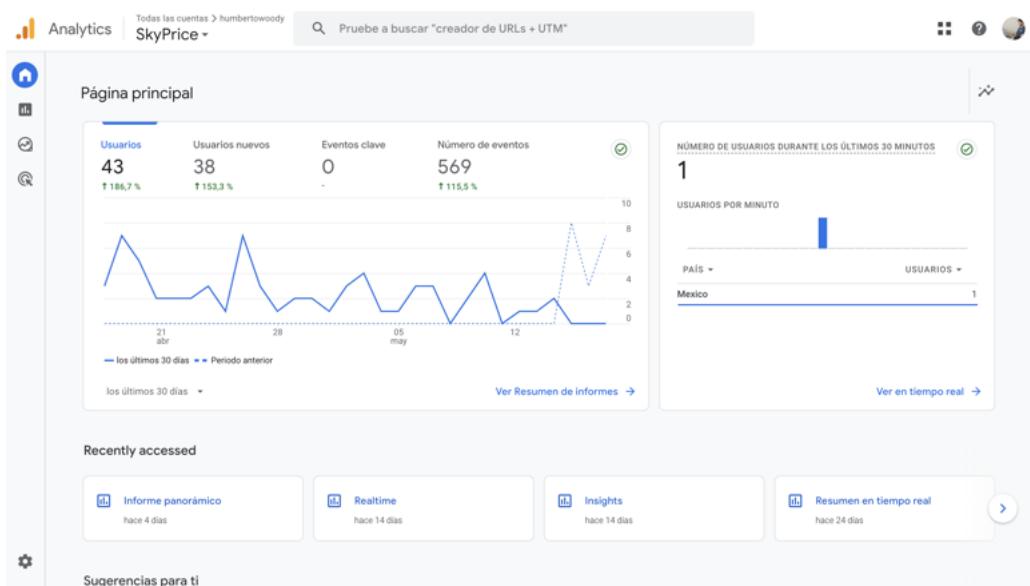


Figura 5.43: Página principal de Google Analytics con información sobre el tráfico del sitio web.

Para integrar Google Analytics en la interfaz gráfica de la aplicación, se utilizó Google Tag Manager, una herramienta que permite gestionar y desplegar

etiquetas de seguimiento en un sitio web de manera sencilla y eficiente. En el listado 5.22 se muestra el fragmento de código que se incluyó en la interfaz gráfica para integrar Google Tag Manager. Este código corresponde al componente principal de la aplicación, y garantiza que Google Analytics se cargue en todas las páginas del sitio web.

Listing 5.22: Fragmento de código para Google Tag Manager

```
1 export default function RootLayout(props: { children: React.ReactNode }) {
2   const urlgtm: string = 'https://www.googletagmanager.com/gtag/js?id=${process.env.NEXT_PUBLIC_GOOGLE_ANALYTICS}';
3   return (
4     <html lang="en">
5       <Script src={urlgtm} />
6       <Script>
7         {
8           window.dataLayer = window.dataLayer || [];
9           function gtag(){dataLayer.push(arguments);}
10          gtag('js', new Date());
11          gtag('config', '${process.env.NEXT_PUBLIC_GOOGLE_ANALYTICS}');
12        }
13       </Script>
14     <body>
15       <AppRouterCacheProvider options={{ enableCssLayer: true }}>
16         <ThemeProvider theme={{theme}}>
17           {/* CssBaseline kickstart an elegant, consistent, and simple baseline to build upon. */}
18           <CssBaseline />
19           <I18nProvider>{props.children}</I18nProvider>
20           </ThemeProvider>
21           </AppRouterCacheProvider>
22         </body>
23       </html>
24     );
25 }
```

Información sobre los usuarios

Los resultados de Google Analytics proporcionan información detallada sobre los usuarios que visitan el sitio web. En la figura 5.44 se muestra un resumen de la información sobre los usuarios, incluyendo la cantidad de usuarios activos, las sesiones, la duración promedio de la sesión y la tasa de rebote.

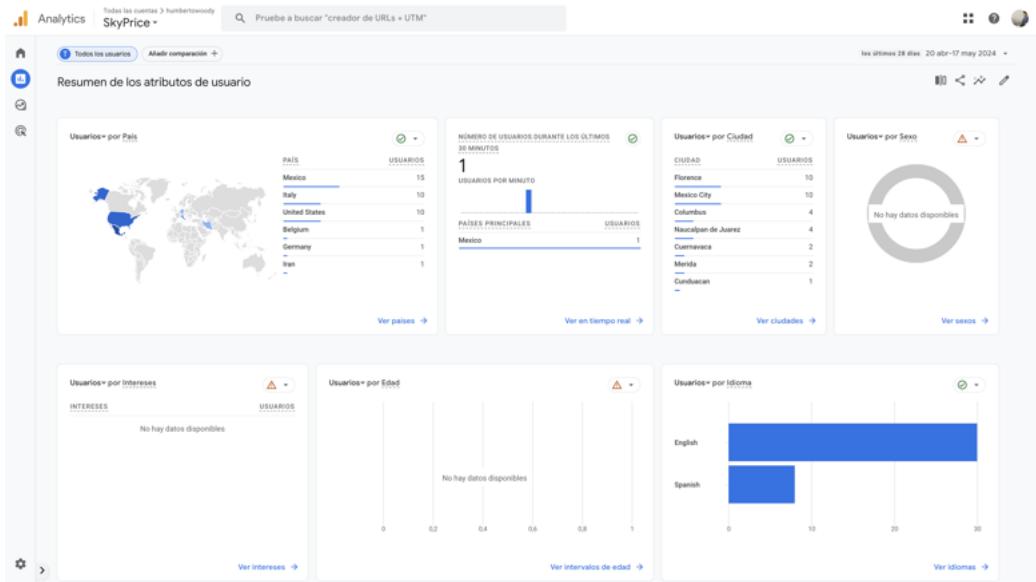


Figura 5.44: Información sobre los usuarios que visitan el sitio web.

Información sobre la tecnología

Google Analytics también proporciona información sobre la tecnología utilizada por los usuarios para acceder al sitio web. En la figura 5.45 se muestra un resumen de la información sobre la tecnología, incluyendo el tipo de dispositivo, el sistema operativo y el navegador utilizado por los usuarios.

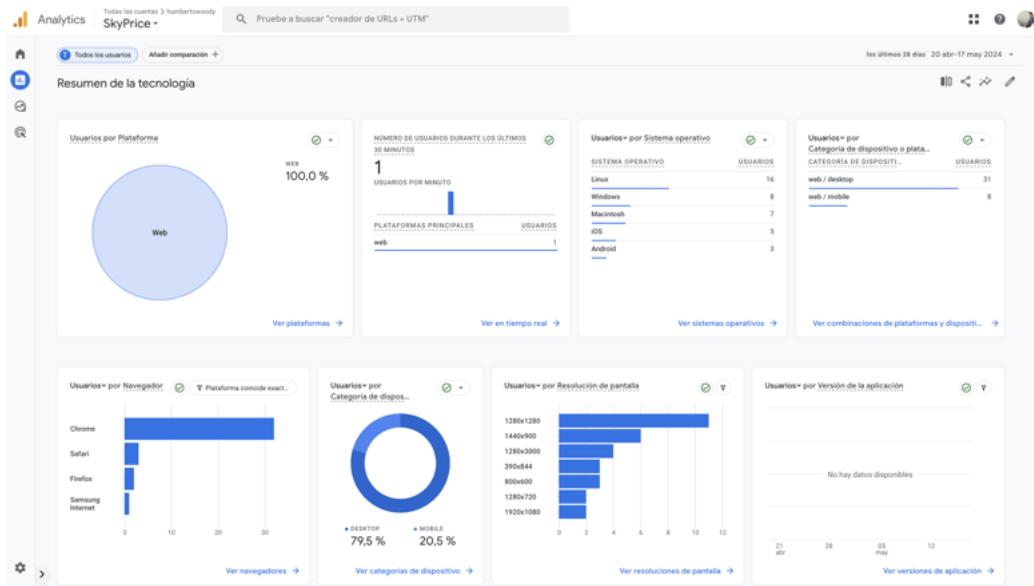


Figura 5.45: Información sobre la tecnología utilizada por los usuarios para acceder al sitio web.

5.7. Costos

La operación de SkyPrice implica costos asociados al despliegue de la aplicación y los servicios externos involucrados. En esta sección se presentan los costos asociados al despliegue y ejecución de la aplicación en el entorno de AWS, así como de servicios externos como Google Maps y ExchangeRate-API.

5.7.1. AWS

El despliegue de la aplicación en la nube se realizó el día 8 de Abril del 2024, con esto se cuenta con un mes de uso de los servicios de AWS lo cual nos permite observar costes finales de operación. En la figura 5.46 se muestra el costo diario de la aplicación, incluyendo servicios como Elastic Beanstalk, S3, CloudFront, Certificate Manager, entre otros.

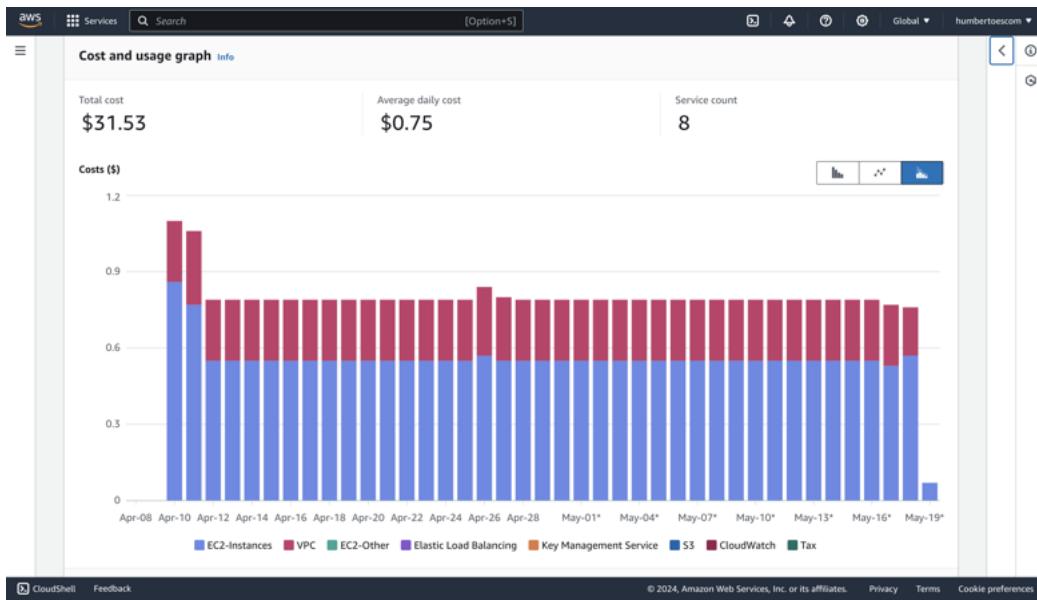


Figura 5.46: Costos diarios de la aplicación en la nube.

Puede destacarse que los dos servicios que más consumen recursos son Elastic Beanstalk, con EC2 que es el servicio de cómputo, con un costo de \$0.50 USD diarios y el tráfico de VPC que es el tráfico de red, con un costo de \$0.24 USD diarios, estimando un costo total de \$0.75 USD diarios (el centavo extra se compone de cargos para el resto de servicios implicados). Esto nos da un costo mensual de \$15.77 USD (\$265.00 MXN), lo cual es un costo accesible para el mantenimiento de la aplicación y queda muy por debajo de las estimaciones realizadas.

5.7.2. Google Maps

El uso de la API de Google Maps para la geocodificación de direcciones implica un coste fijo por petición de dirección geocodificada de \$0.005 USD. Al igual que AWS, la cuenta de Google Cloud ofrece un crédito inicial de \$300 USD para nuevos usuarios, lo cual permite realizar un número considerable de peticiones sin coste adicional. En la figura 5.47 se muestra el costo registrado desde el día 8 de Abril del 2024 que es el día en que se realizó el despliegue de la aplicación.

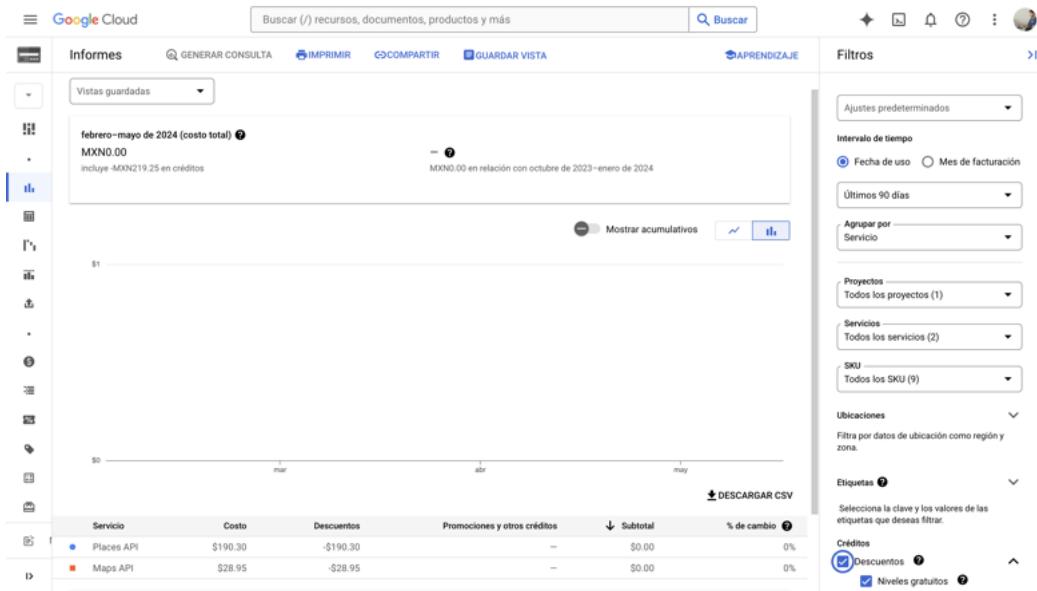


Figura 5.47: Costos de la API de Google Maps para la geocodificación de direcciones.

Se puede observar que el coste total de la API de Google Maps es de \$0.00 USD, lo cual se debe al crédito inicial de \$300 USD que se ha utilizado para cubrir los costes de las peticiones de geocodificación de direcciones.

5.7.3. ExchangeRate-API

ExchangeRate-API utiliza un modelo de precios basado en paquetes, para nuestro proyecto se empleó el paquete gratuito que ofrece 1500 peticiones mensuales sin coste adicional. En la figura 5.48 se muestra el coste registrado desde el día 23 de Abril del 2024 que es el día en que se realizó el despliegue de la aplicación con la integración de la API de ExchangeRate-API.

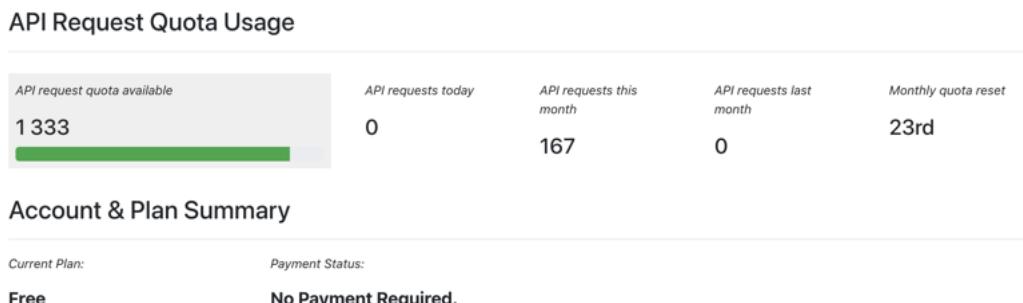


Figura 5.48: Costos de la API de ExchangeRate-API para la conversión de monedas.

Se puede observar que se disponen de 1333 peticiones mensuales restantes al momento de la captura, lo cual indica que se ha utilizado un total de 167 peticiones de conversión, lo cual nos deja con un umbral aceptable sin coste adicional.

Capítulo 6

Conclusiones

El desarrollo de una aplicación web para la predicción de precios de apartamentos, como SkyPrice, ha demostrado ser una herramienta valiosa y eficiente para los usuarios interesados en el mercado inmobiliario. A través de este proyecto, se ha logrado integrar diferentes tecnologías y metodologías avanzadas, destacando el uso de algoritmos de inteligencia artificial y aprendizaje automático para proporcionar estimaciones precisas y confiables.

La plataforma SkyPrice fue diseñada con un enfoque en la accesibilidad y la usabilidad, permitiendo a los usuarios acceder a las predicciones desde cualquier dispositivo con conexión a internet y un navegador web. Además, se implementó un diseño intuitivo y moderno, facilitando la navegación y la interacción del usuario con la aplicación. Las instrucciones claras en cada paso del proceso garantizan una experiencia amigable y satisfactoria.

El modelo de predicción utilizado en SkyPrice se entrenó con un conjunto de datos representativo del mercado inmobiliario, lo que permitió obtener una precisión significativa en las estimaciones de precios. Los resultados del modelo mostraron una alta correlación entre los precios predichos y los precios reales de los apartamentos, validando la efectividad del enfoque adoptado.

Además, se analizaron los datos de interacción de los usuarios dentro de la plataforma, proporcionando valiosa información sobre las secciones más relevantes y el impacto de la aplicación en línea. Estos datos no solo ayudaron a mejorar la experiencia del usuario, sino que también proporcionaron insights sobre las tendencias y preferencias del mercado inmobiliario.

En conclusión, SkyPrice no solo facilita la toma de decisiones informadas para compradores y vendedores de bienes raíces, sino que también demuestra

el potencial de las tecnologías emergentes para transformar y optimizar procesos en diversas industrias. El éxito de este proyecto subraya la importancia de seguir explorando y aplicando innovaciones tecnológicas para resolver problemas reales y mejorar la calidad de vida de las personas.

Capítulo 7

Trabajo a futuro

Después del lanzamiento inicial de SkyPrice, se han identificado varias áreas de mejora y expansión que pueden ser abordadas en trabajos futuros. Las siguientes propuestas buscan optimizar las funcionalidades actuales y explorar nuevas capacidades para aumentar el valor y la precisión de la plataforma.

1. Mejora del Modelo de Predicción

Actualmente, el modelo de predicción de precios se basa en un conjunto de datos específico. Como trabajo a futuro, se propone:

- **Ampliación del Conjunto de Datos:** Integrar datos adicionales de otras fuentes confiables para mejorar la precisión del modelo. Esto incluye información de mercados inmobiliarios de diferentes ciudades y regiones.
- **Optimización de Algoritmos:** Implementar y comparar diferentes algoritmos de aprendizaje automático (e.g., Random Forest, Gradient Boosting) para determinar cuál proporciona las mejores predicciones.
- **Entrenamiento Continuo:** Desarrollar un sistema de actualización continua del modelo para incorporar nuevos datos y tendencias del mercado en tiempo real.

2. Integración de Factores Ambientales y Sociales

Para ofrecer un análisis más completo del valor de los apartamentos, se propone:

- **Calidad del Aire y Ruido:** Integrar datos sobre la calidad del aire y niveles de ruido en las predicciones de precios, utilizando sensores y fuentes de datos públicas.
- **Accesibilidad y Servicios:** Añadir información sobre la accesibilidad a servicios esenciales (e.g., transporte público, hospitalares, escuelas) y su impacto en los precios inmobiliarios.
- **Seguridad y Criminalidad:** Incorporar estadísticas de seguridad y criminalidad para proporcionar un panorama más claro sobre la seguridad del área.

3. Personalización y Experiencia del Usuario

Para mejorar la experiencia del usuario y proporcionar predicciones más personalizadas:

- **Perfil del Usuario:** Permitir a los usuarios crear perfiles donde puedan especificar sus preferencias y criterios de búsqueda, para recibir recomendaciones más personalizadas.
- **Interfaz Interactiva:** Desarrollar una interfaz más interactiva y visual que permita a los usuarios explorar datos y predicciones de manera intuitiva. Incluir visualizaciones avanzadas y mapas interactivos.
- **Notificaciones y Alertas:** Implementar un sistema de notificaciones que informe a los usuarios sobre cambios significativos en el mercado inmobiliario que puedan afectar sus decisiones.

4. Expansión de la Cobertura Geográfica

Para aumentar el alcance y utilidad de SkyPrice:

- **Nuevas Ciudades y Regiones:** Extender la cobertura de la plataforma a nuevas ciudades y regiones, adaptando el modelo a las características específicas de cada área.

- **Internacionalización:** Evaluar la posibilidad de expandir SkyPrice a nivel internacional, comenzando con estudios de viabilidad y recopilación de datos relevantes en otros países.

5. Análisis Predictivo y Tendencias del Mercado

Para proporcionar insights más profundos y valiosos:

- **Análisis de Tendencias:** Implementar herramientas de análisis que permitan identificar y visualizar tendencias del mercado inmobiliario a lo largo del tiempo.
- **Predicción de Demanda:** Desarrollar modelos para predecir la demanda futura de apartamentos en diferentes áreas, basados en factores económicos, demográficos y sociales.
- **Simulación de Escenarios:** Crear simulaciones que permitan a los usuarios visualizar cómo diferentes factores (e.g., políticas gubernamentales, desarrollos urbanos) podrían impactar los precios inmobiliarios en el futuro.

Estas propuestas no solo buscan mejorar la funcionalidad y precisión de SkyPrice, sino también explorar nuevas oportunidades y aplicaciones de la plataforma, asegurando su relevancia y utilidad continua en un mercado inmobiliario dinámico y en constante evolución.

Anexos

.1. Código fuente para preprocesamiento de datos y entrenamiento de modelos

```
1 # @Author: Humberto Alejandro Ortega Alcocer
2 # @Date: 2024-Abril-8
3 # @Description: Script para entrenar modelos de aprendizaje automático y
4 # guardarlos en archivos
5 # joblib y h5.
6 # @Usage: python entrenar-modelos.py
7 # @Output: Archivos rf_model.joblib, svm_model.joblib, nn_model.h5,
8 # preprocessor.joblib
9 # con los modelos entrenados y el preprocesador.
10 # @Dependencies: pandas, numpy, scikit-learn, tensorflow, joblib
11 # @Dataset: dataset.csv
12 import pandas as pd
13 from sklearn.model_selection import train_test_split, GridSearchCV
14 from sklearn.preprocessing import StandardScaler, OneHotEncoder
15 from sklearn.compose import ColumnTransformer
16 from sklearn.pipeline import Pipeline
17 from sklearn.ensemble import RandomForestRegressor
18 from sklearn.svm import SVR
19 from sklearn.metrics import mean_absolute_error
20 from joblib import dump
21 from constants import *
22 import numpy as np
23 from scikeras.wrappers import KerasRegressor
24 import keras as keras
25 from fuzzywuzzy import process
26
27 # Mensaje de inicio
28 print("Script de entrenamiento de modelos")
29 print("Cargando datos...")
30
31 # Cargar el dataset
32 df = pd.read_csv(ARCHIVO_DATASET)
33 print(f'Dimensiones del dataset: {df.shape}')
34
35 # Convertir aquellos precios en USD a pesos mexicanos usando la tasa de
36 # cambio en constants
37 df.loc[df['Currency'] == 'USD', 'Price'] *= np.int64(TASA_CAMBIO_USD_MXN)
38
39 # Convertir 'USD' a 'MN' en la columna 'Currency'
40 df.loc[df['Currency'] == 'USD', 'Currency'] = 'MN'
41 print(f'Valores únicos en Currency: \n{df["Currency"].unique()}')
42
43 # Eliminar filas duplicadas a partir de la columna 'ID'
44 #df.drop_duplicates(subset='ID', inplace=True)
45 print(f'Dimensiones del dataset después de eliminar duplicados: {df.shape}')
46
47 # Eliminar columnas no numéricas
48 columnas_numericas = ['Price', 'Size_Terrain', 'Size_Construction', 'Rooms',
49 'Bathrooms', 'Parking', 'Age', 'Lat', 'Lng']
50 df = df[[ 'Municipality', *columnas_numericas]]
51 print(f'Columnas después de eliminar columnas no numéricas: {df.columns}')
52
53 # Validar que Municipality sea una alcalía de la CDMX, o tratar de
54 # aproximarla, si no es posible, eliminar la fila
55
56 # Función para ajustar los nombres de las alcaldías al valor más cercano de
57 # la lista
```

```

52 def ajustar_municipality(nombre, lista_alcaldias):
53     # Encuentra la coincidencia más cercana en la lista
54     coincidencia, _ = process.extractOne(nombre, lista_alcaldias)
55     return coincidencia
56
57 # Aplica la función a la columna 'Municipality',
58 df['Municipality'] = df['Municipality'].apply(lambda x: ajustar_municipality
59     (x, municipalities))
60
61 print(f'Dimensiones del dataset después de filtrar por alcaldías: {df.shape}
62 ')
63
64 # Sustituir infinitos por NaN
65 #df.replace([np.inf, -np.inf], np.nan, inplace=True)
66 print(f'Valores nulos después de sustituir infinitos: \n{df.isnull().sum()}')
67
68 # Eliminar columnas no numéricas
69 for columna in columnas_numericas:
70     df[columna] = pd.to_numeric(df[columna], errors='coerce')
71 print(f'Dimensiones del dataset después de eliminar columnas no numéricas: {df.shape}')
72
73 # Sustituir NaN por la media en columnas numéricas de baja cardinalidad
74 for columna in ['Size_Terrain', 'Size_Construction', 'Rooms', 'Bathrooms', ,
75     'Parking', 'Age', 'Lat', 'Lng']:
76     if df[columna].isnull().sum() > 0:
77         df[columna].fillna(df[columna].mean())
78 print(f'Valores nulos después de sustituir NaN por la media: \n{df.isnull().sum()}')
79
80 # Eliminar filas con NaN
81 df.dropna(inplace=True)
82 print(f'Dimensiones del dataset después de eliminar NaN: {df.shape}')
83
84 # Algunos valores en 'Age' no corresponden a la antigüedad sino al año de
85 # construcción, identificarlos y corregirlos
86 year = pd.to_datetime('today').year
87 df['Age'] = df['Age'].apply(lambda x: year - x if x > 1000 else x)
88
89 # Filtrar filas que cumplen con los rangos deseados
90 df = df[
91     (df['Price'] >= MIN_PRICE) & (df['Price'] <= MAX_PRICE) &
92     (df['Size_Terrain'] >= MIN_SIZE_TERRAIN) & (df['Size_Terrain'] <=
93     MAX_SIZE_TERRAIN) &
94     (df['Size_Construction'] >= MIN_SIZE_CONSTRUCTION) & (df['
95     Size_Construction'] <= MAX_SIZE_CONSTRUCTION) &
96     (df['Rooms'] >= MIN_ROOMS) & (df['Rooms'] <= MAX_ROOMS) &
97     (df['Bathrooms'] >= MIN_BATHROOMS) & (df['Bathrooms'] <= MAX_BATHROOMS) &
98     (df['Parking'] >= MIN_PARKING) & (df['Parking'] <= MAX_PARKING) &
99     (df['Age'] >= MIN_AGE) & (df['Age'] <= MAX_AGE) &
100    (df['Lat'] >= MIN_LAT) & (df['Lat'] <= MAX_LAT) &
101    (df['Lng'] >= MIN_LNG) & (df['Lng'] <= MAX_LNG)
102 ]
103 print(f'Estadísticas después de aplicar límites:\n {df.describe()}')
104
105 print("Valor FAST_TEST: ", FAST_TEST)
106 print("Evaluación de FAST_TEST: ", FAST_TEST and "rápida" or "lenta")
107 print("Valor TEST_SIZE: ", TEST_SIZE)
108
109 # Esperar a que el usuario presione una tecla

```

```

104 input("Presiona una tecla para continuar...")
105
106 # Dividir el dataset en conjuntos de entrenamiento y prueba
107 X = df[['Municipality', 'Size_Terrain', 'Size_Construction', 'Rooms',
108         'Bathrooms', 'Parking', 'Age', 'Lat', 'Lng']]
109 y = df['Price']
110 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=
111             TEST_SIZE, random_state=RANDOM_STATE)
112 print(f'Dimensiones de los conjuntos de entrenamiento y prueba: {X_train.
113     shape}, {X_test.shape}')
114
115 # Guardar los conjuntos de entrenamiento y prueba
116 X_train.to_csv(ARCHIVO_X_TRAIN, index=False)
117 X_test.to_csv(ARCHIVO_X_TEST, index=False)
118 y_train.to_csv(ARCHIVO_Y_TRAIN, index=False)
119 y_test.to_csv(ARCHIVO_Y_TEST, index=False)
120 print(f'Conjuntos de entrenamiento y prueba guardados en {ARCHIVO_X_TRAIN},
121       {ARCHIVO_X_TEST}, {ARCHIVO_Y_TRAIN}, {ARCHIVO_Y_TEST}')
122
123 # Crear un procesador para las columnas numéricas
124 numeric_transformer = Pipeline(steps=[
125     ('scaler', StandardScaler())
126 ])
127
128 # Obtener las columnas numéricas
129 numeric_features=columnas_numericas[1:]
130
131 # Crear un procesador para las columnas categóricas
132 preprocessor = ColumnTransformer(
133     transformers=[
134         ('cat', OneHotEncoder(categories=[municipalities]), ['Municipality',
135                               ]),
136         ('num', numeric_transformer, numeric_features),
137     ])
138
139 # Mensaje de inicio de entrenamiento
140 print("Entrenando modelos...")
141
142 # Redes Neuronales
143 print("Entrenando Red Neuronal...")
144
145 # Preprocesar los datos para la red neuronal
146 X_train_preprocessed = preprocessor.fit_transform(X_train)
147
148 # Función para obtener el modelo de red neuronal
149 def get_reg(meta, hidden_layer_sizes, dropout):
150     n_features_in_ = meta["n_features_in_"]
151     model = keras.models.Sequential()
152     model.add(keras.layers.Input(shape=(n_features_in_,)))
153     for hidden_layer_size in hidden_layer_sizes:
154         model.add(keras.layers.Dense(hidden_layer_size, activation="relu"))
155         model.add(keras.layers.Dropout(dropout))
156     model.add(keras.layers.Dense(1))
157     return model
158
159 # Crear el modelo de red neuronal
160 reg = KerasRegressor(
161     model=get_reg,
162     loss=keras.losses.MeanSquaredError,
163     optimizer=keras.optimizers.Adam,
164     metrics=[keras.metrics.R2Score],
165     hidden_layer_sizes=(100,),
166

```

```

161     dropout=0.5,
162 )
163
164 # Parámetros a buscar
165 param_grid_nn_fast = {
166     "hidden_layer_sizes": [(10, 10)],
167     "dropout": [0.5],
168     "batch_size": [50],
169     "epochs": [50],
170     "optimizer": [keras.optimizers.Adam],
171     "optimizer_learning_rate": [0.1],
172     "loss": [keras.losses.MeanSquaredError],
173     "metrics": [keras.metrics.R2Score],
174 }
175 param_grid_nn_slow = {
176     "hidden_layer_sizes": [(10, 10), (10, 10, 10), (30, 30)],
177     "dropout": [0.5, 0.7],
178     "batch_size": [50, 100],
179     "epochs": [50, 100, 200, 500],
180     "optimizer": [keras.optimizers.Adam],
181     "optimizer_learning_rate": [0.1, 0.01, 0.001],
182     "loss": [keras.losses.MeanSquaredError],
183     "metrics": [keras.metrics.R2Score],
184 }
185 param_grid_nn = FAST_TEST and param_grid_nn_fast or param_grid_nn_slow
186
187 # Crear el objeto GridSearchCV
188 grid_search_nn = GridSearchCV(
189     estimator=reg,
190     param_grid=param_grid_nn,
191     refit=False,
192     cv=5,
193     verbose=2,
194     n_jobs=-1,
195 )
196
197 # Debido a que el preprocesamiento ya está hecho, podemos usar
# X_train_preprocessed directamente aquí
198 grid_search_nn.fit(X_train_preprocessed, y_train)
199
200 # Obtener los mejores parámetros
201 mejores_parametros = grid_search_nn.best_params_
202
203 # Imprimir los mejores Parámetros
204 print(f"\t- Mejores parámetros: {mejores_parametros}")
205
206 # Crear el modelo y compilarlo con los mejores parámetros
207 opt = keras.optimizers.Adam(learning_rate=mejores_parametros[
208     "optimizer_learning_rate"])
209 nn_model = get_reg({"n_features_in_": X_train_preprocessed.shape[1]},
210                     mejores_parametros["hidden_layer_sizes"],
211                     mejores_parametros["dropout"])
212 nn_model.compile(
213     optimizer=opt,
214     loss=keras.losses.MeanSquaredError(),
215     metrics=[keras.metrics.R2Score]
216     ())
217 nn_model_history = nn_model.fit(X_train_preprocessed,
218                                 y_train,
219                                 batch_size=mejores_parametros["batch_size"],
220                                 epochs=mejores_parametros["epochs"],
221                                 verbose='2')
222
223 print("Modelo Red Neuronal entrenado")

```

```

219 # Random Forest
220 print("Entrenando Random Forest...")
221 # Crear el pipeline
222 rf_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
223                             ('regressor', RandomForestRegressor(
224                               random_state=42, oob_score=True, n_jobs=-1))])
225
226 # Parámetros a buscar
227 param_grid_rf_fast = {
228     'regressor__n_estimators': [100],
229     'regressor__max_depth': [None],
230     'regressor__min_samples_split': [2],
231 }
232 param_grid_rf_slow = {
233     'regressor__n_estimators': [100, 500, 1000],
234     'regressor__max_depth': [None, 10, 20, 30],
235     'regressor__min_samples_split': [2, 5, 10]
236 }
237 param_grid_rf = FAST_TEST and param_grid_rf_fast or param_grid_rf_slow
238
239 # Crear el objeto GridSearchCV
240 grid_search_rf = GridSearchCV(rf_pipeline, param_grid=param_grid_rf, cv=5,
241                               verbose=2, n_jobs=-1)
242
243 # Ejecutar la búsqueda
244 grid_search_rf.fit(X_train, y_train)
245
246 # Obtener el mejor modelo
247 rf_pipeline = grid_search_rf.best_estimator_
248
249 # Imprimir los mejores parámetros
250 print(f"\t- Mejores parámetros: {grid_search_rf.best_params_}")
251 print("Modelo Random Forest entrenado")
252
253 # Máquina de Soporte Vectorial
254 print("Entrenando SVM...")
255 # Crear el pipeline
256 svm_pipeline = Pipeline(steps=[('preprocessor', preprocessor),
257                               ('svr', SVR())])
258
259 # Parámetros a buscar
260 param_grid_svm_fast = {
261     'svr__C': [0.1],
262     'svr__gamma': ['scale'],
263     'svr__epsilon': [0.01],
264     'svr__kernel': ['rbf']
265 }
266 param_grid_svm_slow = {
267     'svr__C': [0.1, 1, 100, 1000],
268     'svr__gamma': ['scale', 'auto', 0.01, 0.001],
269     'svr__epsilon': [0.01, 0.1, 1, 10],
270     'svr__kernel': ['rbf', 'linear', 'poly', 'sigmoid'],
271 }
272 param_grid_svm = FAST_TEST and param_grid_svm_fast or param_grid_svm_slow
273
274 # Crear el objeto GridSearchCV
275 grid_search_svm = GridSearchCV(svm_pipeline, param_grid=param_grid_svm, cv
276                               =5, verbose=2, n_jobs=-1)
277
278 # Ejecutar la búsqueda
279 grid_search_svm.fit(X_train, y_train)

```

```

278
279 # Obtener el mejor modelo
280 svm_pipeline = grid_search_svm.best_estimator_
281
282 # Imprimir los mejores parámetros
283 print(f"\t- Mejores parámetros: {grid_search_svm.best_params_}")
284 print("Modelo SVM entrenado")
285
286 # Guardar los modelos
287 print("Guardando modelos...")
288 dump(rf_pipeline, ARCHIVO_MODELO_RF)
289 dump(svm_pipeline, ARCHIVO_MODELO_SVM)
290 dump(nn_model, ARCHIVO_MODELO_RN)
291 dump(nn_model_history, ARCHIVO_MODELO_RN_HISTORIA)
292 dump(preprocessor, ARCHIVO_PREPROCESADOR)
293 print(f"Modelos guardados en archivos {ARCHIVO_MODELO_RF}, {
294     ARCHIVO_MODELO_SVM}, {ARCHIVO_MODELO_RN}, {ARCHIVO_MODELO_RN_HISTORIA},
295     {ARCHIVO_PREPROCESADOR}")
296
297 # Validar los modelos e imprimir sus resultados
298 def validate_model(model, X, y, name):
299     predictions = model.predict(X)
300     mae = mean_absolute_error(y, predictions)
301     print(f'\t- MAE de {name}: {mae}')
302
303 print("Validando modelos...")
304 validate_model(rf_pipeline, X_test, y_test, 'Random Forest')
305 validate_model(svm_pipeline, X_test, y_test, 'SVM')
306 validate_model(nn_model, preprocessor.transform(X_test), y_test, 'Red
307 Neuronal')
308 print("Modelos validados")
309
310 # Fin del script
311 print("Fin del script")

```

.2. Código fuente de la API

```
1 # @Author: Humberto Alejandro Ortega Alcocer
2 # @Date: 2024-Abril-8
3 # @Description: API para predecir el precio de una propiedad utilizando tres
   modelos de aprendizaje automático: Random Forest, SVM y Redes
   Neuronales.
4 # @Dependencies: joblib, tensorflow, pydantic, pandas, matplotlib, tempfile,
   fastapi
5 # @Usage: uvicorn main:app --reload
6 # @URL: http://localhost:8000
7 # @Docs: http://localhost:8000/openapi
8 # @Redoc: http://localhost:8000/redoc
9 # @License: MIT
10 from fastapi.middleware.cors import CORSMiddleware # Para configurar CORS
11 import datetime # Para obtener la fecha y hora actual
12 from joblib import load # Para cargar los modelos de aprendizaje automático
13 from sklearn.metrics import mean_absolute_error, mean_squared_error,
   root_mean_squared_error # Para calcular el error absoluto medio
14 from pydantic import BaseModel, Field # Para definir la clase de la
   propiedad y sus campos
15 import pandas as pd # Para trabajar con los datos de la propiedad
16 from fastapi import FastAPI # Para crear la aplicación de FastAPI
17 from fastapi.responses import FileResponse # Para servir archivos estáticos
18 import matplotlib # Para configurar el renderizador
19 import matplotlib.pyplot as plt # Para generar gráficas
20 import logging # Para mostrar mensajes de depuración
21 from constants import *
22
23 # Mensaje de depuración
24 logging.basicConfig(level=logging.INFO)
25 logging.info("Cargando modelos de aprendizaje automático...")
26
27 # Elegimos usar el renderizador "Agg" que no requiere usar el entorno grá
   fico
28 # de nuestro Sistema Operativo.
29 matplotlib.use('Agg')
30
31 # Carga de modelos
32 rf_model = load(ARCHIVO_MODELO_RF)
33 svm_model = load(ARCHIVO_MODELO_SVM)
34 nn_model = load(ARCHIVO_MODELO_RN)
35 nn_model_history = load(ARCHIVO_MODELO_RN_HISTORIA)
36 preprocessor = load(ARCHIVO_PREPROCESADOR)
37
38 logging.info("Modelos de aprendizaje automático cargados con éxito")
39 logging.info("Cargando datos de entrenamiento y prueba...")
40
41 # Carga de datos de entrenamiento y prueba
42 X_train = pd.read_csv(ARCHIVO_X_TRAIN)
43 X_test = pd.read_csv(ARCHIVO_X_TEST)
44 y_train = pd.read_csv(ARCHIVO_Y_TRAIN)
45 y_test = pd.read_csv(ARCHIVO_Y_TEST)
46
47 # Carga del dataset original
48 df = pd.read_csv(ARCHIVO_DATASET)
49
50 logging.info("Datos de entrenamiento y prueba cargados con éxito")
51 logging.info("Evaluando los modelos de aprendizaje automático...")
52
53 # Predecir con cada modelo
```

```

54 rf_preds = rf_model.predict(X_test)
55 svm_preds = svm_model.predict(X_test)
56 nn_preds = nn_model.predict(preprocessor.transform(X_test)).flatten()
57
58 # Calcular el error cuadrático medio de cada modelo
59 rf_mse = mean_squared_error(y_test, rf_preds)
60 svm_mse = mean_squared_error(y_test, svm_preds)
61 nn_mse = mean_squared_error(y_test, nn_preds)
62
63 # Calcular el RMSE (raíz del error cuadrático medio) de cada modelo
64 rf_rmse = root_mean_squared_error(y_test, rf_preds)
65 svm_rmse = root_mean_squared_error(y_test, svm_preds)
66 nn_rmse = root_mean_squared_error(y_test, nn_preds)
67
68 # Calcular los intervalos de confianza de cada modelo
69 rf_ci = (rf_preds - y_test.squeeze()).mean() - 1.96 * (rf_preds - y_test.
    squeeze()).std(), (rf_preds - y_test.squeeze()).mean() + 1.96 * (
    rf_preds - y_test.squeeze()).std()
70 svm_ci = (svm_preds - y_test.squeeze()).mean() - 1.96 * (svm_preds - y_test.
    squeeze()).std(), (svm_preds - y_test.squeeze()).mean() + 1.96 * (
    svm_preds - y_test.squeeze()).std()
71 nn_ci = (nn_preds - y_test.squeeze()).mean() - 1.96 * (nn_preds - y_test.
    squeeze()).std(), (nn_preds - y_test.squeeze()).mean() + 1.96 * (
    nn_preds - y_test.squeeze()).std()
72
73 # Calculamos el error absoluto medio de cada modelo
74 rf_mae = mean_absolute_error(y_test, rf_preds)
75 svm_mae = mean_absolute_error(y_test, svm_preds)
76 nn_mae = mean_absolute_error(y_test, nn_preds)
77
78 # Calculamos el coeficiente de determinación de cada modelo
79 rf_r2 = rf_model.score(X_test, y_test)
80 svm_r2 = svm_model.score(X_test, y_test)
81 nn_r2 = nn_model_history.history['r2_score'][-1]
82
83 logging.info("Modelos de aprendizaje automático evaluados con éxito")
84 logging.info("Generando gráficas de predicciones vs valores reales...")
85
86 # Generación de gráficas
87 models = {'Random Forest': rf_preds, 'SVM': svm_preds, 'Neural Network':
    nn_preds}
88 axs = plt.subplots(1, 3, figsize=(15, 5))[1]
89
90 # Graficar las predicciones vs valores reales
91 for ax, (model_name, preds) in zip(axs, models.items()):
92     ax.scatter(y_test, preds, alpha=0.5)
93     ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--',
94             lw=2)
95     ax.set_xlabel('Valores Reales')
96     ax.set_ylabel('Predicciones')
97     ax.set_title(model_name)
98
99 # Ajustar el espacio entre las gráficas
100 plt.tight_layout()
101
102 # Servimos la gráfica sin tocar el disco
103 plt.savefig('plots.png')
104
105 # Cerramos la gráfica
106 plt.close('all')
107 logging.info("Gráficas generadas con éxito")

```

```

108 # Inicialización de la aplicación
109 app = FastAPI(title="SkyPrice API",
110                 description="Hola, bienvenido a la API de **SkyPrice**. Aquí
111                 puedes predecir el precio de una propiedad utilizando tres modelos de
112                 aprendizaje automático: Random Forest, SVM y Redes Neuronales. Para
113                 predecir el precio de una propiedad, envía una solicitud POST a la ruta
114                 /predict con los datos de la propiedad. También puedes obtener informaci
115                 ón sobre los modelos y sus características en la ruta /models. ¡Divié
116                 rtete!",
117                 version="1.0.0",
118                 docs_url="/openapi",
119                 openapi_url="/openapi.json",
120                 redoc_url="/redoc",
121                 summary="API para predecir el precio de un departamento en
122                 la CDMX",
123                 contact={
124                     "name": "Humberto Alejandro Ortega Alcocer",
125                     "email": "hortegaa1500@alumno.ipn.mx",
126                     "url": "https://humbertowoodly.xyz",
127                 },
128                 license_info={
129                     "name": "MIT",
130                     "url": "https://opensource.org/licenses/MIT",
131                 },
132                 terms_of_service="https://opensource.org/licenses/MIT",
133                 servers=[
134                     {
135                         "url": f"{HOSTNAME}",
136                         "description": "URL de la API"
137                     }
138                 ]
139             )
140
141             # Definición de una propiedad
142             class Property(BaseModel):
143                 Size_Terrain: float = Field(..., examples=[120.5], description="El tamañ
144                 o del terreno en metros cuadrados")
145                 Size_Construction: float = Field(..., examples=[250.0], description="El
146                 tamaño de la construcción en metros cuadrados")
147                 Rooms: int = Field(..., examples=[3], description="El número de
148                 habitaciones")
149                 Bathrooms: float = Field(..., examples=[2.5], description="El número de
150                 baños")
151                 Parking: int = Field(..., examples=[1], description="El número de
152                 espacios de estacionamiento disponibles")
153                 Age: int = Field(..., examples=[5], description="La edad de la propiedad
154                 en años")
155                 Lat: float = Field(..., examples=[19.432608], description="La latitud de
156                 la propiedad")
157                 Lng: float = Field(..., examples=[-99.133209], description="La longitud
158                 de la propiedad")

```

```

154     Municipality: str = Field(..., examples=["Benito Juárez"], description="El municipio donde se encuentra la propiedad")
155
156 # Definición de la respuesta del endpoint principal
157 class PrincipalResponse(BaseModel):
158     message: str = Field(..., examples=["Bienvenido a la API de predicción inmobiliaria"], description="Mensaje de bienvenida")
159     time: datetime.datetime = Field(..., examples=[datetime.datetime.now()], description="La hora actual del servidor")
160     version: str = Field(..., examples=["0.1"], description="La versión de la API")
161     description: str = Field(..., examples=["API para predecir el precio de una propiedad"], description="Descripción de la API")
162     openapi: str = Field(..., examples=[f"{HOSTNAME}/openapi"], description="Enlace a la documentación de la API")
163     redoc: str = Field(..., examples=[f"{HOSTNAME}/redoc"], description="Enlace a la documentación de la API en formato ReDoc")
164
165 # Definición de la respuesta del endpoint de predicciones
166 class PredictResponse(BaseModel):
167     random_forest: float = Field(..., examples=[2500000.0], description="La predicción del precio de la propiedad con el modelo Random Forest")
168     svm: float = Field(..., examples=[2700000.0], description="La predicción del precio de la propiedad con el modelo SVM")
169     neural_network: float = Field(..., examples=[2600000.0], description="La predicción del precio de la propiedad con el modelo de Redes Neuronales")
170
171 # Definición de la respuesta del endpoint de modelos
172 class ModelsResponse(BaseModel):
173     dataset: dict = Field(..., examples=[{"original": (1000,8), "training": {"X": (1000, 8), "y": (1000, 1)}, "testing": {"X": (250, 8), "y": (250, 1)}}], description="Información sobre los datos de entrenamiento y prueba")
174     models: dict = Field(..., examples=[{"random_forest": {"mse": 10000000000.0, "ci": (9000000000.0, 11000000000.0), "mae": 30000.0, "r2": 0.9, "feature_importances": [0.1, 0.2, 0.3, 0.4, 0.0, 0.0, 0.0, 0.0], "max_features": "auto", "max_depth": 10, "n_estimators": 100, "oob_score": True}, "svm": {"mse": 15000000000.0, "ci": (14000000000.0, 16000000000.0), "mae": 40000.0, "r2": 0.8, "kernel": "rbf", "C": 1.0, "epsilon": 0.1}, "neural_network": {"mse": 20000000000.0, "ci": (19000000000.0, 21000000000.0), "mae": 50000.0, "r2": 0.7, "learning_rate": 0.001, "beta_1": 0.9, "beta_2": 0.999, "epsilon": 1e-07}}], description="Información sobre los modelos y sus características")
175
176 # Ruta principal
177 @app.get("/", summary="Ruta principal", description="Ruta principal de la API de predicción inmobiliaria.", tags=["Principal"], response_description="Mensaje de bienvenida", response_model=PrincipalResponse)
178 async def principal():
179     # Regresamos un mensaje de bienvenida, la hora actual del servidor, la fecha, la versión del API, la descripción y un link a los docs de swagger.
180     return {"message": "Bienvenido a la API de SkyPrice",
181             "time": datetime.datetime.now(),
182             "version": "1.0.0",
183             "description": "API para predecir el precio de un departamento en la Ciudad de México, si deseas ver la documentación de la API, visita el enlace de OpenAPI (Swagger UI) o ReDoc:",
184             "openapi": f"{HOSTNAME}/openapi",
185             "redoc": f"{HOSTNAME}/redoc"}

```

```

186
187
188 # Ruta para obtener las gráficas de predicciones vs valores reales
189 @app.get("/plots", response_class=FileResponse, summary="Obtener gráficas de
    predicciones vs valores reales", description="Obtener las gráficas de
    predicciones vs valores reales de los modelos Random Forest, SVM y Redes
    Neuronales.", tags=["Gráficas"], response_description="Gráficas de
    predicciones vs valores reales")
190 async def plots():
191     return FileResponse('plots.png')
192
193
194 # Ruta para predecir el precio de una propiedad
195 @app.post("/predict", summary="Predecir el precio de una propiedad",
    description="Predecir el precio de una propiedad utilizando tres modelos
        de aprendizaje automático: Random Forest, SVM y Redes Neuronales.",
    tags=["Predicciones"], response_description="Predicciones del precio de
        la propiedad", response_model=PredictResponse)
196 async def predict(property: Property):
197     # Convertir entrada a DataFrame para preprocesamiento
198     input_data = [property.Municipality, property.Size_Terrain, property.
        Size_Construction, property.Rooms, property.Bathrooms, property.Parking,
        property.Age, property.Lat, property.Lng]
199     input_df = pd.DataFrame([input_data], columns=['Municipality', ,
        'Size_Terrain', 'Size_Construction', 'Rooms', 'Bathrooms', 'Parking', ,
        'Age', 'Lat', 'Lng'])
200
201     # Predecir con cada modelo
202     rf_pred,      = rf_model.predict(input_df)
203     svm_pred = svm_model.predict(input_df)[0]
204     # Preprocesar para la red neuronal y luego predecir
205     nn_input = preprocessor.transform(input_df)
206     nn_pred = nn_model.predict(nn_input)[0][0]
207
208     # Devolver las predicciones
209     return {
210         "random_forest": float(rf_pred),
211         "svm": float(svm_pred),
212         "neural_network": float(nn_pred)
213     }
214
215 # Ruta para listar los modelos y sus características (ajuste de datos de
    entrenamiento, hiperparámetros, etc.)
216 @app.get("/models", summary="Obtener información sobre los modelos y sus
    características", description="Obtener información sobre los modelos de
    aprendizaje automático utilizados en la API, incluyendo sus características,
    ajuste de datos de entrenamiento, hiperparámetros, etc.", tags=[

    "Modelos"], response_description="Información sobre los modelos y sus
    características", response_model=ModelsResponse)
217 async def models_info():
218     # Regresar información sobre los modelos y sus características
219     return {
220         "dataset": {
221             "original": df.shape,
222             "training": {
223                 "X": X_train.shape,
224                 "y": y_train.shape
225             },
226             "testing": {
227                 "X": X_test.shape,
228                 "y": y_test.shape
229             }
230     }

```

```

230     },
231     "models": {
232         "random_forest": {
233             "mse": rf_mse,
234             "rmse": rf_rmse,
235             "ci": rf_ci,
236             "mae": rf_mae,
237             "r2": rf_r2,
238             "feature_importances": rf_model['regressor'].
239             feature_importances_.tolist(),
240             "max_features": rf_model['regressor'].max_features,
241             "max_depth": rf_model['regressor'].max_depth,
242             "n_estimators": rf_model['regressor'].n_estimators,
243             "oob_score": rf_model['regressor'].oob_score,
244         },
245         "svm": {
246             "mse": svm_mse,
247             "rmse": svm_rmse,
248             "ci": svm_ci,
249             "mae": svm_mae,
250             "r2": svm_r2,
251             "kernel": svm_model['svr'].kernel,
252             "C": svm_model['svr'].C,
253             "epsilon": svm_model['svr'].epsilon
254         },
255         "neural_network": {
256             "mse": nn_mse,
257             "rmse": nn_rmse,
258             "ci": nn_ci,
259             "mae": nn_mae,
260             "r2": nn_r2,
261             "learning_rate": float(nn_model.optimizer.learning_rate.
262             numpy()),
263             "beta_1": nn_model.optimizer.beta_1,
264             "beta_2": nn_model.optimizer.beta_2,
265             "epsilon": nn_model.optimizer.epsilon
266         }
267     }
268 }
```

Glosario

Dataset Conjunto de datos que se utilizan para entrenar un modelo de aprendizaje automático [27]. 15, 20, 33, 34

Inmueble Todos los intereses, beneficios, derechos y gravámenes inherentes a la propiedad de bienes raíces físicos, donde los bienes raíces son la tierra junto con todas las mejoras que están permanentemente adheridas a ella y todos los accesorios asociados a la misma [3]. 23, 26

Valuación Valuación es la acción o efecto de valuar, a su vez, significa valorar, entendido como el reconocimiento o apreciación del valor de algo [119]. 23

Acrónimos

ARIMA Modelo Autoregresivo de Media Móvil. 27

AVM Modelo de Valuación Automatica. 27

BID Banco Interamericano de Desarrollo. 26

CNBV Comisión Nacional Bancaria y de Valores. 25

CSV Comma Separated Values (Valores Separados por Comas). 38

DEPS Density Estimation and Profit Simulation. 27

IA Inteligencia Artificial. 27

INDAABIN Instituto de Administración y Avalúos de Bienes Nacionales.
24

IVSC International Valuation Standards Council. 25

LFPDPPP Ley Federal de Protección de Datos Personales en Posesión de
los Particulares. 95, 102

RNA Redes Neuronales Artificiales. 27, 30, 33, 34

SEDECO Secretaría de Desarrollo Económico. 25

SHN Sociedad Hipotecaria Nacional. 25

Bibliografía

- [1] H. Alcocer, “Skyprice api docker repository,” April 2024, accessed: 2024-05-18. [Online]. Available: <https://hub.docker.com/r/humbertowood/skyprice-api/tags>
- [2] R. Caruana and A. Niculescu-Mizil, “An empirical comparison of supervised learning algorithms,” in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML ’06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 161–168. [Online]. Available: <https://doi.org/10.1145/1143844.1143865>
- [3] E. Pagourtzi, V. Assimakopoulos, and T. Hatzichristos, “Real estate appraisal: a review of valuation methods,” *J. Property Invest. Finance*, vol. 21, no. 4, pp. 383–401, 2003.
- [4] Z. Nedović-Budić, A. K. Pinto, and L. D. Budic, “Gis database development and exchange: Interaction mechanisms and motivations,” *J. Urban Plan. Dev.*, vol. 126, no. 2, pp. 51–73, 2000.
- [5] E. M. Worzala, M. J. Lenk, and A. Silva, “An exploration of neural networks and its application to real estate valuation,” *J. Real Estate Res.*, vol. 10, no. 2, pp. 185–201, 1995.
- [6] V. Limsombunchai, “House price prediction: Hedonic price model vs. artificial neural network,” *Int. J. Econ. Manag.*, vol. 1, no. 1, pp. 9–30, 2004.
- [7] L. H. T. Choy and W. K. O. Ho, “The use of machine learning in real estate research,” *Land*, vol. 12, no. 4, p. 740, 2023.
- [8] Zillow. (2023) How much is my home worth? Accessed: 24-Apr-2023. [Online]. Available: <https://www.zillow.com/how-much-is-my-home-worth/>

- [9] HouseCanary. (2021) Home. Accessed: 24-Apr-2023. [Online]. Available: <https://www.housecanary.com/>
- [10] ¿cuánto vale? inmuebles24. Accessed: 24-Apr-2023. [Online]. Available: <https://www.inmuebles24.com/cuantovale/>
- [11] K. Schwaber and J. Sutherland, *La Guía Definitiva de Scrum: Las Reglas del Juego*. Scrum Guides, 2016.
- [12] (2023, 10) Trello. [Online]. Available: <https://trello.com/>
- [13] C. J. y de Servicios Legales del DF, “Definición de inmueble,” 2014, el inmueble es aquella cosa que no puede trasladarse de un lugar a otro, por ejemplo, un terreno o edificio. [Online]. Available: <https://data.consejeria.cdmx.gob.mx/index.php/component/glossary/Glosario-ConsejerÃa-1/I/INMUEBLE-85>
- [14] S. O. Organismo Nacional de Normalización y Certificación de la Construcción y Edificación, “Nmx-c-459-scfi-onncce-2007 servicios de valuación,” Secretaría de Economía, Dirección General de Normas, México, D.F., Tech. Rep., 2007, publicada el 27 de agosto de 2007. Entrada en vigor 60 días naturales después de su publicación. No equivalente a ninguna norma internacional. [Online]. Available: <https://www.dof.gob.mx/normasOficiales/2663/seeco/seeco.htm>
- [15] J. M. Salas Tafoya, “La valuación inmobiliaria tradicional: un modelo para repensar,” *Paakat: Revista de Tecnología y Sociedad*, vol. 4, no. 6, Marzo–Agosto 2014. [Online]. Available: <https://www.udgvirtual.udg.mx/paakat//index.php/paakat/article/view/220/324#b8>
- [16] C. N. B. y de Valores (CNBV), “Anexo 42 cub,” 2023. [Online]. Available: <https://www.cnbv.gob.mx/Anexos/Anexo%2042%20CUB.pdf>
- [17] Inmuebles24, “Certificación de agente inmobiliario en méxico,” 2023. [Online]. Available: <https://www.inmuebles24.com/noticias/sabias-que/certificacion-agente-inmobiliario-mexico/>
- [18] Mexhome, “Valuación de propiedades en méxico: Una guía completa,” 2021. [Online]. Available: <https://www.mexhome.com/valuacion-de-propiedades-en-mexico/>
- [19] H. Eguino, D. Erba, E. Da Silva, M. Piumetto, T. Iturre, A. Rodríguez, and A. De Oliveira, “Catastro, valoración inmobiliaria y tributación municipal: Experiencias para mejorar su articulación y efectividad,”

Jun 2020, editor: Eguino, Huáscar; Erba, Diego. [Online]. Available: <http://dx.doi.org/10.18235/0002437>

- [20] R. J. Shiller and A. N. Weiss, “Evaluating real estate valuation systems,” *The Journal of Real Estate Finance and Economics*, vol. 18, no. 2, pp. 147–161, 1999. [Online]. Available: <https://doi.org/10.1023/A:1007756607862>
- [21] J. McCarthy, “What is artificial intelligence?” 2004.
- [22] A. Moreno, E. Armengol, J. Béjar Alonso, L. A. Belanche Muñoz, C. U. Cortés García, R. Gavaldà Mestre, J. M. Gimeno, M. Martín Muñoz, and M. Sánchez-Marrè, “Aprendizaje automático,” 1994.
- [23] MathWorks España, “Machine learning in matlab,” <https://es.mathworks.com/help/stats/machine-learning-in-matlab.html>, 2023, accedido en 26-Noviembre-2023.
- [24] P. Cunningham, M. Cord, and S. J. Delany, *Supervised Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–49. [Online]. Available: https://doi.org/10.1007/978-3-540-75171-7_2
- [25] E. Grossi and M. Buscema, “Introduction to artificial neural networks,” *European Journal of Gastroenterology & Hepatology*, vol. 19, no. 12, pp. 1046–1054, Dec 2007.
- [26] I. Vida, “Imagen en: Neural dendrites reveal their computational power,” <https://www.quantamagazine.org/neural-dendrites-reveal-their-computational-power-20200114/>, 2020, neuroCure Cluster, Charité – Universitätsmedizin Berlin.
- [27] IBM. (2023) Neural networks. [Online]. Available: <https://www.ibm.com/topics/neural-networks>
- [28] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, “Scikit-learn: Machine learning in python,” *the Journal of machine Learning research*, vol. 12, pp. 2825–2830, 2011.
- [29] S. learn developers, “Scikit-learn: Machine learning in python,” <https://scikit-learn.org/stable/>, 2023, herramientas simples y eficientes para el análisis predictivo de datos, accesibles para todos y reutilizables en varios contextos. Construido sobre NumPy, SciPy, y matplotlib. Código abierto, utilizable comercialmente - Licencia BSD.

- [30] M. Abadi, “Tensorflow: learning functions at scale,” in *Proceedings of the 21st ACM SIGPLAN international conference on functional programming*, 2016, pp. 1–1.
- [31] K. Team, “About keras,” 2023, accedido el 26 de noviembre de 2023. [Online]. Available: <https://keras.io/about/>
- [32] Agencia Digital de Innovación Pública, “Sistema abierto de información geográfica (sigcdmx),” <https://sig.cdmx.gob.mx/>, disponible en.
- [33] D. S. Sirisuriya *et al.*, “A comparative study on web scraping,” 2015.
- [34] “Selenium webdriver,” <https://www.selenium.dev/documentation/webdriver/>, 2023, accedido: 26-Nov-2023.
- [35] “Beautiful soup documentation,” <https://www.crummy.com/software/BeautifulSoup/bs4/doc/#module-bs4>, 2023, accedido: 26-Nov-2023.
- [36] “Scrapy overview,” <https://docs.scrapy.org/en/latest/intro/overview.html>, 2023, accedido: 26-Nov-2023.
- [37] “What curl does,” <https://everything.curl.dev/project/does>, 2023, accedido: 26-Nov-2023.
- [38] Y. Shafrazi, “Common format and mime type for comma-separated values (csv) files,” Tech. Rep., 2005.
- [39] “Web service - wikipedia,” [Online; accessed 26-November-2023]. [Online]. Available: https://en.wikipedia.org/wiki/Web_service
- [40] “What are web services? - geeksforgeeks,” [Online; accessed 26-November-2023]. [Online]. Available: <https://www.geeksforgeeks.org/what-are-web-services/>
- [41] W. Beltrán, “Diferencia entre api y servicio web,” *Medium*, Mayo 2019. [Online]. Available: <https://medium.com/beltranc/diferencia-entre-api-y-servicio-web-5f204af3aedb>
- [42] “What is python? executive summary | python.org,” [Online; accessed 26-November-2023]. [Online]. Available: <https://www.python.org/doc/essays/blurb/>
- [43] “Welcome to flask — flask documentation (3.0.x),” [Online; accessed 26-November-2023]. [Online]. Available: <https://flask.palletsprojects.com>
- [44] “What is flask python - python tutorial,” [Online; accessed 26-November-2023]. [Online]. Available: <https://pythonbasics.org/what-is-flask-python>

- [45] “Django introduction - learn web development | mdn,” [Online; accessed 26-November-2023]. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>
- [46] “Fastapi - tiangolo,” [Online; accessed 26-November-2023]. [Online]. Available: <https://fastapi.tiangolo.com/>
- [47] “What is a rest api? - red hat,” [Online; accessed 26-November-2023]. [Online]. Available: <https://www.redhat.com/en/topics/api/what-is-a-rest-api>
- [48] Nick, “What is rest api?” *The Iron.io Blog*, November 2020. [Online]. Available: <https://blog.iron.io/what-is-rest-api/>
- [49] “Working with json - learn web development | mdn - mdn web docs,” [Online; accessed 26-November-2023]. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/JSON>
- [50] “Secure sockets layer - ssl shopper,” [Online; accessed 26-November-2023]. [Online]. Available: <https://www.sslshopper.com/what-is-ssl.html>
- [51] “What is https? | cloudflare,” [Online; accessed 26-November-2023]. [Online]. Available: <https://www.cloudflare.com/learning/ssl/what-is-https/>
- [52] IT@Cornell, “Dns definition and basics,” 2022. [Online]. Available: <https://it.cornell.edu/dns/dns-definition-and-basics>
- [53] MDN Web Docs, “Cdn,” 2022. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Glossary/CDN>
- [54] Docker Docs, “Docker overview,” <https://docs.docker.com/get-started/overview/>, 2023, docker es una plataforma abierta para desarrollar, distribuir y ejecutar aplicaciones, que permite separar aplicaciones de infraestructura para entregar software rápidamente.
- [55] M. Jazayeri, “Some trends in web application development,” in *Future of Software Engineering (FOSE '07)*, 2007, pp. 199–213.
- [56] MDN Web Docs, “Javascript | mdn,” 2023. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [57] WHATWG, “Html standard,” 2023. [Online]. Available: <https://html.spec.whatwg.org/>

- [58] World Wide Web Consortium, “Cascading style sheets home page - world wide web consortium (w3c),” 2023. [Online]. Available: <https://www.w3.org/Style/CSS/>
- [59] Bootstrap, “Introduction · bootstrap,” 2023. [Online]. Available: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
- [60] MUI Team, “Getting started with material-ui,” <https://mui.com/material-ui/getting-started/>, 2023, introducción y guía de inicio para Material-UI, un framework de interfaz de usuario para React que implementa Material Design de Google.
- [61] “Quick start – react,” Facebook, 2023, accedido el 26 de noviembre de 2023. [Online]. Available: <https://react.dev/learn>
- [62] “Introduction | vue.js,” 2023, accedido en 26-Noviembre-2023. [Online]. Available: <https://vuejs.org/guide/introduction.html>
- [63] A. Huth and J. Cebula, “The basics of cloud computing,” *United States Computer*, pp. 1–4, 2011.
- [64] A. S. John, “Q1 2023 cloud spending crosses \$63 billion globally,” May 2023, disponible en: <https://wire19.com/cloud-infrastructure-spending/>. [Online]. Available: <https://wire19.com/cloud-infrastructure-spending/>
- [65] J. J. Geewax, *Google Cloud platform in action*. Simon and Schuster, 2018.
- [66] A. Wittig and M. Wittig, *Amazon Web Services in Action: An In-depth Guide to AWS*. Simon and Schuster, 2023.
- [67] Thinkwik, “Everything you wanted to know about amazon web services (aws),” Medium, May 2018, accessed: 2023-11-26. [Online]. Available: <https://medium.com/@thinkwik/everything-you-wanted-to-know-about-amazon-web-services-aws-25376e8462a9>
- [68] A. E. C. Cloud, “Amazon web services,” Retrieved November, vol. 9, no. 2011, p. 2011, 2011.
- [69] P. Burrough, “Gis and geostatistics: Essential partners for spatial analysis,” *Environmental and ecological statistics*, vol. 8, pp. 361–377, 2001.
- [70] J. Evers, “Gis (geographic information system),” National Geographic Society, November 2023, Último acceso: 2023-11-26. [Online]. Available: <https://education.nationalgeographic.org/resource/geographic-information-system-gis/>

- [71] P. J. Wyatt, “The development of a gis-based property information system for real estate valuation,” *International Journal of Geographical Information Science*, vol. 11, no. 5, pp. 435–450, 1997.
- [72] QGIS Development Team, *QGIS 3.28 Desktop User Guide*, QGIS, <https://qgis.org>, 2023, accedido: 26-Nov-2023. [Online]. Available: <https://docs.qgis.org/3.28/pdf/es/QGIS-3.28-DesktopUserGuide-es.pdf>
- [73] D. Alonso. (2023, March) Las novedades más destacadas de qgis 3.30. MappingGIS. Último acceso: 2023-11-26. [Online]. Available: <https://mappinggis.com/2023/03/las-novedades-mas-destacadas-de-qgis-3-30/>
- [74] Kepler.gl, “Welcome to kepler.gl documentation,” 2023, Último acceso: 2023-11-26. [Online]. Available: <https://docs.kepler.gl/>
- [75] “Getting started with google maps platform,” Google for Developers, November 2023, Último acceso: 2023-11-26. [Online]. Available: <https://developers.google.com/maps/get-started>
- [76] (2023) Geocoding api overview. [Online]. Available: <https://developers.google.com/maps/documentation/geocoding/overview>
- [77] G. Van Rossum *et al.*, “Python programming language.” in *USENIX annual technical conference*, vol. 41, no. 1. Santa Clara, CA, 2007, pp. 1–36.
- [78] W. McKinney *et al.*, “pandas: a foundational python library for data analysis and statistics,” *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.
- [79] T. E. Oliphant *et al.*, *Guide to numpy*. Trelgol Publishing USA, 2006, vol. 1.
- [80] S. Seabold and J. Perktold, “Statsmodels: Econometric and statistical modeling with python,” in *Proceedings of the 9th Python in Science Conference*, vol. 57, no. 61. Austin, TX, 2010, pp. 10–25 080.
- [81] F. Voron, *Building Data Science Applications with FastAPI: Develop, manage, and deploy efficient machine learning applications with Python*. Packt Publishing Ltd, 2023.
- [82] S. Morgenthaler, “Exploratory data analysis,” *WIREs Computational Statistics*, vol. 1, no. 1, pp. 33–44, 2009. [Online]. Available: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.2>

- [83] J. H. Stock, M. W. Watson, and R. S. Larrión, “Introducción a la econometría,” 2012.
- [84] M. Greenacre, P. J. Groenen, T. Hastie, A. I. d’Enza, A. Markos, and E. Tuzhilina, “Principal component analysis,” *Nature Reviews Methods Primers*, vol. 2, no. 1, p. 100, 2022.
- [85] G. Biau and E. Scornet, “A random forest guided tour,” *Test*, vol. 25, pp. 197–227, 2016.
- [86] B. Müller, J. Reinhardt, and M. T. Strickland, *Neural networks: an introduction*. Springer Science & Business Media, 1995.
- [87] Amazon Web Services, Inc., “AWS Free Tier,” <https://aws.amazon.com/free>, 2023, accedido: 2023-11-27.
- [88] Congreso de la Unión, “Ley federal de protección de datos personales en posesión de los particulares,” Diario Oficial de la Federación, 2010. [Online]. Available: <https://www.diputados.gob.mx/LeyesBiblio/pdf/LFPDPPP.pdf>
- [89] L. Jacobson and J. R. G. Booch, “The unified modeling language reference manual,” 2021.
- [90] M. García González, Juan Carlos André Ampuero, “Implementación del enfoque de reglas de negocio utilizando motores de reglas en el desarrollo de aplicaciones java,” *Ciencias de la Información*, 2015. [Online]. Available: <https://www.redalyc.org/articulo.oa?id=181439409006>
- [91] M. Cohn, *User stories applied: For agile software development*. Addison-Wesley Professional, 2004.
- [92] X. Li, Z. Liu, and H. Jifeng, “A formal semantics of uml sequence diagram,” in *2004 Australian Software Engineering Conference. Proceedings.*, 2004, pp. 168–177.
- [93] “Functional vs non-functional requirements,” 2022. [Online]. Available: <https://www.geeksforgeeks.org/functional-vs-non-functional-requirements/>
- [94] (2023, November). [Online]. Available: <https://www.inmuebles24.com/departamentos-en-venta-en-ciudad-de-mexico.html>
- [95] scikit-learn developers, *GridSearchCV Documentation*, 2024, accessed: 2024-05-18. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

- [96] I. de Planeación Democrática y Prospectiva de la Ciudad de México, “Centros comerciales, plazas comerciales y tiendas de autoservicio - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/centros-comerciales-plazas-comerciales-y-tiendas-de-autoservicio>
- [97] S. de Desarrollo Económico (SEDECO), “Concentraciones de comerciantes - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/concentraciones-de-comerciantes>
- [98] C. y Procuraduría Ambiental y del Ordenamiento Territorial (PAOT), “Contaminación acústica en ciudad de méxico - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/id/dataset/contaminacion-acustica-en-ciudad-de-mexico>
- [99] ——, “Contaminación de agua en la ciudad de méxico - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: https://datos.cdmx.gob.mx/es_AR/dataset/contaminacion-de-agua-en-la-ciudad-de-mexico
- [100] C. de México, “Códigos postales de la ciudad de méxico - portal de datos abiertos de la cdmx,” 2024, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/ca/dataset/codigos-postales>
- [101] S. de Gestión Integral de Riesgos y Protección Civil, “Escuelas privadas - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/escuelas-privadas>
- [102] S. de Educación Ciencia y Tecnología e Innovación (SECTEI), “Escuelas públicas - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/escuelas-publicas>
- [103] S. de Información y Gestión Educativo (SIGED), “Escuelas de nivel superior en la zona metropolitana del valle de méxico - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/escuelas-de-nivel-superior-en-la-zona-metropolitana-del-valle-de-mexico>
- [104] C. y Instituto de Planeación Democrática y Prospectiva (IPDP), “Grado de marginalidad y violencia urbana por colonia en la ciudad de méxico - portal de datos abiertos de la cdmx,” 2022, accessed:

- 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/grado-de-marginalidad-y-violencia-urbana-por-colonia-en-la-ciudad-de-mexico>
- [105] S. de Salud de la Ciudad de México, “Hospitales y centros de salud - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/hospitales-y-centros-de-salud>
- [106] I. N. de Estadística y Geografía (INEGI), “Límite de alcaldías (áreas geoestadísticas municipales) - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/alcaldias>
- [107] S. de Desarrollo Económico (SEDECO), “Mercados públicos - portal de datos abiertos de la cdmx,” 2022, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/mercados-publicos>
- [108] S. de Transporte Colectivo y Secretaría de Movilidad (SEMOVI), “Ubicación de líneas y estaciones del sistema de transporte colectivo metro - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/lneas-y-estaciones-del-metro>
- [109] S. de Gestión Integral de Riesgos y Protección Civil, “Sismos registrados en la ciudad de méxico - portal de datos abiertos de la cdmx,” 2023, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/sismos-registrados-cdmx>
- [110] S. de Desarrollo Económico (SEDECO), “Tianguis de la ciudad de méxico - portal de datos abiertos de la cdmx,” 2024, accessed: 2024-05-18. [Online]. Available: <https://datos.cdmx.gob.mx/dataset/tianguis-de-la-ciudad-de-mexico>
- [111] I. Amazon Web Services, “Aws educate - cloud services for education,” 2024, accessed: 2024-05-18. [Online]. Available: <https://aws.amazon.com/education/awseducate/>
- [112] I. Namecheap, “Namecheap official website,” 2024, accessed: 2024-05-18. [Online]. Available: <https://www.namecheap.com/>
- [113] I. Docker, *BuildKit Documentation*, 2024, accessed: 2024-05-18. [Online]. Available: <https://docs.docker.com/build/buildkit/>
- [114] ——, *Multi-platform images Documentation*, Docker, Inc., <https://docker.com>, 2024, accessed: 2024-05-18. [Online]. Available: <https://docs.docker.com/build/building/multi-platform/>

- [115] I. Amazon Web Services, *¿Qué es Amazon CloudFront? - Amazon CloudFront*, 2024, accessed: 2024-05-18. [Online]. Available: https://docs.aws.amazon.com/es_es/AmazonCloudFront/latest/DeveloperGuide/Introduction.html
- [116] A. T. P. Ltd., “Exchangerate-api - free & pro currency converter api,” 2024, accessed: 2024-05-18. [Online]. Available: <https://www.exchangerate-api.com/>
- [117] G. Developers, “About pagespeed insights,” 2024, accessed: 2024-05-18. [Online]. Available: <https://developers.google.com/speed/docs/insights/v5/about>
- [118] ———, “Learn analytics,” 2024, accessed: 2024-05-18. [Online]. Available: <https://developers.google.com/analytics/learn>
- [119] J. M. Salas Tafoya, “El modelo de valuación inmobiliaria en México,” *Revista Iberoamericana para la Investigación y el Desarrollo Educativo: RIDE*, vol. 5, no. 10, pp. 31–54, 2015. [Online]. Available: <https://dialnet.unirioja.es/descarga/articulo/5343118.pdf>