

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE



**“SOFTWARE COMO SERVICIO PARA LA GESTIÓN
INTEGRAL DE RESIDUOS ORGÁNICOS EN
COMUNIDADES”**

FRANCISCO ALBERTO SALAZAR HERRERA

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA**

Profesor Guía: Cecilia Reyes Covarrubias
Correferente: Sebastián Felipe Torres Garfe

ENERO - 2020

DEDICATORIA

A todas aquellas personas
que estuvieron conmigo
en todo momento,
dando su brazo y aliento
para culminar este proceso.

AGRADECIMIENTOS

Agradecimientos en primer lugar para mi madre quien en todo momento está cuestionando el mejor camino que podríamos tomar para poder ser mejores en esta vida, esmerándose en que las oportunidades se conviertan en realidad pese a las condiciones adversas. Además, por mostrar el camino a seguir, el cual ha sido una guía constante en esta vida.

A mis hermanas, quienes me apoyan en mis decisiones y creen en el esfuerzo.

A mi hermano, quien con su compañía me ha enseñado bastante.

A mis compañeros de la universidad, quienes también viven una renovación y cuestionamiento constante sobre el sistema y la vida, obligándolos a ser mejores día a día.

A mi excompañera Ximena, quien con su amor y su compañía motivó las ganas de comenzar y terminar este proceso.

A mis compañeros de la vida y trabajo, Cristopher y Alonso, quienes con sus ganas de ser mejores profesionales me motivan a estar en un aprendizaje constante y sacar lo mejor de mí.

A los que me acompañan en otros aspectos de la vida, enseñándome cosas imprescindibles más allá de lo que se puede aprender en la universidad.

A mi profesora Cecilia Reyes, quien me acompañó durante el desarrollo de este proyecto y proceso en Memorias Multidisciplinarias, además de motivar el avance y término de este proyecto.

A Sebastián Torres, por su buena disposición a ayudarme desde el primer día en que le solicité, quien ha sido clave en la parte del diseño y desarrollo de esta solución gracias a su buen nivel técnico y humano.

A todos los que me han apoyado, y quienes comparten conmigo el hecho de que para lograr algo debemos esforzarnos y ser constantes hasta conseguirlo, sin duda el aprendizaje de esta y muchas otras experiencias.

¡Muchas gracias!

RESUMEN

La gestión de residuos en el mundo ha sido un problema enorme desde el momento en que empezamos como humanos a considerar el concepto de “basura”. En Chile no ha sido un problema aislado, pero lo que sí podemos decir es que es un país en donde el manejo de residuos está recién comenzando. Es por lo anterior y por el cambio climático en el que nos encontramos viviendo, que se hace imprescindible que surjan proyectos en este ámbito.

En esta memoria se propone un sistema integral de gestión de residuos orgánicos, donde mediante el uso de tecnologías de la información, se busca crear un sistema inteligente para gestionar estos residuos, con miras en su reutilización, transformándolos en otras fuentes de energía tales como *Compost* o *Humus*.

El diseño del software que se presenta está dentro del marco del programa de Memorias Multidisciplinarias de la USM, por lo tanto, es un proyecto desarrollado por estudiantes de diversas carreras de ingeniería. El módulo informático que se describe en esta memoria es el diseño de una *API REST*, la cual se comunicará con dispositivos de *Internet of Things* para la captura de datos.

Palabras Clave—*Gestión de residuos, Residuos orgánicos, Gestión inteligente de residuos, Internet of Things, API-REST.*

ABSTRACT

Waste management in the world It has been a big problem since we started to adopt the concept of "Waste".

In Chile It has not been an isolated problem, but what we can say is that is a country where Waste management is just beginning. It's for the above and for Climate change in which we are living, that it's essential that projects in this area arise.

In this work an integral system of organic waste management is proposed, where with information technologies, it's sought to create an intelligent system that manage this residual, with a view to the reuse, by transforming it into other energy sources such as Compost or Humus.

Software design that it's presented is within the Multidisciplinary Memories USM program, thus, it's a project developed by students of diverse careers of the engineering area.

In the informatic solution, REST API design is proposed, which Will communicate with internet of things devices for data capture.

Keywords— *Waste management, Organic waste, Smart waste management, Internet of Things, API-REST.*

GLOSARIO

ACL: Access Control List

API: Application Programming Interface

CONAMA: Comisión Nacional de Medio Ambiente

EM: Effective Microorganisms

FORM: Fracción Orgánica de Residuos Municipales

GPS: Global Positioning System

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Security

IoT: Internet of Things

I+D: Innovación + Desarrollo

JSON: JavaScript Object Notation

PEAD: Polietileno de alta densidad

PEBD: Polietileno de baja densidad

PET: Polyethylene Terephthalate

PP: Polipropileno

PS: Poliestireno

RBAC: Role-Based Access Control

REST: Representational State Transfer

RFC: Request for Comment

RSM: Residuos Sólidos Municipales

SaaS: Software as a Service

SI: Sistema de información

SQL: Structured Query Language

TCP: Transmission Control Protocol

TIC: Tecnologías de la Información y Comunicación

UID: Unique Identifier

UTM: Unidad Tributaria Mensual

URI: Uniform Resource Identifier

UTM: Unidad Tributaria Mensual

UTFSM: Universidad Técnica Federico Santa María

INDICE DE CONTENIDOS

RESUMEN	4
ABSTRACT	4
ÍNDICE DE FIGURAS.....	11
INDICE DE TABLAS.....	14
INTRODUCCIÓN	15
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA.....	16
1.1. MOTIVACIÓN	16
1.2. CONTEXTO	16
1.2.1. LA COMUNIDAD ECO URBE	16
1.2.2. ORGANIGRAMA DE LA COMUNIDAD.....	17
1.2.3. ORGANIGRAMA DE LA COMUNIDAD.....	18
1.2.4. GESTIÓN DE RESIDUOS EN CHILE: ANTECEDENTES EN TORNO A LA GESTIÓN MUNICIPAL	18
1.2.5. MODELO DE GESTIÓN DE RESIDUOS EN LA COMUNIDAD ECO URBE	19
1.2.6. RENDIMIENTO DEL SISTEMA ACTUAL DE GESTIÓN DE RESIDUOS EN LA COMUNIDAD ECO URBE	23
1.3. PROBLEMAS Y OPORTUNIDADES.....	25
1.4. PLANTEAMIENTO DE SOLUCIÓN PARA MEMORIA MULTIDISCIPLINARIA	27
1.5. OBJETIVOS DE LA SOLUCIÓN	27
1.5.1. OBJETIVO GENERAL MEMORIA	27
1.5.2. OBJETIVOS ESPECÍFICOS	27
1.6. ALCANCE DE LA SOLUCIÓN	28
CAPÍTULO 2: MARCO CONCEPTUAL	29
2.1. ESTADO DEL ARTE EN EL MUNDO	29
2.1.1. BIG BELLY	29
2.1.2. ENEVO.....	29
2.1.3. ECUBE LABS	29
2.1.4. COMPOLOGY	30

2.1.5. SENSONEO	30
2.1.6. CONCLUSIONES ESTADO DEL ARTE INTERNACIONAL.....	31
2.2. ESTADO DEL ARTE EN CHILE	32
2.2.1. URBANATIKA.....	32
2.2.2. ECOVEN	33
2.2.3. RECICLAPP	33
2.2.4. BICICLA.....	34
2.2.5. KARUBAG.....	34
2.2.6. COMPOST URBANO	35
2.2.7. SR COMPOST	35
2.2.8. CONCLUSIONES ESTADO DEL ARTE EN CHILE	35
2.3. MARCO CONCEPTUAL.....	37
2.3.1. API REST.....	37
2.3.2. SOFTWARE COMO SERVICIO (SaaS).....	38
2.3.3. IOT, SENSORES Y BIG DATA	39
2.3.4. BASES DE DATOS NO RELACIONALES (NOSQL)	39
2.3.5. ARQUITECTURA DE MICROSERVICIOS	39
2.3.6. APLICACIONES MULTIUSUARIOS Y ACCESS CONTROL LIST	40
2.3.7. BACKEND Y FRONTEND.....	40
2.3.8. TECNOLOGÍAS DE DESARROLLO PARA EL BACKEND.....	41
2.3.8.6. NGINX: REVERSE PROXY SERVER	44
2.3.8.7. POSTMAN	44
CAPÍTULO 3: PROPUESTA DE SOLUCION	45
3.1. LA SOLUCIÓN	45
3.1.1. DESCRIPCIÓN GENERAL Y METODOLOGÍA DE DESARROLLO.....	45
3.2. ARQUITECTURA DE LA SOLUCIÓN	46
3.3. USUARIOS DE LA SOLUCIÓN	47

3.4. DIAGRAMA DE DOMINIO	47
3.5. HISTORIAS DE USUARIO E ITERACIONES	49
3.6. ITERACIÓN 1: HISTORIAS DE USUARIO	50
3.6.1. ITERACIÓN 1: MODELO DE DATOS	51
3.6.2. ITERACIÓN 1: REQUERIMIENTOS FUNCIONALES.....	52
3.6.3. ITERACIÓN 1: ENDPOINTS	53
3.7. ITERACIÓN 2: HISTORIAS DE USUARIO	54
3.7.1. ITERACIÓN 2: MODELO DE DATOS	54
3.7.2. ITERACIÓN 2: REQUERIMIENTOS FUNCIONALES.....	55
3.7.3. ITERACIÓN 2: ENDPOINTS	55
3.8. ITERACIÓN 3: HISTORIAS DE USUARIO	57
3.8.1. ITERACIÓN 3: MODELO DE DATOS	58
3.8.2. ITERACIÓN 3: REQUERIMIENTOS FUNCIONALES.....	59
3.8.3. ITERACIÓN 3: ENDPOINTS	59
3.9. ITERACIÓN 4: HISTORIAS DE USUARIO	60
3.9.1. ITERACIÓN 4: MODELO DE DATOS	61
3.9.2. ITERACIÓN 4: REQUERIMIENTOS FUNCIONALES.....	61
3.9.3. ITERACIÓN 4: ENDPOINTS	62
CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	63
4.1. MÉTODO DE VALIDACIÓN	63
4.2. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 1	63
4.2.1. HISTORIA DE USUARIO 1	63
4.2.2. HISTORIA DE USUARIO 2	64
4.2.3. HISTORIA DE USUARIO 3	65
4.2.5. HISTORIA DE USUARIO 5	68
4.2.6. HISTORIA DE USUARIO 6	69
4.3. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 2	70

4.3.1. HISTORIA DE USUARIO 1	70
4.3.2. HISTORIA DE USUARIO 2	71
4.3.3. HISTORIA DE USUARIO 3	72
4.3.4. HISTORIA DE USUARIO 4	74
4.4. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 3	76
4.4.1. HISTORIA DE USUARIO 1	76
4.4.2. HISTORIA DE USUARIO 2	77
4.4.3. HISTORIA DE USUARIO 3	78
4.5. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 4	79
4.5.1. HISTORIA DE USUARIO 1	79
4.6. VALIDACIÓN DE SEGURIDAD BASADA EN ROLES.....	83
CAPÍTULO 5: CONCLUSIONES	85
5.1. CONCLUSIONES GENERALES.....	85
5.2. CONCLUSIONES ESPECÍFICAS.....	85
5.3. RECOMENDACIONES	85
5.4. TRABAJO FUTURO.....	86
5.5. PALABRAS FINALES DEL AUTOR.....	88
BIBLIOGRAFÍA	89

ÍNDICE DE FIGURAS

Figura 1. Estructura funcional de la comunidad Eco Urbe que permite acogerse a la Ley de Copropiedad Inmobiliaria. En amarillo están las entidades externas a la comunidad. Fuente: Elaboración propia.	18
Figura 2. Lista con la caracterización de los materiales que contempla el modelo de gestión heredado desde la planificación realizara por Fundamenta. Fuente: Elaboración propia basada en la documentación entregada por Martín Garrido, jefe de Sustentabilidad en Inmobiliaria Fundamenta.	19
Figura 3. Matriz que relaciona los objetivos del sistema de gestión de residuos propuesto por Fundamenta y sus respectivas medidas. Fuente: Elaboración propia.	20
Figura 4. Ciclo de la gestión de residuos elaborado por Fundamenta. Fuente: Elaboración propia basada en la documentación entregada por Martín Garrido, jefe de Sustentabilidad de Inmobiliaria Fundamenta.	21
Figura 5. Diagrama de contexto que ilustra a los actores involucrados en el proceso de gestión de residuos en Eco Urbe y sus interacciones.	23
Figura 6. Distribución de la cantidad de residuos reciclados en la comunidad Eco Urbe durante el período enero 2017 – marzo 2018. La línea de tendencia indica claramente el aumento sostenido sobre las cantidades recicladas.	24
Figura 7. Cantidad de residuos reciclados por tipo de material período enero 2017 – marzo 2018. Se debe destacar que materiales como los aceites de cocina, artículos electrónicos y libros poseen nulas tasas de reciclaje.	25
Figura 8. Software as a Service	38
Figura 9. Arquitectura de Microservicios orquestada por Docker Compose.	40
Figura 10. Containerized applications vs VMs. Fuente:.....	43
Figura 11. JSON Web token flujo.....	44
Figura 12. Ciclo de vida de un desarrollo con metodología Agile. Fuente: https://www.cognodata.com/blog/principios-metodologia-agile-desarrollo-proyectos/	45
Figura 13. Diagrama de Arquitectura de la solución. Fuente: Elaboración propia.....	46
Figura 14. Diagrama de dominio de la solución. Fuente: Elaboración propia.....	48
Figura 15. Modelo de datos para la iteración 1. Fuente: Elaboración propia.	51
Figura 16. Modelo de datos para la iteración 2. Fuente: Elaboración propia.	54
Figura 17. Primera parte del flujo de asignación de activos. Fuente: Elaboración propia.	56
Figura 18. Parte dos del flujo de asignación de activos. Fuente: Elaboración propia.	57
Figura 19. Modelo de datos de la iteración 3. Fuente: Elaboración propia.	58

Figura 20. Modelo de datos para la iteración 4. Fuente: Elaboración propia.	61
Figura 21. Vista principal de Postman. Fuente: Elaboración propia.....	63
Figura 22. Test asociado a la U.S. 1 de la Iteración 1. Fuente: Elaboración propia.	64
Figura 23. Respuesta para el test de la U.S.1. Fuente: Elaboración Propia.	64
Figura 24. Respuesta para el test de la U.S.2. Fuente: Elaboración propia.	65
Figura 25. Código para almacenar el residentId y residentToken al momento de hacer el login. Fuente: Elaboración propia.	66
Figura 26. Respuesta de la API para el login de Resident. Fuente: Elaboración propia.	66
Figura 27. Variables del entorno Conciencity dev. Fuente: Elaboración propia.....	66
Figura 28. Respuesta para la historia de usuario 3.....	67
Figura 29. Respuesta para PATCH /Residents/{id}. Fuente: Elaboración propia.	68
Figura 30. Respuesta para el método Resident Custom Delete. Fuente: Elaboración propia.....	69
Figura 31. Respuesta para el método DELETE /Recyclers/{id}/CustomDelete. Fuente: Elaboración propia....	69
Figura 32. Respuesta para el método POST /Buckets. Fuente: Elaboración propia.	70
Figura 33. Respuesta del test POST /Buckets. Fuente: Elaboración propia.....	71
Figura 34. Respuesta para el método POST /Scales. Fuente: Elaboración propia.....	71
Figura 35. Respuesta para el test POST /Scales. Fuente: Elaboración propia.	72
Figura 36. Respuesta para el método POST /Residents/{id}/Assignments. Fuente: Elaboración propia.	73
Figura 37. Test para el endpoint POST /Residents/{id}/Assignments. Fuente: Elaboración propia.	73
Figura 38. Asignación de un Bucket para el Residente por parte del Administrador. Fuente: Elaboración propia.	74
Figura 39. Respuesta para el método POST /Recyclers/{id}/Assignments. Fuente: Elaboración propia.	75
Figura 40. Respuesta para el test POST /Recyclers/{id}/Assignments. Fuente: Elaboración propia.	75
Figura 41. Respuesta para PUT /Recyclers/{id}/Assignments/{fk}. Fuente: Elaboración propia.....	76
Figura 42. Validación del método POST /Residents/{id}/Data. Fuente: Elaboración propia.....	77
Figura 43. Respuesta para GET /Recyclers/{id}/Data. Fuente: Elaboración propia.....	78
Figura 44. Respuesta para el endpoint GET /Residents/{id}/RecolectionData. Fuente: Elaboración propia. ..	79
Figura 45. Tests y respuesta para el endpoint GET /Data/totalWaste. Fuente: Elaboración propia.....	80

Figura 46. Respuesta para GET /Residents. Fuente: Elaboración propia.	80
Figura 47. Respuesta para GET /Residents/count. Fuente: Elaboración propia.....	81
Figura 48. Respuesta para GET /Recyclers/count. Fuente: Elaboración propia.	81
Figura 49. Respuesta para GET /Residents/{id}/RecolectionData. Fuente: Elaboración propia.	82
Figura 50. Respuesta para la petición GET /Recyclers/{id}/Data. Fuente: Elaboración propia.	83
Figura 51. Validación de la autenticación basada en Roles. Fuente: Elaboración propia.	84
Figura 52. Arquitectura multi Tenant. Fuente: https://adrianalonso.es/arquitectura-del-software/enfoques-de-arquitecturas-multitenant-para-aplicaciones-saas/	87

INDICE DE TABLAS

Tabla 1. Comparación entre las soluciones internacionales de gestión inteligente de residuos.	32
Tabla 2. Comparación emprendimientos gestores de residuos orgánicos en Chile.	37
Tabla 3. Usuarios de la solución. Fuente: Elaboración propia.	47
Tabla 4. Historias de usuario e iteraciones. Fuente: Elaboración propia.	49
Tabla 5. Endpoints y permisos para la iteración 1. Fuente: Elaboración propia.	53
Tabla 6. Endpoints y permisos para la iteración 2. Fuente: Elaboración propia.	56
Tabla 7. Endpoints y permisos para la iteración 3. Fuente: Elaboración propia.	59
Tabla 8. Endpoints y permisos para la iteración 4. Fuente: Elaboración propia.	62

INTRODUCCIÓN

La gestión de residuos en Chile es un gran tema al igual que para el resto de los países del mundo, siendo lamentablemente este país uno de los que aún no emplea sistemas de información modernos para abordar la generación y captura de datos de interés para mejorar la gestión en el área. Ya estando ad portas del año 2020, un año en que las tecnologías de la información han llegado a un nivel importante, y donde se pueden crear sistemas de una manera más rápida y robusta que en tiempos anteriores, es que se hace imprescindible el estudio y diseño de sistemas informáticos que apoyen en la solución de este tipo de problemas.

La presente memoria está orientada al diseño de un sistema integral de gestión de residuos orgánicos, que corresponden al 50% del total de residuos generados en el hogar, y que pueden tratarse y *convertirse* en otras fuentes de energía como el *Compost* y *Humus*.

Este proyecto se hizo dentro del marco del programa Memorias Multidisciplinarias de la Universidad Técnica Federico Santa María en el año 2018, donde en él participaron 4 integrantes: 2 del área informática (Christopher y Francisco), 1 del área telemática (Giercko) y 1 del área ambiental (Estefanía).

La parte abordada en esta memoria es el diseño y desarrollo del software que sustenta todas las operaciones requeridas por los distintos *tipos de usuario*, siendo esta última una característica importante, junto con las características de seguridad que lo conviertan en un sistema robusto. Cabe mencionar que al ser este un proyecto multidisciplinario, se complementa con otros componentes que corresponden a la interfaz gráfica (creada en memoria de Christopher), la factibilidad medioambiental (propuesta en tema de memoria de Estefanía) y a la componente de *IoT* (creada en memoria de Giercko).

En el capítulo 1, se entrega un estudio en profundidad sobre una comunidad habitacional a modo de evidenciar cómo se están gestionando los residuos actualmente en condominios o comunidades. Además, en este capítulo se estudia la gestión de residuos en Chile y el mundo, lo cual entrega antecedentes suficientes para ver por qué es importante estudiar y proponer soluciones en este rubro.

En el capítulo 2, se explican brevemente todas las tecnologías y conceptos utilizados en esta memoria, esenciales para comprender bien la propuesta de solución desarrollada en el capítulo 3, además, se hace un estudio del Estado del Arte, analizando y comparando todas las soluciones afines encontradas a la fecha tanto para Chile como para el mundo.

En el capítulo 3, se presenta la metodología Agile utilizada y se detalla el diseño y desarrollo del software, que corresponde a una *API REST* con capacidad de interactuar con distintos tipos de usuario o roles, y que además provee una capa de seguridad para cada uno de los roles mencionados.

Finalmente, en el capítulo 4, se valida la solución a un nivel técnico probando todas las funcionalidades propuestas en el capítulo 3, para terminar con las conclusiones presentadas en el capítulo 5.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1. MOTIVACIÓN

La sociedad a la que pertenecemos ha tenido que lidiar con un progresivo aumento en la cantidad de materiales y productos “desechables” a lo largo del tiempo, esto de la mano con el aumento en la producción de las empresas que en gran parte generan productos hechos de este tipo de materiales, entendiéndose como desechable algo que se decreta como “inservible” o no reutilizable. Lo anterior genera una concepción negativa sobre un concepto que denominamos “basura”, donde la mayor parte de los materiales que se crean van a parar a los rellenos sanitarios y/o basurales del país.

El crecimiento sostenido de los materiales desechados en las comunidades a lo largo del país está tomando cada vez más relevancia. Lo anterior se suma al crecimiento continuo de la población y a la concentración de ésta en las ciudades principales, las que tienen que lidiar, a través de sus distintos municipios, con el “problema de la basura”, problema que surge debido a la falta de cultura en cuanto a los residuos y su gestión en general. Lo anterior hace inminente la preocupación por parte de públicos y privados por hacer algo al respecto.

El Estado de Chile ya está al tanto de esta problemática y de sus consecuencias, las que van desde repercusiones severas en el medio ambiente, hasta los enormes costos que los municipios tienen que destinar para poder gestionar de alguna forma los residuos.

Los Residuos Sólidos Municipales (*RSM*), son aquellos generados en los hogares, vías públicas, comercios, oficinas, edificios domiciliarios e instituciones, tales como escuelas y universidades, entre otros. Los datos publicados por *CONAMA* [1] plantean que para el año 2009, la generación estimada de *RSM* corresponde a 6,5 millones de toneladas, esto equivale al 38,5% del total de residuos sólidos, donde el 61,5% restante corresponde a la generación de residuos sólidos industriales.

Se estima que en la Región Metropolitana se llegó a 2.807.247 toneladas de *RSM*, siendo ésta la región más generadora de residuos del país. Además, de acuerdo con los índices de generación per cápita de residuos municipales, para el año 2009 en Chile se alcanzó a 1,05 kg/habitante de residuos por día, donde la Fracción Orgánica de Residuos Municipales (*FORM*) corresponde al 53% de los *RSM*. Estos se refieren a restos de cocina, vegetales de tamaño pequeño, césped, hojas y poda, donde su tasa de valorización es menor al 10%.

1.2. CONTEXTO

1.2.1. LA COMUNIDAD ECO URBE

Este proyecto se estudió para ser aplicado en la comunidad Eco Urbe de la inmobiliaria Fundamenta, aunque, si bien el proyecto se enmarca en un tipo de comunidad con una jerarquía determinada, éste será escalable o replicable a otras comunidades con otro tipo de jerarquía tales como condominios.

La convivencia y administración de las comunidades de edificios como Eco Urbe, está regulada por la Ley de Copropiedad Inmobiliaria Nro. 19.537¹. La forma concreta de acogerse a esta ley de copropiedad inmobiliaria implica que los copropietarios deben organizarse en una asamblea, a la cual deben asistir personalmente o representados; formar un comité de administración; elegir a un administrador; dictar un reglamento y abrir

¹ Es un régimen de propiedad inmueble, en el cual coexisten tanto bienes comunes, como aquellos de cada propietario. El dueño de la copropiedad se denomina copropietario.

una cuenta bancaria exclusiva para la copropiedad. Uno de los aspectos más relevantes de esta ley es que faculta a la administración del edificio a realizar cobros ejecutivos y selectivos de gastos comunes, multas y cortes de suministros eléctrico. Eco Urbe posee una estructura funcional (ver figura 1), tal como lo estipula la ley antedicha, lo cual implica la asignación de las siguientes actividades y roles.

1.2.2. ORGANIGRAMA DE LA COMUNIDAD

- **Asamblea de copropietarios:** Corresponde a una instancia de reunión que tienen los copropietarios para tratar cualquier asunto de interés para ellos y adoptar acuerdos. Sólo pueden asistir a esta asamblea los copropietarios o sus representantes.
- **Comité de administración:** Compuesto por copropietarios, sus cónyuges o sus representantes. Debe contar con a lo menos tres miembros, entre los cuales se debe elegir a un presidente que, en caso de ser necesario, reemplazará al administrador.
- **Comité consultivo:** Grupo de expertos encargado de entregar objetividad e imparcialidad a las decisiones tomadas por el comité administrador.
- **Administrador:** Es designado a través de la asamblea de copropietarios, y a falta de tal designación, el presidente del comité de administración actuará como administrador. El administrador podrá mantenerse en su cargo mientras cuente con la confianza de la asamblea de copropietarios, pudiendo ser removido en cualquier momento por acuerdo de la asamblea. Las funciones que realiza son las establecidas en el reglamento de copropiedad, así como aquellas conferidas por la asamblea, por ejemplo; cuidar los bienes comunes, cobrar y recaudar los gastos comunes, representar en juicio a los copropietarios, entre otras.

1.2.3. ORGANIGRAMA DE LA COMUNIDAD

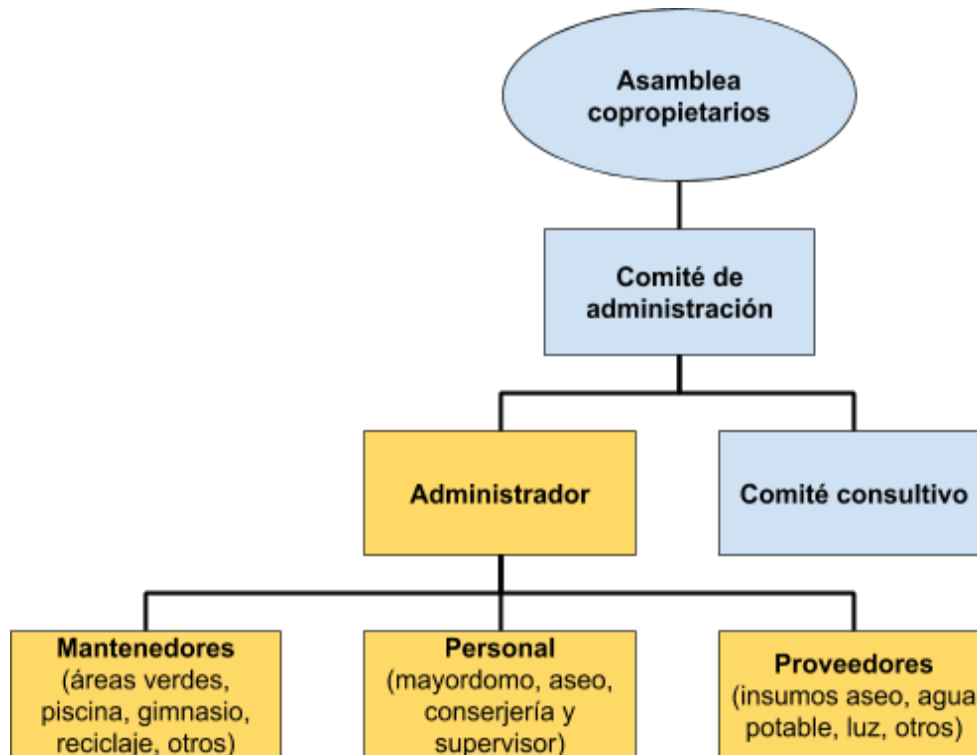


Figura 1. Estructura funcional de la comunidad Eco Urbe que permite acogerse a la Ley de Copropiedad Inmobiliaria. En amarillo están las entidades externas a la comunidad. Fuente: Elaboración propia.

1.2.4. GESTIÓN DE RESIDUOS EN CHILE: ANTECEDENTES EN TORNO A LA GESTIÓN MUNICIPAL

Primero se debe mencionar que la tarea de gestionar los residuos en el país está asignada exclusivamente a los municipios. Son ellos los encargados de velar para que esta labor se lleve a cabo de forma efectiva y de reducir al mínimo los riesgos relacionados con la salud de sus habitantes y del entorno. Sin embargo, esta labor representa cerca del 30% del gasto municipal, lo cual es un número muy elevado considerando lo que se podría potencialmente hacer con ese dinero.

Debido a que los municipios no poseen la infraestructura técnica requerida para gestionar los residuos domiciliarios, éstos llaman a licitación pública este servicio. En sus bases se especifican como mínimo la recolección y el transporte de residuos sólidos domiciliarios, barrido y lavado de veredas y calzadas para toda la comuna². Para que esta labor se pueda sostener económicamente, cada municipio adopta el mecanismo de recaudar fondos a través de un pago por concepto de “derecho de aseo domiciliario”³ que pagan todos los habitantes de la comuna cuyas viviendas tengan un avalúo fiscal mayor a 225 UTM⁴. Sin embargo, debido en gran medida a la falta de educación cívica, los usuarios no tienen conocimiento de este cobro, quedando su deuda impaga y teniendo la falsa creencia de que el servicio de la “basura” es gratuito. La deuda colectiva que

² Esta información fue inferida desde la licitación pública ofertada por el municipio de Ñuñoa (ID: 5482-2-LR18) a través del portal <http://www.mercadopublico.cl>

³ El D.L. 3.063, ley de rentas municipales, faculta a las municipalidades para cobrar directamente el derecho de aseo domiciliario, por el servicio de recolección y disposición de residuos domiciliarios de la comuna. Este derecho se cobra desde el año 1996 a la fecha.

⁴ Para el año 2018 en Ñuñoa, el monto por este servicio es de 1.18 UTM (UTM al 30/06/2017 \$46.740). <http://www.nunua.cl/index.php/2-uncategorised/1093-tarifa-ase-2018.html>

se genera producto de esta desinformación acarrea desórdenes en el flujo de caja de los municipios, que a su vez se ven obligados a contraer deuda para financiar la concesión del servicio, y aunque no estén facultados a realizar esta operación, en la práctica sí lo hacen. Esta deuda es principalmente con la misma empresa proveedora del servicio licitado [2].

1.2.5. MODELO DE GESTIÓN DE RESIDUOS EN LA COMUNIDAD ECO URBE

Desde sus inicios, la comunidad Eco Urbe cuenta con un modelo de gestión de residuos domiciliarios que considera el reciclaje de ciertos productos de consumo masivo (ver figura 2).

CARTONES Y PAPELES	
	Cartón corrugado, papeles impresos, diarios, revistas y cartulinas
	Papel desechable, caja de pizza y huevo, termolaminados
LATAS DE ALUMINIO	
	Latas de bebida, jugo y cerveza limpios
	Tarros de conserva
BOTELLAS PLÁSTICAS PET Y PE	
	Botellas de bebestibles, shampoo, acondicionador, jabón y bolsas
	Envases de aceite, detergente y yogurt. Plásticos desechables, juguetes.
ENVASES TETRA PAK	
	Cajas de leche, vino y jugo limpias
	Cajas con restos líquidos
VIDRIO	
	Botellas de vidrio de jugos, vino y frascos.

Figura 2. Lista con la caracterización de los materiales que contempla el modelo de gestión heredado desde la planificación realizada por Fundamenta. Fuente: Elaboración propia basada en la documentación entregada por Martín Garrido, jefe de Sustentabilidad en Inmobiliaria Fundamenta.

El modelo de gestión que opera hoy en la comunidad fue diseñado de tal manera que responda a través de una serie de medidas a los objetivos fijados por Fundamenta en favor de su estrategia de convertir sus

proyectos inmobiliarios en comunidades sustentables y autosuficientes, sobre todo en relación con el tratamiento de sus residuos. Los objetivos fijados por la Inmobiliaria son los siguientes:

- Comodidad para el usuario
- Capacidad para operar con altas cantidades
- Económico
- Posibilidad de reciclar diversos materiales
- Trazabilidad
- Debe ser limpio y ordenado

Para cada uno de los objetivos definidos anteriormente, se toman como:

- Punto limpio en clóset ecológico
- Acopio diferenciado
- Sala de gestión de residuos
- Empresa especialista
- Material de difusión

Cada medida descrita satisface uno o más de los objetivos que desean alcanzar. Esta relación está descrita en la matriz de la figura 3.

		Medidas				
		Punto limpio en closet ecológico	Acopio diferenciado	Sala de gestión de residuos	Empresa especialista	Material de difusión
Objetivos	Cómodo para el usuario	X				
	Capacidad para operar con altas cantidades	X		X	X	X
	Ecónomico			X	X	
	Posibilidad de reciclar diversos materiales				X	
	Trazable				X	
	Debe ser limpio y ordenado	X	X	X		X

Figura 3. Matriz que relaciona los objetivos del sistema de gestión de residuos propuesto por Fundamenta y sus respectivas medidas. Fuente: Elaboración propia.

Conforme a lo anterior, la implementación de todas estas medidas se traduce en un método de gestión de residuos que se refleja en el siguiente ciclo (ver figura 4):

▪ **Ciclo de gestión de residuos en comunidad Eco Urbe**

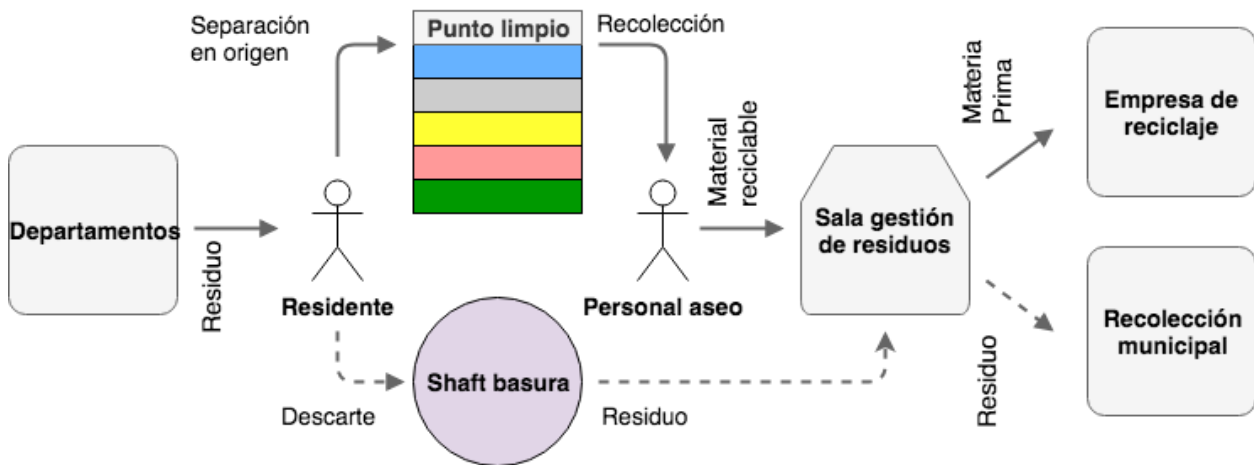


Figura 4. Ciclo de la gestión de residuos elaborado por Fundamenta. Fuente: Elaboración propia basada en la documentación entregada por Martín Garrido, jefe de Sustentabilidad de Inmobiliaria Fundamenta.

Primero, ocurre la generación de diversos residuos en cada departamento por parte de los residentes. Estos residuos pueden estar separados en origen, o bien pueden estar mezclados tanto con material reciclable, como con materiales de otro tipo (materia orgánica, tóxicos, electrónicos, entre otros). Después, cada residente se dirige a disponer sus residuos al lugar que corresponda.

Si los residuos son reciclables:

1. Se deberá disponer el residuo compuesto por material reciclable en el clóset ecológico o punto limpio de acuerdo con su categoría.
2. Personal de aseo debe recurrir a los closets ecológicos ubicados en cada piso, recolectar el material reciclable y ubicarlo en su carro de transporte.
3. Personal de la empresa de reciclaje⁵ se dirige a la sala de gestión de residuos, donde realiza un pretratamiento del material reciclable dejándolo apto para su retiro.
4. Empresa de reciclaje transporta la materia prima desde la sala de gestión hacia su centro de acopio para luego ser comercializado.

De lo contrario, si los materiales no son reciclables o el residente no separa en origen, mezclando todos sus residuos:

1. Se deberá disponer el residuo en un *shaft* ubicado en cada piso.

⁵ Hasta mayo del 2018, la empresa encargada de realizar esta labor es Triciclos. Para más información ver <http://www.triciclos.net/es/>.

2. Los residuos viajan por el *shaft* directo hacia la sala de gestión de residuos.
3. Personal de aseo se encarga de gestionar los contenedores que recepcionan los residuos que provienen del *shaft* con el fin de evitar el desbordamiento de los residuos.
4. Personal de aseo dispone los contenedores con residuos para el retiro municipal.

Durante todo este proceso, interactúan una serie de actores. Estos son detallados a continuación (ver figura 5):

- **Residente:** Persona que habita el departamento, puede ser copropietario o arrendatario. Es el generador de residuos y encargado de realizar el pago del gasto común, la separación en origen de los residuos potencialmente reciclables y de llevar el resto de los residuos al lugar destinado para ellos, el *shaft* de basura. Además, es quien debe adoptar las herramientas de gestión propuestas por la Inmobiliaria.
- **Comité administración:** Entidad encargada de velar por los intereses de la comunidad en torno a los servicios prestados por las demás entidades involucradas en el proceso de gestión de residuos. También propone, evalúa e implementa soluciones en favor de la comunidad.
- **Administrador:** Encargado de proveer insumos y personal de aseo para la operación del proceso de gestión de residuos. También se encarga de gestionar el pago del gasto común realizado por los residentes y destina parte de éste al proceso de gestión de residuos.
- **Fundamenta:** Empresa inmobiliaria encargada de suministrar las herramientas necesarias para llevar a cabo el proceso de gestión de residuos de la comunidad de manera que pueda sostenerse de forma autónoma y sustentable. Fundamenta entrega los edificios con la infraestructura inicial para que se efectúe el reciclaje, esto es; contenedores, closets ecológicos, etc. Una vez que el edificio entra en operación, la inmobiliaria se desliga y todo lo necesario para que continúen correctamente las labores de reciclaje dependerá de la comunidad.
- **Triciclos:** Empresa de reciclaje que realiza el pretratamiento del material reciclable provisto por cada residente y recolectados por el personal de aseo. Triciclos además de recibir un pago por este servicio⁶, debe proporcionar reportes que consideren las cantidades de material reciclado por la comunidad.
- **Recolector municipal:** Servicio de recolección provisto por el municipio, el cual no contempla la separación de residuos.

⁶ Hoy en día, y de acuerdo con una serie de negociaciones, el pago por este servicio asciende a UF 12 (UF al 05/06/2018 \$27.091). F. Salazar, Primera entrevista con Gloria Guerra, presidenta del comité administración de la comunidad Eco Urbe. Santiago, 2018.

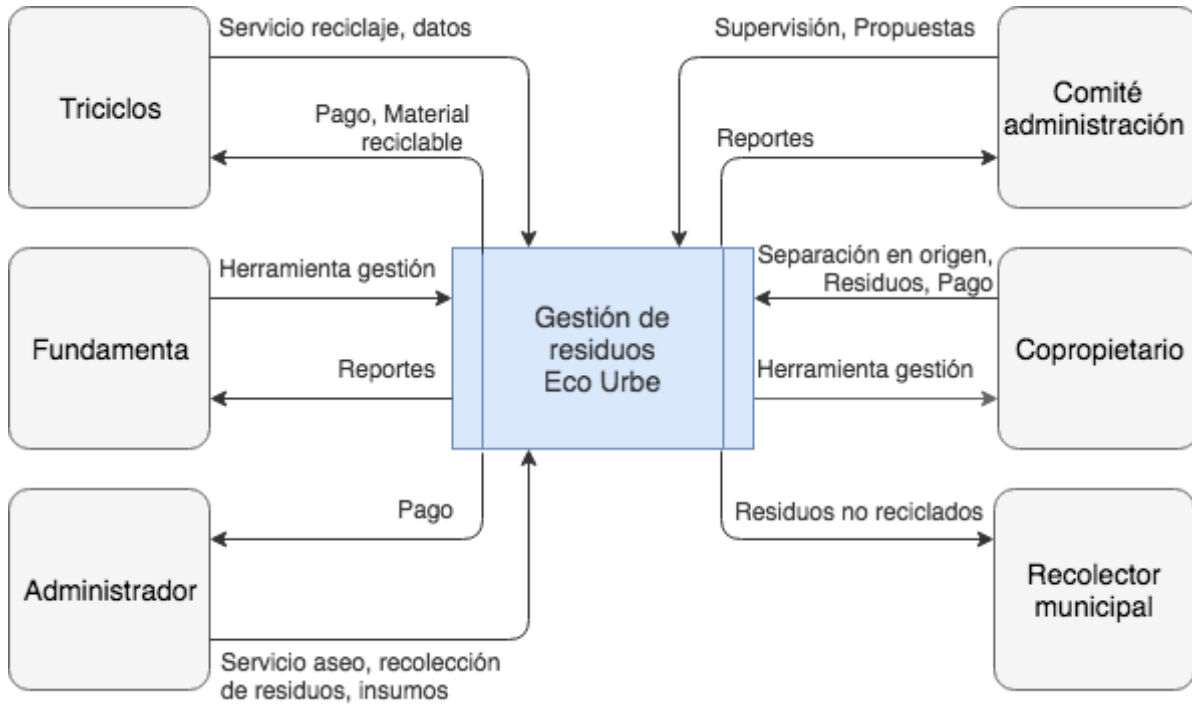


Figura 5. Diagrama de contexto que ilustra a los actores involucrados en el proceso de gestión de residuos en Eco Urbe y sus interacciones.

1.2.6. RENDIMIENTO DEL SISTEMA ACTUAL DE GESTIÓN DE RESIDUOS EN LA COMUNIDAD ECO URBE

Según los datos aportados por Fundamenta⁷, desde enero del 2017 hasta marzo del 2018, se reciclaron 39,7 toneladas de residuos en la comunidad Eco Urbe. La distribución de esta cantidad a lo largo del período mencionado se ilustra la figura 6.

⁷ Documentación acerca de las cantidades recolectadas y recicladas en la comunidad Eco Urbe por la empresa de reciclaje Triciclos durante el período enero 2017 hasta marzo 2018. Entregada por Martín Garrido, Jefe de Sustentabilidad en Inmobiliaria Fundamenta.

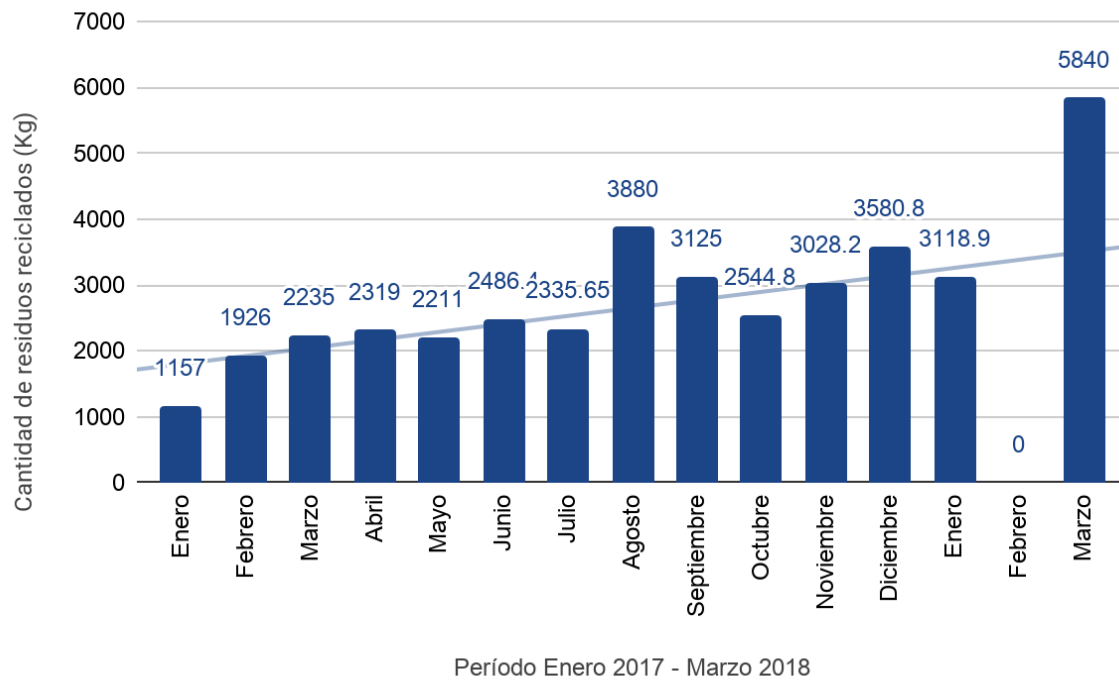


Figura 6. Distribución de la cantidad de residuos reciclados en la comunidad Eco Urbe durante el período enero 2017 – marzo 2018. La línea de tendencia indica claramente el aumento sostenido sobre las cantidades recicladas.

Por otro lado, el detalle de los tipos de material reciclado a lo largo del período, se ilustran en la figura 7, donde claramente se aprecia que el tipo de material que más se recicla en la comunidad corresponden a: vidrio, cartón y plástico *PET* (en inglés Polyethylene Terephthalate). Otros tipos de material como el aluminio, plástico *PEAD*, *PEBD* (en base a Polietileno de alta densidad), envases Tetra Pack, papel blanco, otros papeles y metales, plásticos *PP* (Polipropileno) y *PS* (Poliestireno), diarios, otros plásticos y film, se reciclan en bajas cantidades.

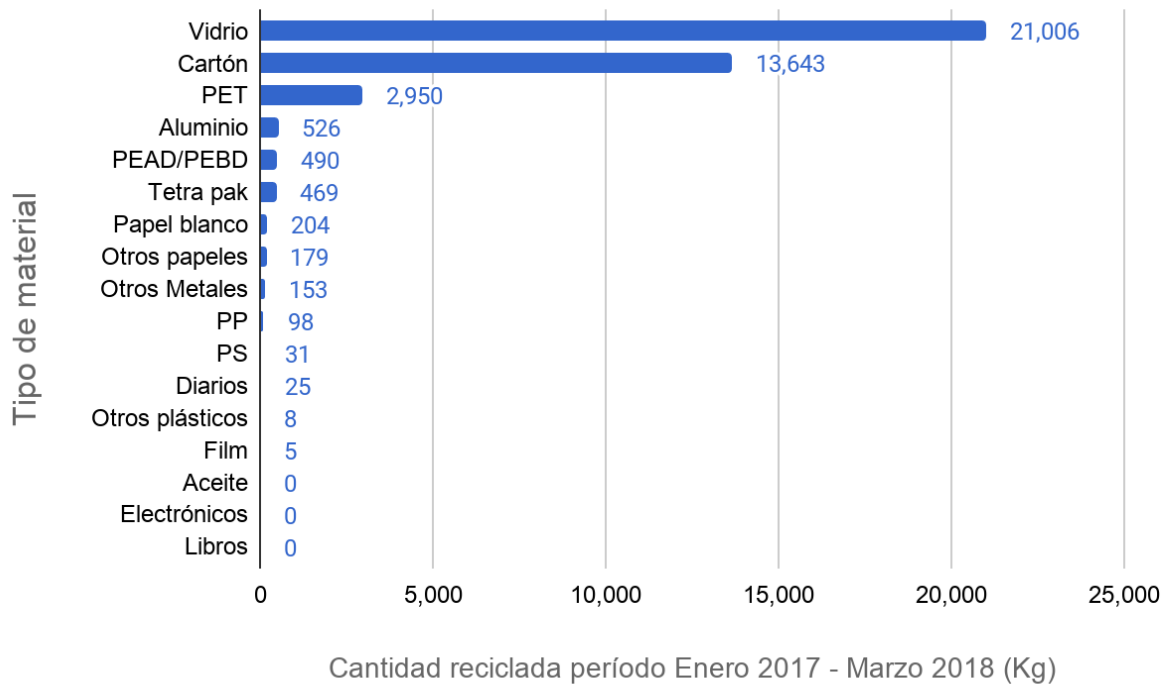


Figura 7. Cantidad de residuos reciclados por tipo de material período enero 2017 – marzo 2018. Se debe destacar que materiales como los aceites de cocina, artículos electrónicos y libros poseen nulas tasas de reciclaje.

1.3. PROBLEMAS Y OPORTUNIDADES

El estudio anterior permite detectar varios dolores que sufren los actores relevantes en este contexto, pasando por los municipios, quienes deben lidiar con la responsabilidad de la gestión de residuos de su comuna, hasta el residente del edificio, quien debe pagar un gasto común por concepto de servicios comunes de la comunidad, de entre los cuales se contempla los ítems de “basura” y “reciclaje”.

Si bien el sistema de gestión de residuos implantado en la comunidad Eco Urbe contempla la interacción de varios actores, sólo se mencionan las problemáticas relativas a los actores internos, tales como los residentes, el comité de administración, el administrador y la inmobiliaria.

Sus problemas actuales son:

- Residentes:
 - Deben pagar un gasto por reciclaje incluido en el gasto común.
 - Empresa de reciclaje utiliza los recursos humanos disponibles para la comunidad como el personal de aseo, quienes deben apoyar las labores de reciclaje sin recibir nada extra a cambio.
- Comité de administración:
 - Sistema de gestión de residuos no contempla las necesidades reales de la comunidad.

- La comunidad no obtiene retribución directa por separar sus residuos en origen y disponer el material reciclable.
- Administrador:
 - Personal de aseo consume tiempo extra en la gestión de residuos debido a la falta de herramientas y recursos.
 - Residentes no hacen un buen uso de las herramientas existentes para el reciclaje y siguen disponiendo material reciclable en el *shaft* de basura.
 - Existe maquinaria ociosa inútil para la gestión de residuos que se podría aprovechar perfectamente para apoyar algunos procesos.
- Fundamenta:
 - Residentes de sus comunidades poseen el paradigma de que la empresa de reciclaje obtiene grandes utilidades por la venta del material reciclable, y, por lo tanto, piensan que no deberían cobrar por la recolección de estos residuos.
 - El espacio disponible para el sistema de gestión de residuos en los edificios es pequeño.
 - Sistema de gestión de residuos no se sostiene de forma autónoma.
 - No se cumplen las metas de ecoeficiencia que ellos mismos proponen.

Lo anterior supone una serie de oportunidades que podrían implementarse en la comunidad, con el fin de satisfacer las necesidades de todos los actores involucrados en el proceso de gestión de residuos.

En primer lugar, se tiene la ventaja de que gran parte de la comunidad ya está llevando a cabo la separación en origen de sus residuos con el objetivo de reciclarlos. Lo anterior implica que es posible permear aún más en sus hábitos y así lograr elevar las tasas de reciclaje en la comunidad. Al elevar la cantidad de material reciclable es posible reconfigurar los costos fijos incurridos por la empresa de reciclaje, de tal modo que esta operación no tenga costo para la comunidad.

Otro punto importante es el apoyo que presta el personal de aseo de la comunidad al proceso de gestión de residuos. Al tener personal disponible para la recolección, es posible darle valor agregado a su trabajo a través de la asignación de labores de preprocesamiento del material reciclable hasta dejarlo apto para su venta como materia prima. Esto significa un aumento en la cadena de valor de los residuos generados por la comunidad, que, a su vez, les permitiría obtener un ingreso de dinero (por la venta del mismo material), con la posibilidad de trasladarlo como bono salarial al personal de aseo.

El sistema de reciclaje que actualmente rige en la comunidad contempla residuos de consumo masivo, tales como los plásticos y el cartón, no obstante, existe una categoría de residuos que es la que más se genera⁸, y no tiene tratamiento alguno en la comunidad: residuos orgánicos.

⁸ Cerca del 50% de los residuos que se generan en cada hogar en Chile corresponden a la categoría de residuo orgánico. Extraído desde la página web del Ministerio del Medio Ambiente. <http://portal.mma.gob.cl/en-barrio-serenense-realizan-talleres-de-reciclaje-de-material-organico/>

Hoy en día existen tecnologías de bajo costo para la comunidad que son capaces de procesar los residuos orgánicos localmente y así obtener una serie de activos, producto del proceso de transformación de estos residuos⁹ en otras fuentes de energía.

1.4. PLANTEAMIENTO DE SOLUCIÓN PARA MEMORIA MULTIDISCIPLINARIA

El objetivo general propuesto en el marco de Memorias Multidisciplinarias UTFSM consiste en desarrollar un prototipo funcional de un sistema de gestión de residuos orgánicos para la comunidad Eco Urbe de la inmobiliaria Fundamenta, este prototipo abarca el desarrollo de componentes electrónicos con sensores *IoT* para la captura de datos de los pesos de los residuos de los residentes como también el censado de las casas que participan en la recolección, de manera de tener data relacionada al proceso de gestión de residuos orgánicos (memoria de Ingeniero Civil Telemático). Además, se propone el desarrollo de una *API REST* que esté disponible como un *SaaS* que permita gerenciar todos los requerimientos funcionales que el sistema requiera con capacidad para tener múltiples tipos de usuario (administrador, residente, reciclador) mientras que todos los datos e información podrán visualizarse mediante otro componente web que es una *dashboard* de visualización de la información (memoria de Ingeniero Civil Informático).

Abarca también el estudio e implementación de una compostera para la comunidad, y el desarrollo de los procesos que involucra la gestión de residuos para que se pueda llevar a cabo de manera efectiva (memoria de Ingeniero Civil Ambiental).

1.5. OBJETIVOS DE LA SOLUCIÓN

1.5.1. OBJETIVO GENERAL MEMORIA

Desarrollar un prototipo funcional de **sistema de gestión** de residuos orgánicos orientado a personas, que permita capturar y almacenar datos de los procesos involucrados, mediante la construcción de un sistema web basado en **microservicios**.

1.5.2. OBJETIVOS ESPECÍFICOS

- Estudiar la situación actual de la gestión de residuos en Chile y particularmente en las comunidades habitacionales para levantar el conjunto de requerimientos que debiera satisfacer el sistema.
- Diseñar y desarrollar una solución informática para múltiples tipos de usuario, segura y escalable, que cumpla con los requisitos propuestos para la gestión de residuos orgánicos de comunidades, y que se implemente en un entorno *cloud* para que quede disponible como un *Software como Servicio (SaaS)*.
- Validar la solución mediante pruebas de historias de usuario que permitan asegurar la calidad del prototipo.

⁹ Existen diversas técnicas disponibles para el tratamiento de residuos orgánicos que entregan como productos; el biogás, humus de lombriz y compost. Estos dos últimos ítems permitirían incluso obtener un tercer activo en la comunidad a través de la producción de fertilizantes y mejoradores de suelo.

1.6. ALCANCE DE LA SOLUCIÓN

Con el desarrollo de un prototipo de gestión de residuos orgánicos se buscan dos cosas principalmente: la primera es evidenciar la ineficiencia y desperdicio de recursos que existen hoy en día en las comunas del país al tratarse de temas como la gestión de residuos y, la segunda, es proponer y validar una solución alternativa o complementaria que se apoye en las tecnologías de vanguardia y que permita gestionar un tipo de residuos que hoy en día no se gestiona adecuadamente.

El sistema permitirá, mediante la captación y almacenamiento de datos, construir una base de información medible para la toma de decisiones.

Es importante mencionar además que la situación actual de datos con respecto a la gestión de residuos en el país es muy deficiente, puesto que en su mayoría los datos que existen en informes y/o fuentes confiables son datos que provienen de fuentes análogas donde hay un factor humano que induce a errores no menores, por lo que la generación de estos mediante fuentes que minimicen el error es un punto clave en sistemas de este estilo.

Este prototipo busca el uso eficiente de los recursos, desde los recursos humanos como los Residentes: quienes dan origen a los residuos que se tratarán, y los voluntarios: quienes recolectarán los residuos casa por casa. Finalmente, tenemos los recursos intangibles que tienen un gran valor para la mejora continua de los procesos de un sistema: información.

El prototipo funcional podrá ser implementado en la comunidad Eco Urbe y/u otras comunidades habitacionales que lo requieran, sin embargo, en esta memoria sólo quedará documentado hasta la fase previa a la implementación.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1. ESTADO DEL ARTE EN EL MUNDO

La industria de gestión de residuos no ha logrado aún una madurez que permita un uso eficiente de los recursos con que cuenta, por lo que en el mundo ya hay varias iniciativas enfocadas a hacer más eficientes los procesos de llenado y recolección de los basureros, mediante componentes *IoT* que permiten mediciones de datos para que se puedan generar, por ejemplo, rutas eficientes de recolección y así ahorrar costos asociados al transporte requerido (combustible) y emitir menos CO₂ al medioambiente.

A continuación, se presentan algunas de las soluciones más emblemáticas, que, aunque no se traten específicamente de gestión de residuos orgánicos, es importante estudiarlas ya que utilizan tecnologías de *IoT* para la captura de datos, y servicios web basados en la nube como *SaaS*.

2.1.1. BIG BELLY

Empresa de USA [3], dicen ser los líderes en la gestión inteligente de basura y reciclaje. Proveen contenedores inteligentes alimentados con energía solar y conectados a un sistema de información en la nube (*SaaS*). El enfoque de su producto se encuentra en la tecnología que provee el contenedor, ya que es capaz de compactar los residuos y así tener una capacidad de hasta 8 veces más que un contenedor tradicional del mismo tamaño.

Ofrecen un software que permite visualizar información interesante para la gestión de los contenedores, como la ubicación, el estado de las baterías y el nivel de llenado de cada uno. Además, proveen analítica para la recolección inteligente de los contenedores llenos y generan estadísticas y métricas de la gestión de residuos.

Entre sus contenedores tienen la opción de reciclar los siguientes tipos de residuos: Basura, latas, papel y residuos orgánicos compostables.

Los contenedores pueden ser de capacidad estándar, es decir, de aproximadamente 190 litros, o de alta capacidad, alrededor de 570 litros.

2.1.2. ENEVO

ENEVO [4] es una empresa ubicada en USA que ofrece un servicio a bajo costo y que genera un impacto medioambiental positivo. El servicio está orientado a gestión de residuos de empresas y consiste en proveer contenedores que poseen un sensor de ultrasonido para medir la capacidad de llenado del contenedor en el tiempo.

Proveen en la solución un sistema de información para poder ver el estado de llenado de los contenedores, así como la generación de una planificación óptima de recolección de los contenedores, al igual que Big Belly.

2.1.3. ECUBE LABS

Empresa de Corea del Sur [5], desarrollaron una solución para la gestión inteligente de residuos, con el fin de mejorar el proceso de recolección de estos mediante la implementación de una red inteligente que permite compactar los residuos a un nivel tal que se logra un almacenaje que permite aprovechar el espacio hasta 5

veces más de lo normal. Los contenedores poseen sensores de llenado, de batería y se alimentan con energía solar.

Mediante una interfaz web, les permiten a sus clientes visualizar rutas eficientes para la recolección de residuos, y de esta forma, ahorrar hasta un 80% de sus costos de recolección.

2.1.4. COMPOLOGY

Compology [6] es un producto que, mediante el uso de cámaras resistentes instaladas en contenedores, y un software basado en la nube (*SaaS*), permiten monitorear nivel de llenado, contenido, ubicación y tener una vista general del servicio. Entre sus ventajas, se menciona que pueden optimizar las rutas de recolección en un 40%.

Usan una cámara angular que captura varias fotos de alta resolución al interior del contenedor, lo que da como resultado datos completos y se puede distinguir el contenido del contenedor. Además, todos los contenedores cuentan con GPS, por lo que se pueden ver su ubicación en el software.

Mediante un acelerómetro en la cámara, miden cuándo la tapa del contenedor es levantada.

El software posee una interfaz de usuario con altos estándares de usabilidad, que permite analizar el comportamiento de los contenedores cada día del año desde cualquier lugar, desde cualquier dispositivo, sin necesidad de descargar nada (ventajas de un *SaaS*).

2.1.5. SENSONEO

Es un proveedor de soluciones de gestión de residuos inteligentes de nivel empresarial que permite a las ciudades y empresas gestionar sus residuos de manera rentable, ser más responsables con el medio ambiente y mejorar el bienestar de las personas [7].

Poseen una plataforma basada en la nube que ayuda a los clientes a configurar, monitorear y planificar las actividades de gestión de residuos todos los días. Los clientes tienen la posibilidad de ver todos los contenedores en un mapa digital vía *Street View* incluyendo capacidad, tipo de residuo, última medición, ubicación *GPS* y la planificación de su recolección.

Sus contenedores poseen sensores configurables que permiten monitorear el nivel de llenado en tiempo real, ver predicciones, planificar rutas de recolección, ver informes de selección y generar informes para los ciudadanos.

El sistema de gestión de residuos está disponible en 10 idiomas y ellos proveen una API mediante la cual dan la posibilidad a los usuarios de integrar estos datos en un sistema de información particular.

Los puntos clave que ofrece su servicio son:

- Monitoreo en tiempo real
- Predicciones para el llenado del contenedor
- Bases de datos con el detalle de los contenedores

- Mapa interactivo del contenedor vía Street View
- Planificación de las rutas de recolección (Smart routes)
- Vista de las planificaciones y rutas ejecutadas
- Bases de datos de reportes ciudadanos
- Alarma de incendio e inclinación del dispositivo

2.1.6. CONCLUSIONES ESTADO DEL ARTE INTERNACIONAL

Existen diversos sistemas y aplicaciones a lo largo del mundo que están ayudando a que la gestión de residuos sea más eficiente, sin embargo, las soluciones presentadas si bien usan componentes tecnológicos como *IoT* y *SaaS*, estos están enfocados en hacer más eficiente el sistema de recolección de residuos y gestión de procesos actual, que consiste básicamente en recolectar los residuos por parte de los municipios, y llevarlos a un lugar de reciclaje o disposición final, no solucionando el problema en su totalidad. Algunos pierden de vista el énfasis en la separación en origen del residuo, y muchos no contemplan la gestión de residuos orgánicos.

A modo resumen podemos ver las comparaciones, pros y contras de las alternativas anteriores mencionadas

Empresa	País	Características principales	Sensores o métodos	SaaS
Big Belly	USA	• Contenedores alimentados por energía solar • Fuerte capacidad para compactar los residuos • Nivel de llenado	• Sensor de ultrasonido • GPS	Sí
Enevo	USA	• Recolección inteligente • Nivel de llenado	• Sensor de ultrasonido • GPS	Sí
Ecube Labs	Corea del sur	• Contenedores con capacidad para compactar y tener hasta 5 veces más capacidad que los tradicionales.	• Sensor de ultrasonido • Nivel de batería • GPS	Sí
Compology	USA	• Uso de imágenes para reconocer el nivel de llenado del contenedor	• Reconocimiento de imágenes • GPS	Sí

		• Uso de imágenes permite saber qué tipos de residuos se encuentran dentro del contenedor		
Sensoneo	Eslovaquia	• Nivel de llenado y predicción • Rutas inteligentes • Alarma de incendio • Generación de reportes	• GPS (Street View) • Sensor de ultrasonido	Sí

Tabla 1. Comparación entre las soluciones internacionales de gestión inteligente de residuos. Fuente: Elaboración propia.

2.2. ESTADO DEL ARTE EN CHILE

Internacionalmente hay diversas empresas que están ofreciendo soluciones de gestión de residuos inteligentes como se muestra en la sección anterior, mediante el uso de sensores y tecnología en sus procesos, sin embargo, en Chile no se encuentran muchos ponentes que usen la tecnología a tal nivel, aunque existen algunos que se aproximan como veremos a continuación.

2.2.1. URBANATIKA

Es un emprendimiento creado por Diego Elorza, estudiante de INACAP. La propuesta de Urbanatika es generar una red de compostadores urbanos y comunitarios mediante un sistema de sensores al que llaman “Internet del Compostaje”, que se encarga de monitorear el estado del compost y entregar información al municipio sobre el comportamiento de los vecinos con respecto al compost, mapas de calor y territorial de reciclaje orgánico y además, le entregan al vecino o usuario información del proceso de su compostaje para que pueda maniobrarlo sin tener conocimiento experto.

Integran sensores a los vecinos que tienen composteras y a los que no, se les entrega una compostera y se les capacita, luego con ayuda del municipio pretenden extender la red del cultivo urbano y colaborativo para generar una gran red de reciclaje orgánico, recuperación del suelo, cultivo y reforestación ciudadana [8].

Los servicios que Urbanatika ofrece según el tipo de cliente o área industrial son:

- **Empresas e industria:** Servicios de *I+D* tecnológico ambiental enfocados en el agro y compostaje para las empresas e industrias generadoras de residuos orgánicos, como casinos, industria de alimentos, ferias libres.
- **Municipios:** Servicio de huertos urbanos automatizados para cultivar en la ciudad y aprender de la agricultura junto a vecinos y el internet del compostaje para hacer gestión de la basura orgánica mediante sensores de *IoT* y compostaje urbano.

- Casas y comunidades: Servicios de composteros con sensores *IoT* conectados a plataforma Urbanatika para recuperar y valorizar la basura orgánica en comunidad y transformar los espacios urbanos con el compost y humus generados.
- Educación: Se brinda una experiencia de aprendizaje en torno a la innovación agrícola, pero para la ciudad. Los jóvenes de colegios conocen sobre *IoT* y compostaje y sus beneficios por medio de la tecnología [9].

2.2.2. ECOVEN

Creado por Mauricio Quezada y Elizabeth Bilbao en Santiago, el emprendimiento busca incentivar el reciclaje a partir de recompensas y premios para sus usuarios. Esto lo hacen a partir de estaciones con contenedores que reciben distintos materiales valorizables. Las máquinas que instalaron son de origen italiano y cada una cuesta cerca de 8.000 euros [10].

Los incentivos hacia sus usuarios consisten en: cargas para la tarjeta Bip o *TNE*, cupones de descuentos para comercios asociados, donación de dinero a fundaciones y premios adicionales a los mejores recicladores.

Lo materiales que reciben sus máquinas son: envases (*PET*, latas, vidrio), celulares, papel, ropa.

En el sitio web se menciona que los contenedores se pueden personalizar añadiéndoles una pantalla, además de que se les puede incorporar una compactadora interna que reduce considerablemente el volumen [11].

Todo lo anterior funciona mediante una *app* de manera sencilla, básicamente el usuario ingresa un residuo y la máquina le entrega un cupón, luego deben ingresar el código del cupón en la aplicación para cargar sus puntos (*ecopuntos*). Finalmente, los usuarios pueden cambiar sus *ecopuntos* por descuentos.

2.2.3. RECICLAPP

ReciclApp es una aplicación móvil que gestiona, optimiza y mejora el proceso de reciclado, permitiendo conectar a todas las personas, instituciones y comercios con recicladores que estén dispuestos a retirar este material para luego comercializarlo libremente [12].

Entre sus características se encuentran:

- Uso de vehículos con cero emisiones contaminantes por parte de los recicladores
- Sistema de gestión de recolección domiciliaria planificada y periódica gratuita o a un costo accesible
- Incentivos a los usuarios a través de entrega de premios
- Apoyo a los recicladores base vinculándolos con las empresas recicladoras

La aplicación permite declarar retiros, para esto es necesario indicar las cantidades y tipo de residuos, para que luego un reciclador vaya a efectuar el retiro del residuo.

2.2.4. BICICLA

Aplicación creada por Felipe Lüer y Francisco Lira en el sur de Chile, consiste en generar una red de reciclaje colaborativo mediante la conexión entre ciudadanos dispuestos a separar sus residuos reciclables, y ciudadanos que ya irán a dejar estos residuos en bicicleta (de ahí el nombre) a los puntos limpios.

Basados en un modelo de tendencia mundial colaborativa tipo Uber, Airbnb, entre otras.

A la fecha, han logrado reciclar más de 16.343 KG de residuos de todo tipo, cuentan con más de 437 colaboradores y están presentes en 9 regiones del territorio nacional.

Entre los residuos que permiten reciclar se encuentran:

- Botellas plásticas sin etiquetas
- Envases Tetra Pak
- Latas de aluminio limpias y aplastadas
- Diarios y revistas

Además, dan la opción de retirar residuos voluminosos como electrodomésticos en desuso que puedan ser reutilizados.

Incentivan a los usuarios a reciclar ofreciendo premios como descuentos en aplicaciones como Uber o Spotify y últimamente adoptaron la retribución de Bitcoins [13].

Su sistema de información consiste en una aplicación móvil.

2.2.5. KARUBAG

Karubag es una empresa dedicada a la gestión y manejo de residuos orgánicos de cocina y jardín. Consiste en una bolsa de papel biodegradable, reciclable y compostable, donde los usuarios pueden poner los desechos de jardín como pasto, hojas y ramas. Luego, Karubag hace el retiro de la bolsa con los residuos orgánicos reciclados por los hogares y elabora abono en base ellos. Proveen planes mensuales, donde mes a mes le van dando *hummus* al usuario si es que este lo quiere.

En los servicios que ofrecen, hacen la separación entre recolección de residuos orgánicos de jardín y recolección de residuos orgánicos de cocina, aunque también tienen un servicio que junta ambos [14].

Sus planes van desde los \$9.990 hasta los \$15.990 mensuales [15] y consisten en la entrega de una cantidad de bolsas Karubag o baldes (Karubucket) y una cantidad de retiros mensuales.

Sus servicios están orientados a los hogares, a las oficinas y a empresas.

No cuentan con un sistema de información que respalde la gestión de los procesos ni el almacenamiento de datos de estos.

2.2.6. COMPOST URBANO

Es un proyecto que nace para incentivar a la comunidad a separar y reciclar los residuos orgánicos que producimos en la cocina, creado por Tere Torrealba, Ecóloga paisajista de la Universidad Central de Chile en el 2016.

Los planes que ofrecen contemplan la entrega de una caja con tapa donde el usuario puede separar sus residuos y guardarlos hasta que Compost Urbano haga el retiro, momento en el que le pasan otra caja limpia y se llevan la caja con residuos. Los retiros los realizan una vez por semana, llevando los residuos a una planta de compostaje y vermicompostaje. Al igual que Karubag, cada tres meses le entregan al usuario Compost, si este así lo quiere.

No hay información de alguna aplicación que apoyen parte de los procesos de la gestión de su operación, por lo que se infiere que este sistema es totalmente análogo.

2.2.7. SR COMPOST

Creado por Álvaro Pineda, arquitecto, quien en el año 2018 crea un servicio de gestión de residuos orgánicos para particulares, el cual, mediante una suscripción mensual, le entrega un balde con una bolsa compostable a sus usuarios, donde ellos deberán ir depositando sus residuos orgánicos y luego, una vez a la semana, el Sr. Compost y los trabajadores del sistema irán a buscar los residuos a los distintos clientes. Al igual que los servicios anteriormente analizados, este también provee la entrega de una bolsa de hummus generado por el compostaje.

El retiro de los residuos lo realizan en bicicleta para reducir aún más la huella de carbono, y además para dar trabajo a gente que quiera y pueda recolectar.

En el año 2019 se une con Urbanatika y, basados en el modelo de Dropoff¹⁰ de ciudades como New York y Washington, desde junio de ese año entrega la opción gratuita de dejar los residuos orgánicos en un punto público de la ciudad, para luego ser compostados con tecnología EM. También están unidos con Reciclo (acción social franco-chilena) [16].

2.2.8. CONCLUSIONES ESTADO DEL ARTE EN CHILE

Existen diversas iniciativas para gestionar residuos tanto orgánicos como de otro tipo, sin embargo, muchas de ellas funcionan, en gran medida, de manera análoga. Lo anterior se debe a que es común que proyectos de este tipo sean liderados por personas no tan ligadas al ámbito tecnológico, pero que sí tienen fuerte interés en el medioambiente y en los procesos, por ejemplo, de compostaje y vermicompostaje.

El no contar con sistemas de información (*SI*) automatizados acarrea algunos problemas y pérdidas de oportunidades en especial en la gestión de los procesos involucrados, pues como los datos siguen siendo análogos, no es tan sencillo establecer mejoras en el rendimiento del sistema. Además, se pierde la posibilidad de usar características importantes y potentes como la gestión de rutas inteligentes de recolección o el manejo de una manera sencilla y ordenada la escalabilidad de los sistemas hacia más clientes.

¹⁰ Consiste en un modelo en el cual, en el caso del Sr. compost, se instalan en la esquina de Av. Pocuro con Av. Lyon en Providencia, para recibir cada domingo los residuos orgánicos de cocina de cualquier familia.

Otra cosa que se evidencia en base al estudio es que se está generando una fuerte red de colaboración entre entidades que partieron siendo independientes, pero que se dieron cuenta que si se unían podían hacer un sistema más completo y mejor, como es el caso del Sr. Compost, Urbanátika y Reciclo [16].

A continuación, se presenta una tabla resumen con las distintas iniciativas de gestión de residuos en Chile:

Empresa	Orientado a	Características principales	Sensores o métodos	Cuenta con SI
Urbanátika	• Empresas • Oficinas • Casas	• Compostaje rápido • Internet del compostaje • Sensores IoT	• No	Sí
Ecoven	• Personas naturales	• Sistema de ecopuntos • Aplicación móvil	• GPS de los contenedores	App
Reciclapp	• Personas que quieran reciclar sus residuos (Recicladores) • Personas que lleven los residuos a los centros de reciclaje (Recolectores)	• Gestión de recolección domiciliar planificada • Sistema de premios	• No	App
Bicicla	• Personas que quieran reciclar (Recicladores) • Personas que recolecten los residuos en bicicleta (Recolectores)	• Uso de vehículos con 0 emisiones de CO2 (Bicicleta) • Funcionamiento tipo Uber • Sistema de premios que pueden ser Bitcoins	• No	App
Karubag	• Hogares • Oficinas • Empresas	• Suscripción mensual contempla entrega de hummus	• No	No

Compost Urbano	Hogares	Subscripción mensual contempla entrega de hummus	No	No
Sr. Compost	- Hogares - Personas naturales	- Mecanismo Dropoff gratuito - Uso de tecnología EM para compostaje - Subscripción contempla entrega de hummus	No	No

Tabla 2. Comparación emprendimientos gestores de residuos orgánicos en Chile. Fuente: Elaboración propia.

Como se evidencia en la tabla, los emprendimientos que tienen algún sistema de información orientado a los usuarios son: Urbanátika, Ecoven, Reciclapp y Bicicla, de los cuales los tres últimos se refieren a aplicación móvil que cumple la función de unificar a las distintas partes involucradas para la gestión de los residuos, como es el caso de Reciclapp y Bicicla, que junta mediante la *app* a los recicladores y recolectores. A su vez, Ecoven muestra la localización de los contenedores y también los puntos que tiene acumulado el usuario.

Urbanátika se ve que tiene una vista más completa en cuanto a un sistema de información que apoye los procesos y permita visualizar datos de estos a los usuarios, y se infiere que se refieren a un *SaaS*, sin embargo, a la fecha aún no es posible ver algún prototipo o sistema funcional operativo.

Los tres emprendimientos restantes; Karubag, Compost Urbano y Sr. Compost, se concluye en base a videos analizados y a la lectura que ofrecen en internet, que no se apoyan de sistemas de información para la gestión de sus operaciones, lo cual, como se menciona anteriormente, abre una oportunidad para el desarrollo de un *SI* que les ayude a tener un control sobre los datos del sistema y que les permita, a la larga, ser escalables y generar una base de datos de la gestión de residuos orgánicos, que como se mencionó en un inicio de esta memoria, no existe.

2.3. MARCO CONCEPTUAL

A continuación, se definirán conceptos que son fundamentales para comprender la solución que se propone en el capítulo 3. Estos abarcan tanto las tecnologías que se utilizaron para el desarrollo del prototipo a nivel de código, como las tecnologías de computación en la nube usadas para la arquitectura de la aplicación.

2.3.1. API REST

El término *REST* (Representational State Transfer) se originó en el año 2000, descrito en la tesis de Roy Fielding, padre de la especificación *HTTP*. Un servicio *REST* no es una arquitectura de software, sino un conjunto de restricciones con las que es posible crear un estilo de arquitectura de software, que puede ser usada para crear aplicaciones web respetando *HTTP* [17].

Hoy en día la mayoría de las empresas utilizan *API REST* para crear servicios. Esto se debe a que es un estándar lógico y eficiente para la creación de servicios web.

Según Fielding, las restricciones que definen a un sistema *RESTful* son:

- **Cliente-servidor:** esta restricción mantiene al cliente y al servidor débilmente acoplados. Esto quiere decir que el cliente no necesita conocer los detalles de implementación del servidor, y el servidor se “despreocupa” de cómo son usados los datos que envía al cliente.
- **Sin estado:** Cada petición que recibe el servidor debería ser independiente, es decir, no es necesario mantener sesiones.
- **Cacheable:** Debe admitir un sistema de almacenamiento en caché. La infraestructura de red debe soportar una caché de varios niveles. Este almacenamiento evitará repetir varias conexiones entre el servidor y el cliente para recuperar un mismo recurso, lo cual lo hace más eficiente en el uso de datos y recursos del sistema.
- **Interfaz uniforme:** Define una interfaz genérica para administrar cada interacción que se produzca entre el cliente y el servidor de manera uniforme, lo cual simplifica y separa la arquitectura. Esta restricción indica que cada recurso del servicio *REST* debe tener una única dirección o “URI”.
- **Sistema de capas:** El servidor puede disponer de varias capas para su implementación, esto ayuda a mejorar la escalabilidad, el rendimiento y la seguridad.

2.3.2. SOFTWARE COMO SERVICIO (SaaS)

Es un modelo de distribución de software en el cual el soporte lógico y los datos que maneja el sistema se alojan en servidores de una compañía de tecnologías de información y comunicación (*TIC*), a los que se accede vía Internet desde un cliente [18].



Figura 8. Software as a Service. Fuente: <https://www.racemi.com/saas-advantages-and-disadvantages/>.

Una de las ventajas más importantes de tener un *SaaS* es que la empresa proveedora *TIC* se ocupa del servicio de mantenimiento, operación diaria y soporte del software usado por el cliente.

2.3.3. IOT, SENSORES Y BIG DATA

El término *IoT* proviene del inglés *Internet of Things*, se refiere a tener dispositivos computacionales, máquinas mecánicas y digitales, objetos, animales o personas que proveen un identificador único (*UIDs*) interrelacionados entre sí mediante Internet, estos deben tener la habilidad de transferir datos mediante Internet sin requerir una interacción humano a humano o humano a computador [19].

En el caso de esta memoria, se hará el uso de sensores para censar los pesos de los residuos provenientes de los usuarios del sistema, para más información leer el capítulo 3.

El concepto de *IoT* está fuertemente ligado al concepto de *Big Data*, puesto que, por ejemplo, mediante el uso de sensores, un sistema irá adquiriendo datos, y en el caso que sea un sensor que mide datos con una frecuencia alta, entonces generará grandes volúmenes de datos que serán alojados en una base de datos, lo que a través del tiempo genera lo que conocemos hoy como *Big Data*.

Big Data se orienta a buscar caminos para analizar, extraer información sistemáticamente o lidiar con *data sets* que son muy grandes o complejos para ser tratados por un procesamiento de datos tradicional.

Big data está asociado con tres conceptos claves: *Volumen*, *Variedad* y *Velocidad* [20].

2.3.4. BASES DE DATOS NO RELACIONALES (NOSQL)

El término *NoSQL* surge como una alternativa a las bases de datos relacionales tradicionales en las cuales la data es almacenada en tablas que poseen un *Schema* y que son cuidadosamente diseñadas antes de construir el *dataset*. Las bases de datos no relacionales son especialmente útiles para trabajar con grandes volúmenes de datos.

Una de las principales características de este tipo de base de datos, es que no usan el lenguaje *SQL* como lenguaje principal de consultas. Además, los datos, al ser *Schemaless*, no poseen una estructura rígida como las tablas en las bases de datos relacionales.

Las bases de datos *NoSQL* se clasifican según su forma de almacenar los datos, y comprenden categorías como *clave-valor*, las implementaciones de *BigTable*, bases de datos documentales, y bases de datos orientadas a grafos [21].

2.3.5. ARQUITECTURA DE MICROSERVICIOS

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 [22].

El uso de *Docker* permite crear contenedores que se encargan de mantener en un ambiente aislado todo lo necesario para ejecutar una aplicación, esto es, códigos fuentes, dependencias y variables de entorno, comandos para la ejecución, etc.

Una arquitectura basada en **Microservicios** comprende la comunicación entre varios servicios pequeños y autónomos que trabajan en conjunto. Lo anterior sumado al concepto de la **contenerización**, ayudan a construir sistemas sin precedentes. La **contenerización** cambia no sólo la arquitectura de los servicios, sino también la estructura de ambientes utilizados para crearlos.

Para crear una aplicación que siga una arquitectura orientada a microservicios, se debe tener en consideración que cada contenedor será un servicio pequeño y aislado, el cual se comunicará con los otros servicios mediante Internet (*HTTP* o *TCP*).

Para poder orquestar múltiples contenedores existe la herramienta llamada Docker Compose, la que se encarga, mediante un archivo de configuración *YAML* (con extensión *.yaml*), de proveer todas las configuraciones que requiere el sistema, de tal forma que se pueda levantar con el uso de un solo comando [23].

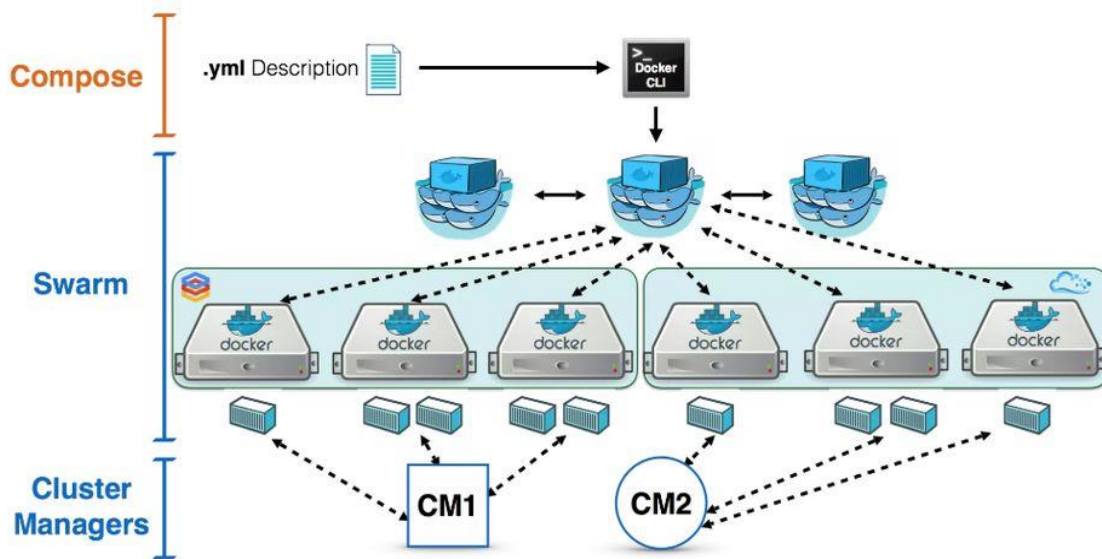


Figura 9. Arquitectura de Microservicios orquestada por Docker Compose. Fuente: Medium.

2.3.6. APLICACIONES MULTIUSUARIOS Y ACCESS CONTROL LIST

Una aplicación multiusuario es un software que permite el ingreso de más de un *tipo* de usuario, estos tipos de usuario se pueden ver como “roles”, ya que cada tipo de usuario o rol tendrá un contenido distinto proporcionado por el sistema, así como una serie de permisos especialmente liberados para ese tipo de usuario, de esta forma se mantiene una separación en la lógica de uso de la aplicación para cada uno de ellos.

Debido a lo anterior, surge el término *Access Control List* o *ACL*, que como su nombre lo indica, es una lista de controles de acceso que le dice a un software qué tipo de acciones sobre ciertos recursos puede realizar o no un usuario (o rol) en particular.

2.3.7. BACKEND Y FRONTEND

Backend es la capa de acceso a datos de un software o cualquier dispositivo, que no es directamente accesible por los usuarios, además, contiene la lógica de la aplicación que maneja dichos datos.

El *Backend* también accede al servidor, que es una aplicación especializada que entiende la forma en que el navegador solicita recursos o accesos.

Por otro lado, el *Frontend* es la parte de un programa o dispositivo a la que un usuario puede acceder directamente. Son todas las tecnologías de diseño y desarrollo web que corren en el navegador y que se encargan de la interacción con los usuarios [24].

En este proyecto de memorias multidisciplinarias se desarrolló tanto *Backend* como *Frontend*, sin embargo, en esta memoria en particular se profundiza en la parte de *Backend*.

2.3.8. TECNOLOGÍAS DE DESARROLLO PARA EL BACKEND

Tanto para el *Backend* como para el *Frontend* existen diversas tecnologías que permiten un desarrollo ágil y efectivo de dichos componentes, las tecnologías que se utilizaron se detallan a continuación.

2.3.8.1. LOOPBACK.IO

LoopBack es un *framework* de *Node.js* altamente extensible, que permite crear *APIs REST* escalables, tanto para aplicaciones móviles como para plataformas web convencionales [25]. Es un *framework* que actúa del lado del servidor (*Backend*). Posee una extensa documentación, permitiendo la integración de diversas librerías y paquetes de terceros, así como también la conexión de múltiples bases de datos, incluyendo *queries* por defecto capaces de hacer uso de todo el *Schema* como una gran base de datos, sin importar cuántas estén involucradas. Tiene la capacidad de integrar tecnologías como *AWS S2* (sistema de archivos *Amazon Web Services*), entre muchas otras funcionalidades. Además, posee una integración con *Swagger.io* para visualizar de una manera ordenada y agradable una documentación funcional de todos los *endpoints* generados por el *Schema*.

LoopBack cuenta con un sistema de seguridad basado en *ACL*, en forma de *RBAC (Role-Based Access Control)*, donde existen dos tipos de roles:

- **Roles estáticos:** Son aquellos roles que están determinados explícitamente dentro de un sistema de información. Por ejemplo, un usuario puede tener un permiso de administrador y eso está físicamente estipulado en la base de datos por medio de una dependencia u atributo.
- **Roles dinámicos:** Son roles más avanzados, que pueden incluso utilizar cálculos y operaciones indirectas para poder saber si realmente un usuario en particular cumple con los requisitos para operar algún recurso en particular.

En esta memoria se utiliza este *framework*, pues permite hacer un sistema escalable y al estar montado sobre *NodeJS*, se obtiene una buena *performance* en cuanto a los tiempos de respuesta.

2.3.8.2. MONGODB

MongoDB es un programa de plataforma cruzada [26] de base de datos orientada a documentos. Utiliza documentos tipo *JSON* con *Schemaless*. Forma parte de las bases de datos denominadas *NoSQL*, ha sido desarrollada por MongoDB Inc. Y licenciada bajo el *SSPL (Server Side Public License)* [27].

Entre sus características principales se encuentran:

- **Ad hoc queries:** Búsquedas que retornan campos específicos de los documentos, pueden ser filtrados por rango o por alguna expresión regular, además soporta el uso de funciones *JavaScript* definidas por el usuario.

- **Indexing:** Concepto importante ya que permite en *MongoDB* y en cualquier base de datos indexar sus documentos con el índice primario y secundario, para más información de *index* en bases de datos, referirse a [28].
- **Replication:** *MongoDB* provee alta disponibilidad con replica sets. Un replica set consiste de dos o más copias de la data. Cada replica set puede actuar en el rol de nodo primario o nodo secundario en el tiempo. Todas las escrituras y lectura son hechas en el nodo primario por default. El nodo secundario mantiene una copia de la data del primario usando *built-in replication*. Cuando un nodo primario falla, el *replica set* automáticamente lleva a un proceso de elección para determinar cuál debiese ser el nodo secundario.
- **Aggregation Framework:** Es una de las características claves de *MongoDB*, ya que permite hacer uso de características como las funciones *map-reduce*. *MongoDB* provee tres tipos de *aggregation*: *aggregation pipeline*, funciones *map-reduce* y métodos de un solo propósito.
Map-reduce puede ser usado para procesar un *batch* de data en las operaciones de *aggregation*. Pero acorde a la documentación de *MongoDB*, el *aggregation pipeline* provee mejor *performance* para la mayoría de las operaciones *aggregation*.
El *aggregation framework* permite a los usuarios obtener el tipo de resultado para el cual se usa la cláusula de *SQL GROUP BY*. Los operadores del *aggregation* pueden ir en secuencia formando un *pipeline*. El *aggregation* además incluye el operador *\$lookup* con el cual se puede realizar el *Join* de documentos de distintas *collections*, o bien hay operadores estadísticos para calcular distintas métricas como la *desviación estándar*.

En esta memoria se utiliza esta base de datos ya que cumple con las características requeridas y, además, proveen un servicio en línea llamado *MongoDB Atlas*, el cual es una *Database as a Service*, que entrega diversas facilidades para el desarrollo del prototipo, en especial la escalabilidad, alto rendimiento y alta disponibilidad [27].

2.3.8.3. DOCKER

La tecnología Docker usa el *kernel* de Linux y las funciones de este, como *Cgroups* y *namespaces*, para segregar los procesos, de modo que puedan ejecutarse de manera independiente. El propósito de los contenedores es esta independencia: la capacidad de ejecutar varios procesos y aplicaciones por separado para hacer un mejor uso de su infraestructura y, al mismo tiempo, conservar la seguridad que tendría con sistemas separados.

Las herramientas del contenedor, como Docker, ofrecen un modelo de implementación basado en imágenes. Esto permite compartir una aplicación, o un conjunto de servicios, con todas sus dependencias en varios entornos. Docker también automatiza la implementación de la aplicación (o conjuntos combinados de procesos que constituyen una aplicación) en este entorno de contenedores.

¿Qué es un container?

Un *container* es una unidad estándar de software que empaqueta código y todas sus dependencias para que la aplicación pueda ejecutarse de manera rápida y segura desde un entorno computacional a otro. Una imagen de un contenedor *Docker* es un paquete de software liviano, autónomo, que incluye todo lo necesario para ejecutar una aplicación: código, tiempo de ejecución, herramientas de sistema, bibliotecas del sistema y configuraciones (Ver figura 10).

Las imágenes de contenedores se convierten en contenedores en tiempo de ejecución y en el caso de los contenedores *Docker*, las imágenes se convierten en contenedores cuando se ejecutan en *Docker Engine* [29].

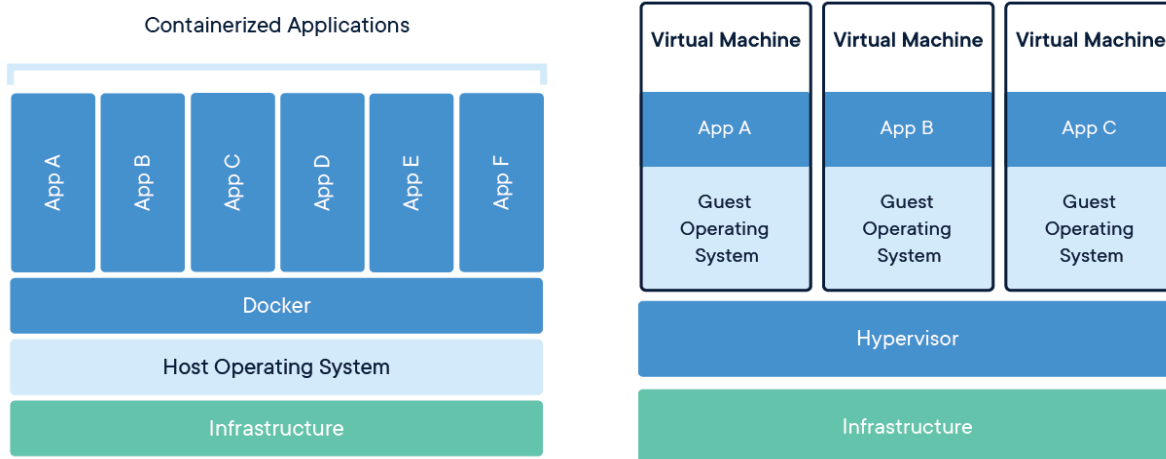


Figura 10. Containerized applications vs VMs. Fuente: <https://www.docker.com/>

2.3.8.4. DIGITAL OCEAN SERVER

DigitalOcean, Inc. es una empresa americana proveedora de infraestructura en la nube con sede en la ciudad de New York con data centers en todo el mundo. Provee a los desarrolladores servicios en la nube que ayudan a desplegar y escalar aplicaciones que se ejecutan simultáneamente en múltiples computadores.

2.3.8.5. JSON WEB TOKEN

JSON Web Tokens es un método para implementar seguridad en un sistema, poniendo una capa de verificación y seguridad entre el cliente y el *endpoint* o recurso al que se está intentando acceder. Está definido por el estándar *RFC 7519* y funciona mediante la generación de *tokens* que permiten acceder a los recursos. Para entender el flujo de funcionamiento ver la figura 11.

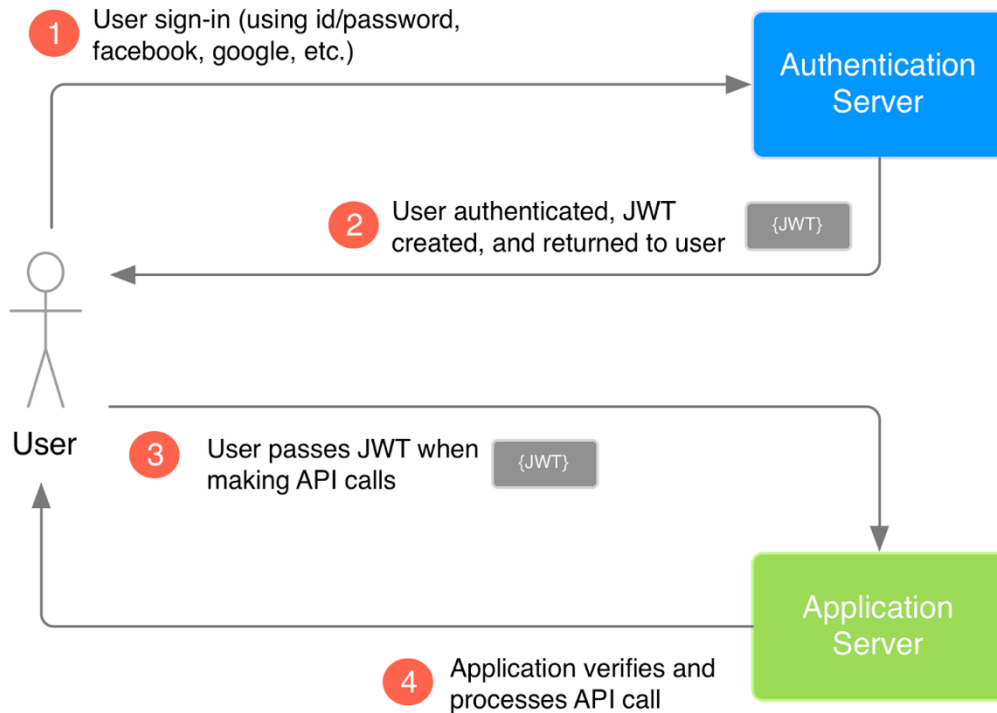


Figura 11. JSON Web token flujo. Fuente: Medium.

2.3.8.6. NGINX: REVERSE PROXY SERVER

NGINX es un proyecto de código abierto que permite levantar servidores web, como también servidores *proxy* o *reverse proxy*, siendo este último el tipo de servidor que permite el redireccionamiento de puertos a servicios corriendo internamente. Para más información referirse a [30].

2.3.8.7. POSTMAN

POSTMAN surge originalmente como una extensión para el navegador Google Chrome. Hoy, existen aplicaciones nativas para Mac y Windows y están trabajando en una aplicación nativa para Linux. En este trabajo se utiliza esta herramienta para la validación de la solución mediante test para las distintas funcionalidades y los distintos tipos de usuario.

Permite entre otras cosas, cargar un archivo *.yaml* con todas las llamadas y comportamientos de una *API REST*, validar cada endpoint y resultados esperados mediante *test* y mantener distintos tipos de ambiente guardados en variables como lo son los distintos tipos de usuario (*Roles* y sus respectivos *AccessToken*).

Para más información acerca de esta herramienta, referirse a [31].

CAPÍTULO 3: PROPUESTA DE SOLUCION

3.1. LA SOLUCIÓN

La solución propuesta a nivel de equipo participante del programa de Memorias Multidisciplinarias UTFSM fue elaborar un sistema de gestión de residuos que apoyado en tecnología de vanguardia permita administrar y optimizar los procesos asociados al manejo de residuos orgánicos en comunidades o condominios. Mediante componentes de *IoT* (uso de sensores), se logra capturar datos referentes a las cantidades de residuos que provienen de los residentes de la comunidad, así como la hora y fecha en los que el residente dispone sus residuos para ser retirados por un reciclador.

Dicha solución consiste en desarrollar un software como servicio (*SaaS*) para la gestión de residuos orgánicos en comunidades, esto es, una plataforma que permita gestionar mediante un panel de administración a los distintos tipos de usuario que el sistema requiere, así como los distintos tipos de operaciones y procesos que estos usuarios puedan realizar en el sistema.

Para elaborar la solución se hará uso de la metodología Agile¹¹, la cual da énfasis en realizar un desarrollo incremental, de esta forma el desarrollo se hará en iteraciones. Se usará, además para elaborar los distintos diagramas, definir los requerimientos y documentación en general, lo propuesto en Agile Modeling [32].

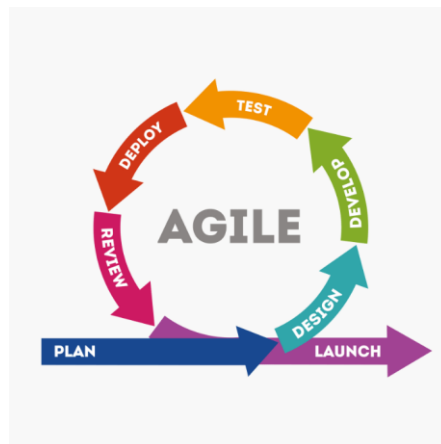


Figura 12. Ciclo de vida de un desarrollo con metodología Agile. Fuente:

<https://www.cognodata.com/blog/principios-metodologia-agile-desarrollo-proyectos/>

3.1.1. DESCRIPCIÓN GENERAL Y METODOLOGÍA DE DESARROLLO

Como se mencionó en el apartado anterior, el desarrollo de esta memoria consiste en la creación de un Software as a Service que comprende de una *API REST* para el intercambio de recursos entre el cliente o *Frontend* y la base de datos. Existen diversos actores que están involucrados en este sistema, tales como los Residentes, Recicladores, Administradores y Super Administradores, por lo que es necesario contar con un

¹¹ El software de desarrollo Agile se refiere a un grupo de metodologías aplicadas en la creación de software que basa su desarrollo en un ciclo iterativo, en el que las necesidades y soluciones evolucionan a través de la colaboración entre los diferentes equipos involucrados en el proyecto.

sistema de seguridad basado en Roles, para otorgarle mediante un *ACL*, los permisos necesarios a cada tipo de usuario para realizar las operaciones que se definirán más adelante.

Es importante repetir que esta memoria se enfoca en la *API REST* que actúa de intermediario entre el *Frontend* y la base de datos, también puede decirse que se trata del *BACKEND* del sistema de gestión de residuos.

El producto final que se presentará consta de una *API REST* funcional, donde se validará mediante los distintos casos de uso el cumplimiento de los requerimientos propuestos.

Por temas de simplicidad, nos abstraeremos de la componente *IoT*, y simularemos los sensores mediante funciones que envíen datos cuando se ejecuten los *endpoints* asociados.

3.2. ARQUITECTURA DE LA SOLUCIÓN

Tanto el *Backend* como el *Frontend* de la solución estarán alojados en un servidor de *DigitalOcean* y vinculados a un *Reverse Proxy* usando *Nginx*. Lo anterior es posible usando una arquitectura orientada a microservicios, orquestando todo con *Docker-Compose*.

Se utiliza una base de datos como servicio (*Database as a Service o DaaS*), lo cual permite abstraerse de la infraestructura asociada a dicha base de datos.

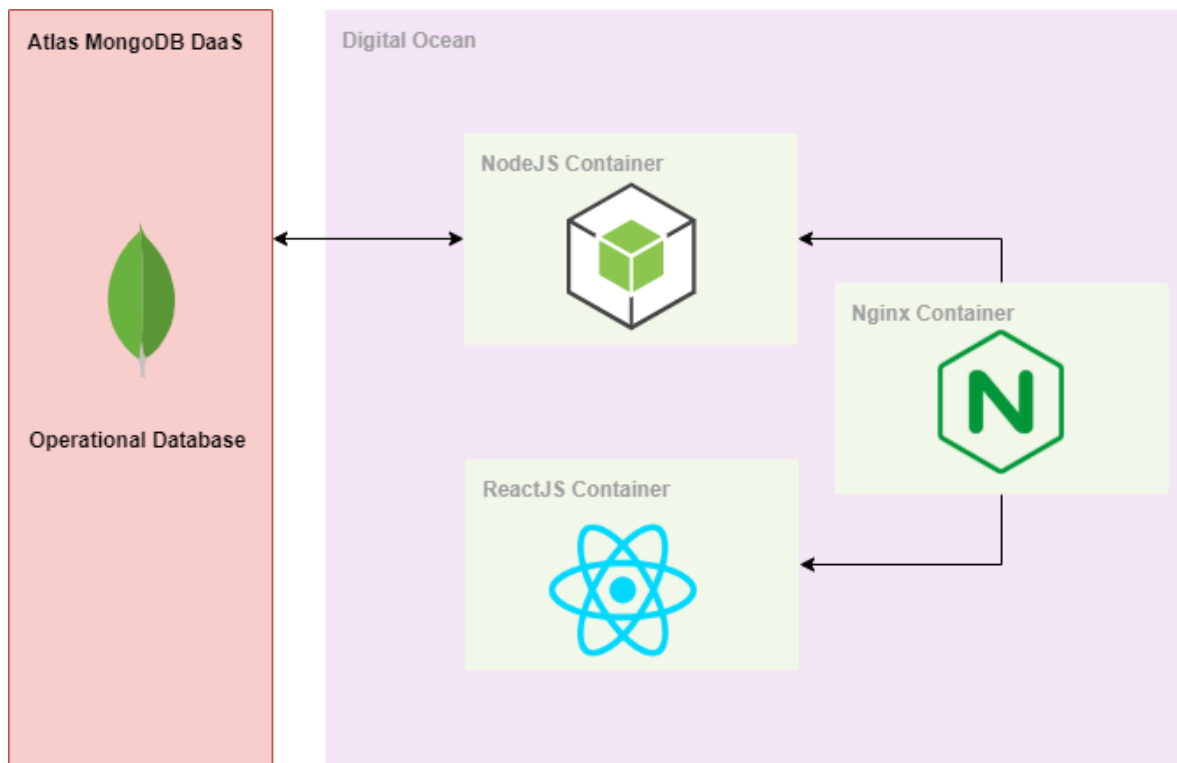


Figura 13. Diagrama de Arquitectura de la solución. Fuente: Elaboración propia.

3.3. USUARIOS DE LA SOLUCIÓN

La solución en un inicio constaba de 4 tipos de usuario, pero finalmente se redujeron a tres, de modo de hacer el sistema más general eliminando la capa de administración de las comunidades.

USUARIO	DESCRIPCIÓN
RESIDENTE	Es el residente del domicilio, el generador del residuo y por tanto quien interactuará con el sistema para poder gestionar la recolección de dichos residuos.
RECICLADOR	Es la persona que se encargará de ir a recolectar los residuos a la casa del residente, y a través de una balanza inteligente enviará los datos del peso del residuo al sistema almacenado en la nube.
CONCIENCITY (ADMINISTRADOR)	Administrador del sistema, podrá visualizar estadísticas y métricas de la gestión de residuos orgánicos provistas por el sistema propuesto.

Tabla 3. Usuarios de la solución. Fuente: Elaboración propia.

3.4. DIAGRAMA DE DOMINIO

El diagrama de dominio relata el contexto general de la solución, es un diagrama muy importante ya que permite saber el alcance de la solución, y poder tener una visión general de las entidades y las relaciones que hay entre ellas, aunque sin llegar a un nivel de detalle mayor como en el caso de los diagramas de clase o modelos de datos.

A continuación, se presenta el diagrama de dominio inicial:

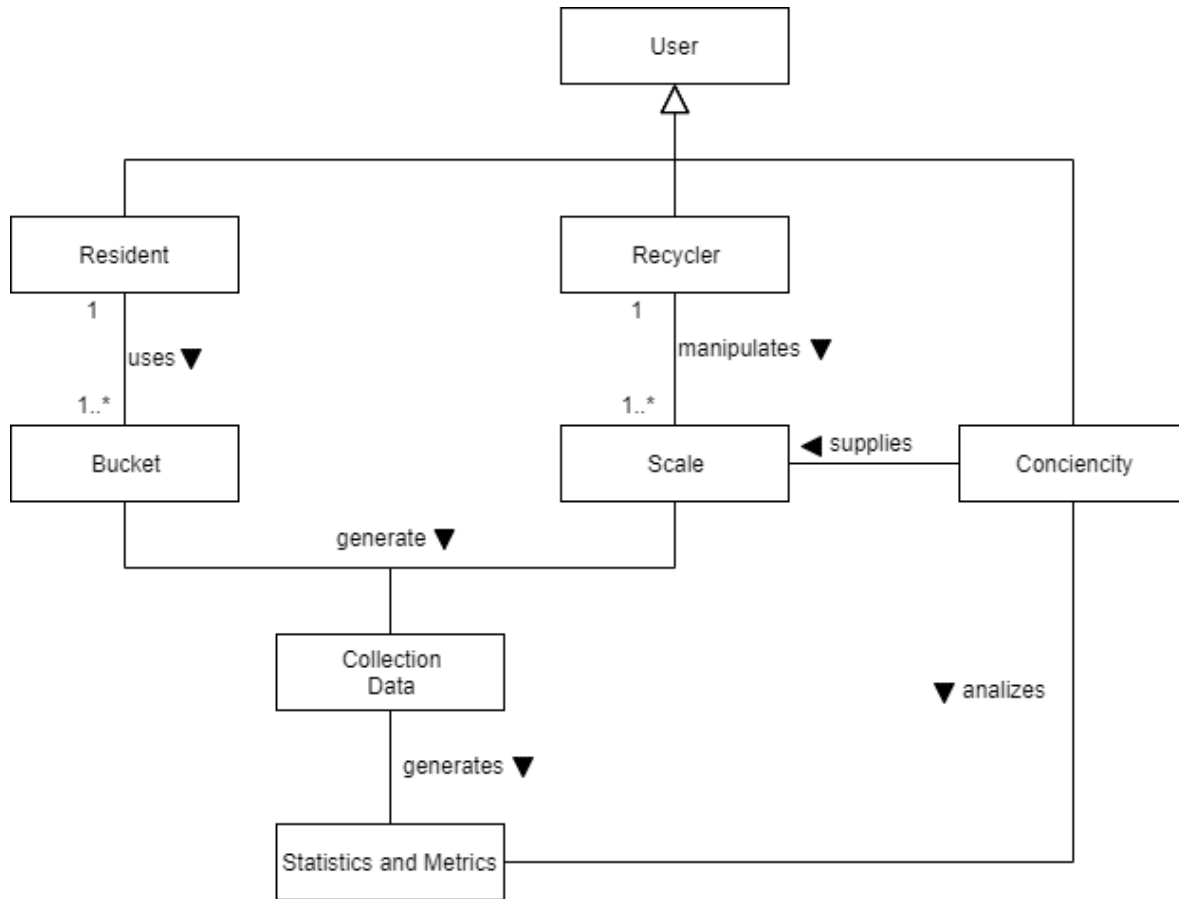


Figura 14. Diagrama de dominio de la solución. Fuente: Elaboración propia.

En el diagrama de la figura 14 se puede ver que tanto el usuario Residente como el Reciclador heredan atributos de la entidad Usuario. Además, se visualiza la pertenencia de activos como el *Bucket* para el Residente y la Balanza o *Scale* para el Reciclador, donde la Balanza censará el Bucket para generar datos de recolección, los cuales serán procesados para generar métricas y estadísticas que serán consultadas y analizadas por Conciency (que es la entidad administradora del sistema), de esa forma podrá visualizarse información de la recolección y gestión de residuos orgánicos. Cabe mencionar también que la entidad Conciency es la que va a proveer los activos como las Balanzas para los Recicladores y los Buckets para el Residente, aunque en el diagrama sólo se explicita para el caso de la Balanza, por simplicidad.

3.5. HISTORIAS DE USUARIO E ITERACIONES

El desarrollo del proyecto se divide en 4 iteraciones de 2 semanas cada una, donde en cada una se desarrollan algunas de las historias de usuario que se detallan a continuación:

Iteración	Historias de usuario
1	• Registrar usuarios: Residente y Reciclador. • Editar usuarios Residente y Reciclador. • Eliminar usuario: Residente. • Eliminar usuario: Reciclador.
2	• Crear Bucket en el sistema. • Crear Balanza en el sistema. • Asignar Bucket a un Residente. • Asignar Balanza a un Reciclador.
3	• Censar el peso de un Bucket con la Balanza. • Ver el historial de censado de una Balanza. • Ver el historial de recolección para un Bucket.
4	• Ver estadísticas de recolección. • Filtrar estadísticas por comuna. • Filtrar estadísticas por tiempo.

Tabla 4. Historias de usuario e iteraciones. Fuente: Elaboración propia.

Para el desarrollo de este proyecto, y debido a que se usa Agile Modeling con ciclos iterativos de 2 semanas, se lleva a cabo el desarrollo y cumplimiento de las historias de usuario que se describen para cada iteración en la tabla.

Las historias de usuario se escriben inicialmente de una manera muy general, y se van enriqueciendo al pasar el tiempo, normalmente eso se hace durante un proceso de feedback y discusión con el cliente.

3.6. ITERACIÓN 1: HISTORIAS DE USUARIO

Para esta iteración, las historias de usuario que deben desarrollarse son:

1. Como usuario **Residente**, quiero **registrarme** en el **Sistema** ingresando mis datos personales para comenzar a gestionar mis residuos orgánicos.
2. Como usuario **Reciclador**, quiero **registrarme** en el **Sistema** ingresando mis datos personales para comenzar a **recolectar** los residuos orgánicos de los **Residentes**.
3. Como usuario **Residente y Reciclador**, quiero visualizar mis datos del sistema tales como mi nombre, mi correo, mi nombre de usuario, dirección, etc.
4. Como usuario **Residente y Reciclador**, quiero **editar** mis datos conforme lo necesite, para poder contar con un dinamismo que me permita, por ejemplo, cambiar mi ubicación del **sistema** y que esto no interfiera en la funcionalidad de la gestión.
5. Como usuario **Residente**, quiero **eliminar** mi cuenta del **sistema**, en caso de que no quiera seguir participando de la gestión de residuos orgánicos.
6. Como usuario **Reciclador**, quiero **eliminar** mi cuenta del **sistema**, en caso de que no quiera o no pueda continuar realizando las labores de **Reciclador**.

3.6.1. ITERACIÓN 1: MODELO DE DATOS

El modelo de datos se va construyendo incrementalmente, de forma de ir abarcando progresivamente las historias de usuario que se definieron para cada iteración.

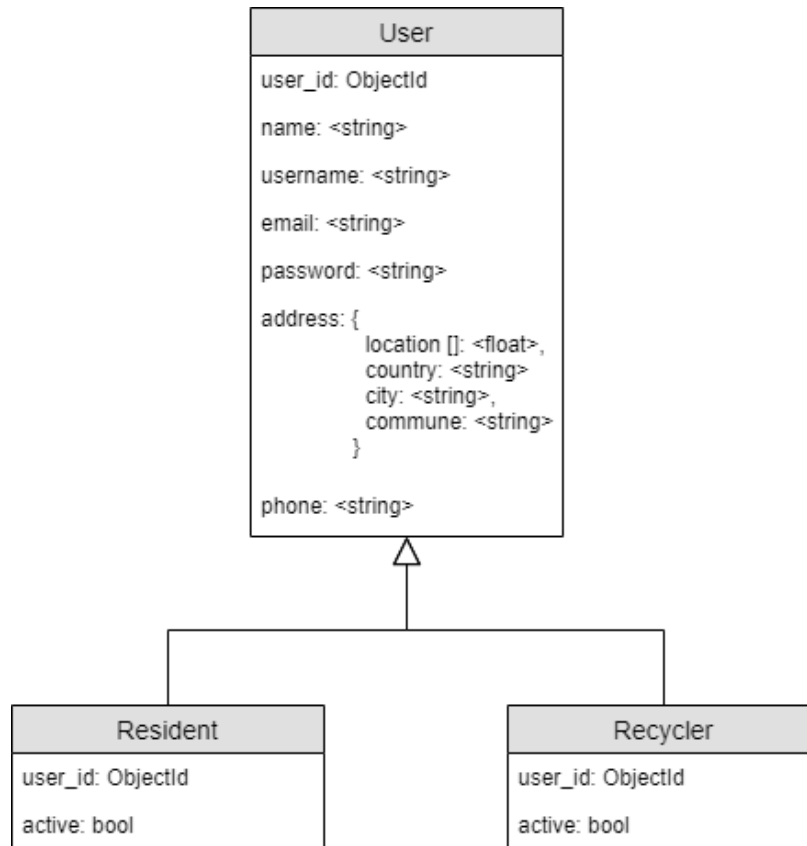


Figura 15. Modelo de datos para la iteración 1. Fuente: Elaboración propia.

Se visualiza que el Usuario mediante el campo *address*, tiene información relacionada a su ubicación geográfica, ese campo es muy importante porque permite hacer un análisis territorial sobre la gestión de residuos, además de ser clave para la gestión de rutas de recolección, uno de los puntos fuertes que se mencionan al inicio de esta memoria.

Las entidades Residente y Reciclador heredan de la entidad User, y por ahora sólo se pone el *_id* de cada uno y el atributo *active* que indicará si el usuario está activo, es decir, que está participando del proceso de recolección.

3.6.2. ITERACIÓN 1: REQUERIMIENTOS FUNCIONALES

Estos requerimientos indican las funciones del *software*. Para ello es necesario estipular cada funcionalidad como un conjunto de entradas, comportamientos y salidas.

Para la iteración 1, el sistema debe cumplir con:

1. Cualquier persona que cumpla el rol de *Residente* o *Reciclador* pueda registrarse en el sistema, **debe** ingresar datos personales como su nombre, dirección física, dirección de correo electrónico, y una contraseña para ingresar.
2. Una vez que un *Reciclador* se registre en el sistema, se notificará al usuario *Conciencity* para que este valide su ingreso y gestione la asignación de una *Scale* para el *Reciclador*, con el fin de que este pueda comenzar a gestionar los residuos orgánicos de los *Residentes* del sistema.
3. Los usuarios *Reciclador* y *Residente* deben ser capaces de visualizar todos sus datos registrados en el sistema (datos referentes a la cuenta). Un usuario de este tipo sólo podrá visualizar sus propios datos y no los datos de otros usuarios.
4. Los usuarios *Residentes* o *Recicladores* deben ser capaces de editar sus datos en el sistema, es decir, podrán editar su nombre, su dirección y su número de teléfonos. Por simplicidad no podrán editar su correo electrónico. Para poder realizar esta operación, los usuarios deberán cumplir con el rol de **Owner** del recurso, es decir, sólo podrán modificar sus *propios* atributos.
5. El usuario *Residente* debe ser capaz de eliminar su cuenta del sistema, en este caso se asigna el valor *False* al atributo *active* del modelo *Residente*, realizando un borrado lógico en el sistema.
6. El usuario *Reciclador* debe ser capaz de eliminar su cuenta del sistema, en este caso se asigna el valor *False* al atributo *active* del modelo *Reciclador*, realizando un borrado lógico en el sistema. Además, el *Reciclador* no podrá tener procesos de recolección pendientes para realizar esta acción.

3.6.3. ITERACIÓN 1: ENDPOINTS

La descripción de los endpoints que cumplen para los requerimientos anteriores son:

API ENDPOINT	DESCRIPTION	ROL
POST /RESIDENTS	Agrega un Residente al sistema.	Everyone
POST /RECYCLERS	Agrega un Reciclador al sistema.	Everyone
GET /RESIDENT/{ID}	Visualiza los datos de un Residente del sistema.	Owner
GET /RECYCLERS/{ID}	Visualiza los datos de un Reciclador del sistema.	Owner
PATCH /RESIDENTS/{ID}	Edita un Residente del sistema.	Owner
PATCH /RECYCLERS/{ID}	Edita un Reciclador del sistema.	Owner
DELETE /RESIDENTS/{ID}/CUSTOMDELETE	Elimina un Residente del sistema.	Owner
DELETE /RECYCLERS/{ID}/CUSTOMDELETE	Elimina un Reciclador del sistema.	Owner

Tabla 5. Endpoints y permisos para la iteración 1. Fuente: Elaboración propia.

3.7. ITERACIÓN 2: HISTORIAS DE USUARIO

1. Como usuario **Conciency**, quiero crear **Bucket** en el sistema y asignarlo a un **Residente**, para que el Residente tenga los implementos necesarios para comenzar a gestionar sus residuos orgánicos.
2. Como usuario **Conciency**, quiero crear **Scales** en el sistema, y asignarlo a un **Reciclador**, para que el Reciclador tenga los implementos necesarios y pueda comenzar a recolectar los residuos orgánicos de los Residentes.
3. Como usuario **Residente**, quiero activar mi perfil para poder gestionar los residuos orgánicos de mi hogar.
4. Como usuario **Reciclador**, quiero activar mi perfil para poder comenzar a recolectar y reciclar los residuos orgánicos de los residentes.

3.7.1. ITERACIÓN 2: MODELO DE DATOS

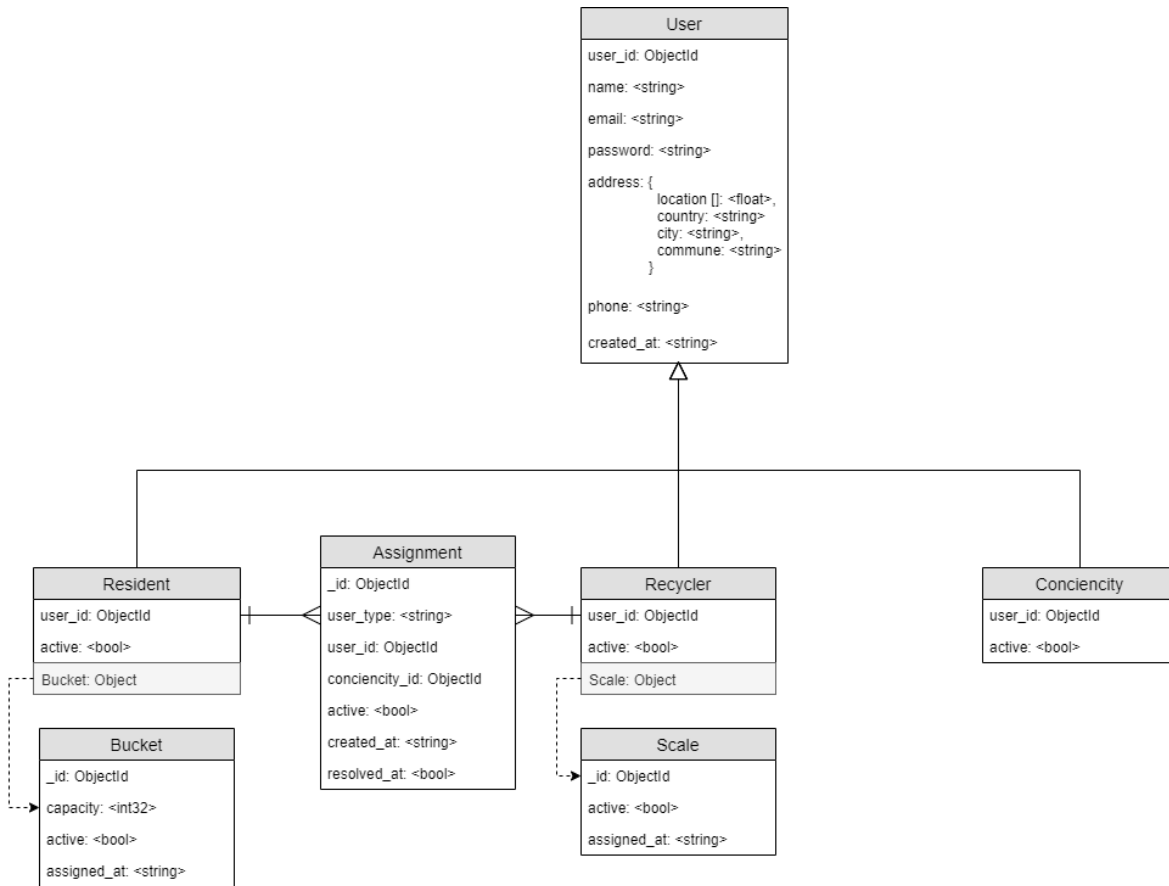


Figura 16. Modelo de datos para la iteración 2. Fuente: Elaboración propia.

Se observa que los objetos *Bucket* y *Scale* son elementos que irán *embebidos* dentro de las colecciones *Resident* y *Recycler*, respectivamente. Las relaciones serán que un *Resident* podrá tener embebido uno o

muchos *Buckets*, asumiendo que estos no serán *tantos*. De la misma manera, un *Recycler* tendrá embebido uno o muchos *Scales*.

Además, se observa la creación de la entidad *Assignment*, que será la encargada de guardar los registros de las asignaciones de *Buckets* para el *Resident* y de *Scales* para el *Recycler*. Esta tabla posee los atributos *user_type* que es el tipo de usuario que está solicitando la asignación, el cual puede ser *Resident* o *Recycler*, el atributo *active* permitirá saber si la solicitud ha sido resuelta o no, y el campo *conciency_id* será el *id* del administrador que la resuelva, además, se registrarán las fechas de creación de la solicitud (*created_at*), y la fecha de resolución de esta (*resolved_at*).

3.7.2. ITERACIÓN 2: REQUERIMIENTOS FUNCIONALES

1. El usuario *Conciency* será el encargado de crear en el sistema los objetos *Bucket* y *Scale*, esto lo hará mediante una interfaz que le permita realizar las operaciones *CRUD* de dichos objetos.
2. Cuando un *Resident* solicite la asignación de un *Bucket* (o un *Recycler* solicite la asignación de un *Scale*), se registrará dicha solicitud en la base de datos en la colección *Assignment*. Las asignaciones deberán ser resueltas por el usuario *Conciency*, asignando finalmente ya sea un *Bucket* para el *Resident* o un *Scale* para un *Recycler*.

3.7.3. ITERACIÓN 2: ENDPOINTS

API ENDPOINT	DESCRIPCIÓN	ROL
POST /Buckets	Crea un Bucket.	Administrador
GET /Buckets	Ver todos los Buckets.	Administrador
PATCH /Buckets/{id}	Edita un Bucket por <i>id</i> .	Administrador
DELETE /Buckets/{id}	Elimina un Bucket por <i>id</i> .	Administrador
POST /Scales	Crea un Scale.	Administrador
GET /Scales	Ver todas las Scales.	Administrador
PATCH /Scales/{id}	Edita un Scale por <i>id</i> .	Administrador
DELETE /Scales/{id}	Elimina un Scale por <i>id</i> .	Administrador
POST /Residents/{id}/Assignments	Crea una solicitud de asignación para el Residente.	Authenticated
GET /Residents/{id}/Assignments	Visualiza una solicitud del Residente.	Authenticated

POST /Recyclers/{id}/Assignments	Crea una solicitud de asignación para el Reciclador.	Authenticated
GET /Recyclers/{id}/Assignments	Visualiza una solicitud del Reciclador.	Authenticated
PUT /Residents/{id}/Assignments/{fk}	Edita una asignación asociada a un Residente.	Administrador (*)
PUT /Recyclers/{id}/Assignments/{fk}	Edita una asignación asociada a un Reciclador.	Administrador (*)
GET /Assignments	Lee todas las solicitudes creadas.	Administrador

Tabla 6. Endpoints y permisos para la iteración 2. Fuente: Elaboración propia.

(*) Este endpoint es súper importante, ya que permite editar la solicitud, es decir, se usará para resolverla por parte del administrador. El administrador, luego de ver todas las solicitudes que están pendientes (*active: true*), entonces podrá y deberá resolver algunas de estas. Luego de resolverlas, lo que se debe agregar a través de este endpoint es que el atributo *active* pasa a ser *false*, además, se agrega el *id* del administrador que la resolvió, y también un registro de la fecha y hora en la que fue resuelta (*resolved_at*).

Además, una vez resuelta una asignación, conforme se asigna un *Bucket* a un *Resident* o un *Scale* a un *Recycler*, entonces se deben asociar estos objetos al *Resident* o *Recycler*, el atributo *active* de dichos objetos pasa a ser *true*.

3.7.3.1. DIAGRAMA DE FLUJO PARA LA ASIGNACIÓN DE ACTIVOS

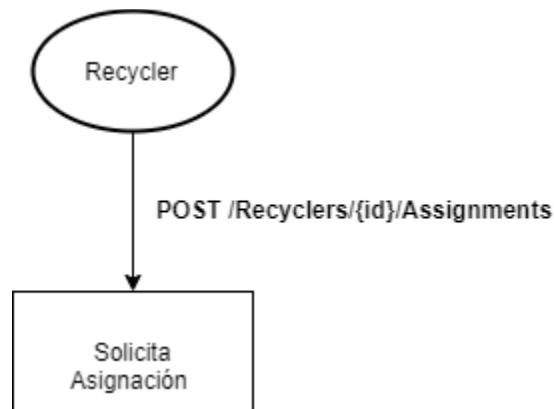


Figura 17. Primera parte del flujo de asignación de activos. Fuente: Elaboración propia.

En la ilustración 10 se visualiza que, para solicitar la asignación de un activo para el *Recycler*, en términos del *Backend*, se requiere enviar una *request* con el método *POST* al endpoint */Recycler/{id}/Assignments*. De esta forma se crea un registro en dicha colección, que como vimos en el modelo de datos, tendrá los datos necesarios para poder asignarle efectivamente el activo. Cabe mencionar que este flujo es válido tanto para

el *Recycler* como para el *Resident*, por lo que sólo se diagrama para el *Recycler*, por simplicidad.

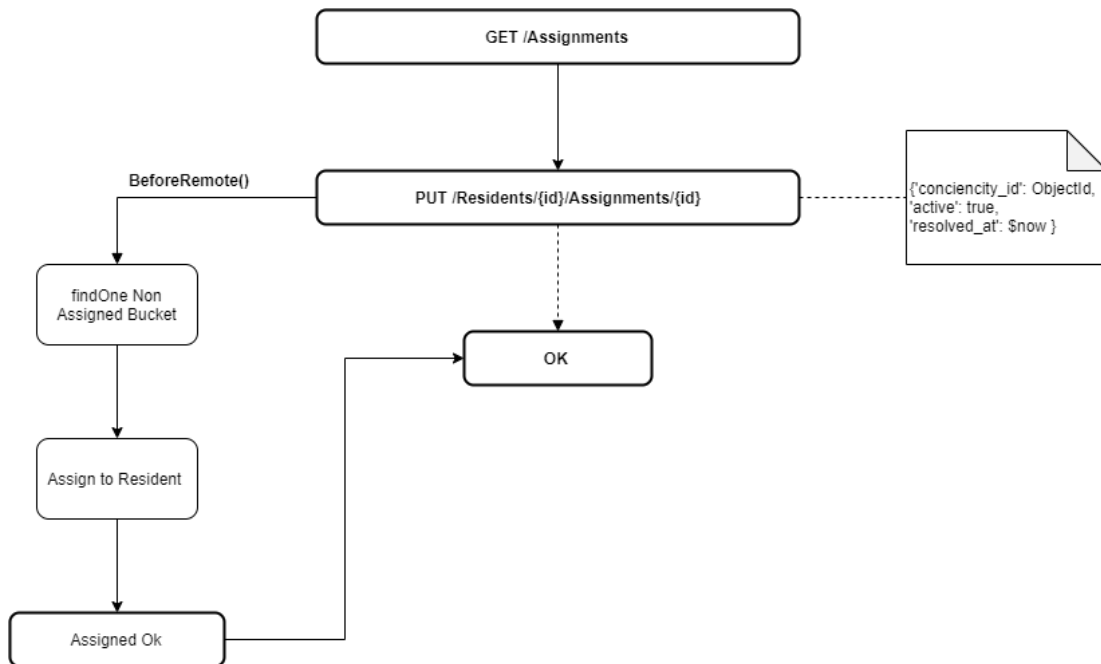


Figura 18. Parte dos del flujo de asignación de activos. Fuente: Elaboración propia.

Esta parte del flujo corresponde a la resolución de asignaciones, de la mano del *Administrador*. El flujo comienza por una petición que hace este usuario a la colección *Assignments* vía *GET*, la cual le retornará todos los registros que haya en esa colección, además, estos podrán ser filtrados por si están activos o no, y también por fecha, teniendo que resolver las asignaciones que tengan un tiempo más antiguo de registro. Una vez que se resuelve, entonces viene el paso final: Activar el objeto dentro de cada colección (*Bucket* y *Scale*) y una vez activo, relacionarlo al usuario al cual fue asignado (*Resident* o *Recycler*). Es importante mencionar que, al efectuar esta operación, se va a ejecutar un método remoto que tendrá lugar *antes* de que se persista la operación en la base de datos. Este método remoto se denomina *remote hook*, y lo que hará será modificar los documentos en la base de datos como se indica en la figura 18.

3.8. ITERACIÓN 3: HISTORIAS DE USUARIO

1. Como usuario **Reciclador**, quiero poder **censar** el peso de un **Bucket** de un Residente mediante la Balanza (**Scale**), con el fin de poder registrar el detalle de la recolección del residuo orgánico para el Residente, así como el peso y la hora del registro.
2. Como usuario **Reciclador**, quiero poder ver mi historial de recolecciones realizadas, con el fin de visualizar estadísticas relacionadas a mi participación en el sistema, tales como la cantidad de recolecciones que he realizado y la cantidad de kilos totales que he recolectado.

3. Como usuario **Residente**, quiero poder ver mi historial de recolección, con el fin de visualizar estadísticas relacionadas a la gestión de mis residuos, tales como la cantidad de veces que me han ido a recolectar los residuos y la cantidad de residuos que he gestionado en el sistema.

3.8.1. ITERACIÓN 3: MODELO DE DATOS

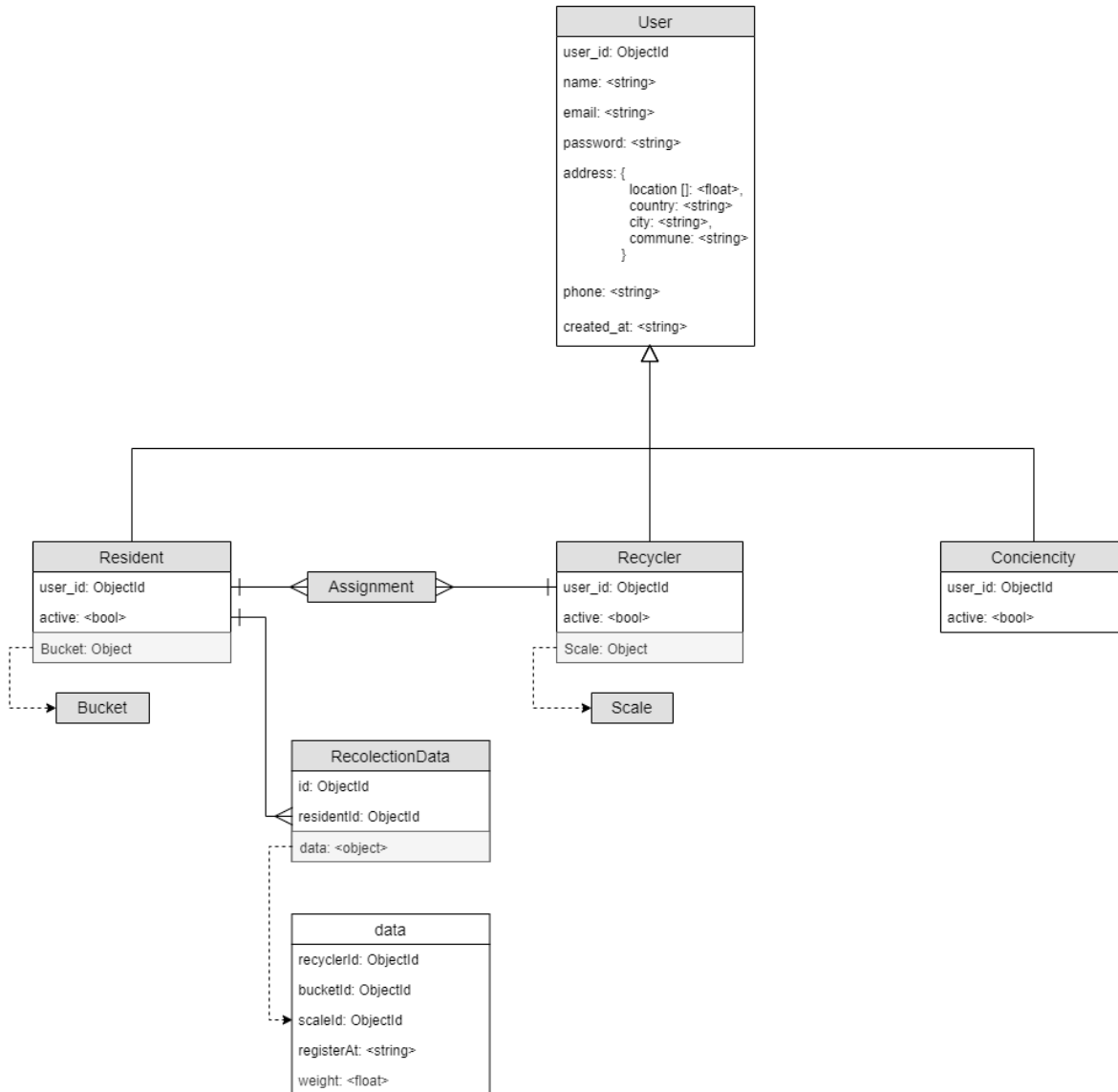


Figura 19. Modelo de datos de la iteración 3. Fuente: Elaboración propia.

En el modelo de la figura 19 vemos que se agrega la *Collection RecolectionData*, que será la encargada de almacenar los registros de los pesos de los residuos contenidos en el *Bucket* y realizados por el *Scale* (Reciclador). Será a través de esta tabla donde se podrán consultar mediante el lenguaje de consultas de *Mongodb* lo necesario para poder cumplir con la información requerida en las historias de usuario. *RecolectionData* está asociado al *Resident*, y en el campo *data* contendrá los datos referentes a la recolección,

esto es, el *id* del *Recycler* que está censando, el *Bucket* y el *Scale*, la fecha y hora del registro y el dato importante: el peso del residuo.

A nivel técnico, la colección posee una relación *embedsMany* con el modelo *Data*. Se eligió este diseño ya que en *MongoDB* se recomienda embeber en las colecciones que más se consultarán, en este caso, el *RecolectionData*.

3.8.2. ITERACIÓN 3: REQUERIMIENTOS FUNCIONALES

1. El usuario **Reciclador** censará mediante la **Balanza** el **Bucket** del residente, el peso del Bucket se enviará al sistema mediante una petición HTTP realizada por la balanza, la cual contendrá el *id* del **Reciclador**, el *id* del **Residente**, la fecha del registro y el peso registrado. Esta funcionalidad se hará con un endpoint `/Residents/{id}/Data` y sólo podrá ser ejecutado por el Reciclador.
2. El usuario **Reciclador** deberá contar con una interfaz que le permita ver un historial de las recolecciones que ha realizado. Deberá poder ver la cantidad de veces que ha recolectado, el total de kilos que ha pesado, y las fechas de dichos eventos.
3. El usuario **Residente** deberá contar con una interfaz que le permita ver su historial de registros de recolección, esto es: cantidad de veces que ha reciclado sus residuos orgánicos, el peso total de los residuos reciclados y un gráfico de las recolecciones realizadas.

3.8.3. ITERACIÓN 3: ENDPOINTS

API ENDPOINT	DESCRIPTION	ROL
POST <code>/Residents/{id}/Data</code>	Agrega un registro de recolección para el residente.	Recycler
GET <code>/Residents/{id}/RecolectionData</code>	Visualiza los registros de recolección para un Residente	Owner
GET <code>/Recycler/{id}/Data</code>	Trae los registros de Data, es necesario filtrar por el campo <code>recyclerId</code> para traer los que el Recycler ha efectuado.	Recycler

Tabla 7. Endpoints y permisos para la iteración 3. Fuente: Elaboración propia.

3.9. ITERACIÓN 4: HISTORIAS DE USUARIO

1. Como usuario **Administrador**, quiero ver estadísticas de la gestión de los residuos orgánicos, tales como:
 - a. Ver el total de kilos de residuos recolectados en el tiempo.
 - b. Ver los lugares en los cuales se están recolectando los residuos en un mapa.
 - c. Ver la cantidad de residentes que están usando el sistema.
 - d. Ver la cantidad de Recicladores que están gestionando los residuos.
 - e. Por cada residente debo ser capaz de ver sus estadísticas de recolección, así como por cada Reciclador también.
2. Como usuario **Administrador**, al momento de ver las estadísticas de la gestión de residuos orgánicos, quiero ser capaz de poder **filtrar** los datos por **comuna**, es decir, ver la información sólo para una comuna en particular, con el fin de poder tener métricas asociadas a las comunas.
3. Como usuario **Administrador**, al momento de ver las estadísticas de la gestión de residuos orgánicos, quiero ser capaz de poder **filtrar** los datos por **tiempo**, es decir, ver las estadísticas para los rangos de tiempo en los cuales yo defina hacerlo, con el fin de ver cómo se comportó la gestión de residuos orgánicos en un período de tiempo en particular.

3.9.1. ITERACIÓN 4: MODELO DE DATOS

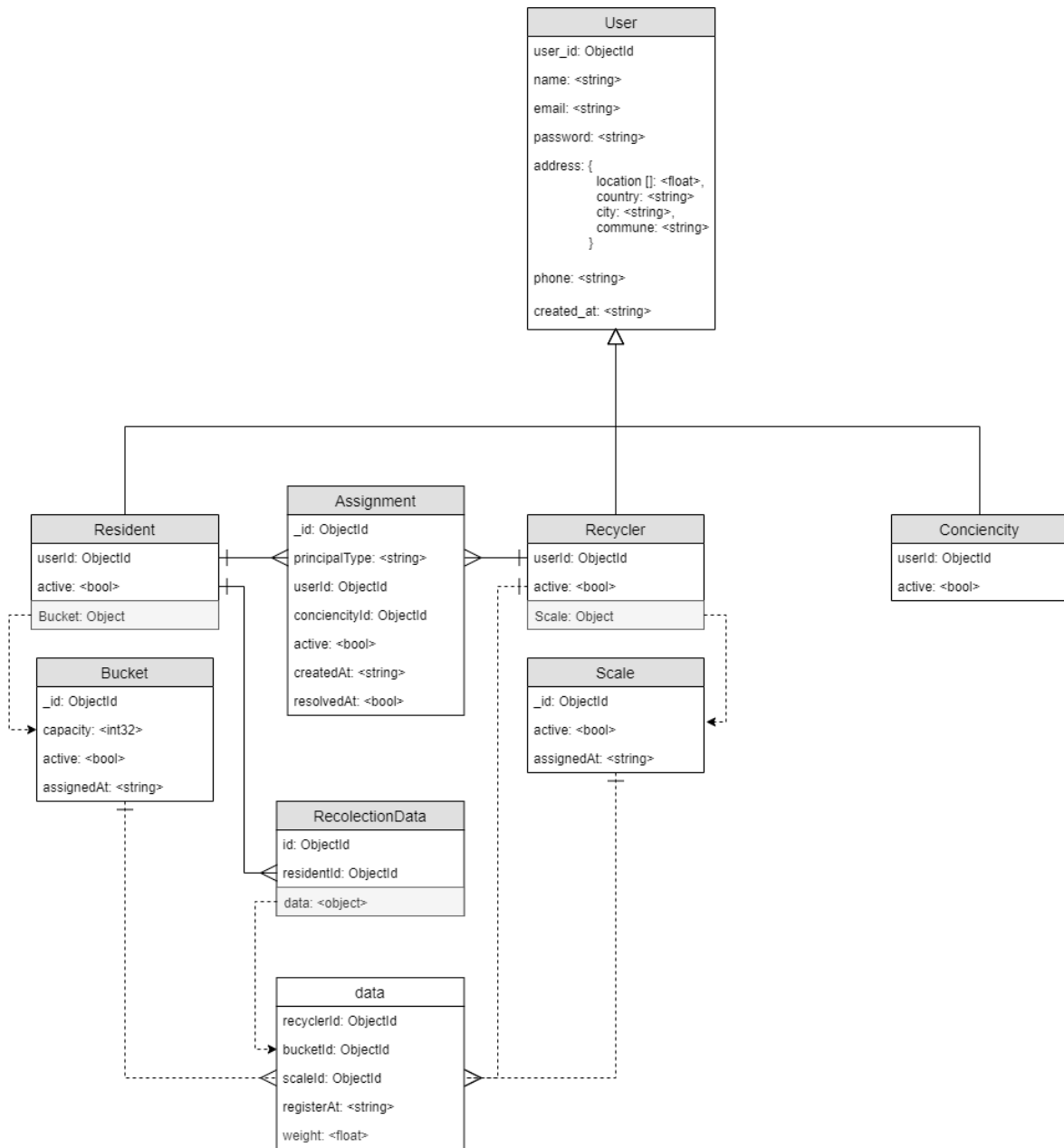


Figura 20. Modelo de datos para la iteración 4. Fuente: Elaboración propia.

3.9.2. ITERACIÓN 4: REQUERIMIENTOS FUNCIONALES

1. Para ver el total de kilos de residuos recolectados en el tiempo bastará con hacer una consulta a la colección *Data*, trayendo bajo el lenguaje de consultas de *MongoDB* la sumatoria de todos los pesos que estén registrados en dicha colección. Esta información podrá ser filtrada por tiempo, usando el mismo lenguaje de consultas.

2. Para ver los lugares en los cuales se están recolectando residuos orgánicos, bastará con hacer una consulta a la colección *Residents*, trayendo del campo *address* el atributo *location*, y posicionándolo en un mapa como el que ofrece Google Maps, estos registros deben ser filtrados por el campo *active*, de esta forma se obtendrán los Residentes que estén activos en el sistema, es decir, gestionando sus residuos orgánicos.
3. Para ver la cantidad de recicladores que están reciclando en el sistema, basta con hacer una petición a la colección *Recyclers*, filtrando por el campo *active*. Para obtener los Residentes, es análogo a lo anterior, pero haciendo las consultas sobre la colección *Residents*.
4. Para ver el detalle de las recolecciones para un Residente en particular, el administrador tendrá permisos para acceder a la información de los datos de la colección *RecolectionData* asociados al Residente que se quiere consultar. Lo mismo se hace para el caso de los historiales de recolección para el Reciclador.

3.9.3. ITERACIÓN 4: ENDPOINTS

API ENDPOINT	DESCRIPTION	ROL
GET /Data/totalWaste	Retorna el total de kilos que se han gestionado en el sistema.	Administrador
GET /Residents	Obtiene todos los residentes del sistema, filtrando por los residentes que estén activos.	Administrador
GET /Residents/{id}/RecolectionData	Visualiza los registros de recolección para un Residente.	Administrador
GET /Recyclers/{id}/Data	Visualiza los registros de recolección para un Reciclador.	Administrador

Tabla 8. Endpoints y permisos para la iteración 4. Fuente: Elaboración propia.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1. MÉTODO DE VALIDACIÓN

Para validar la solución diseñada e implementada, se realizaron *testing* para cada historia de usuario, esto es, una validación técnica de que cada endpoint cumple con su respectiva funcionalidad. Para llevar a cabo estas pruebas, se utilizaron esencialmente dos herramientas: *POSTMAN* y *Javascript*.

Para comenzar con el proceso de testing, fue necesario exportar la definición de la *API Loopback*, esto se hace a través de un comando y genera un archivo de extensión *.yaml* con la definición de todos los modelos y endpoints creados. Este archivo se importa en el programa *POSTMAN*, y luego se visualiza la interfaz que se muestra en la figura 21.

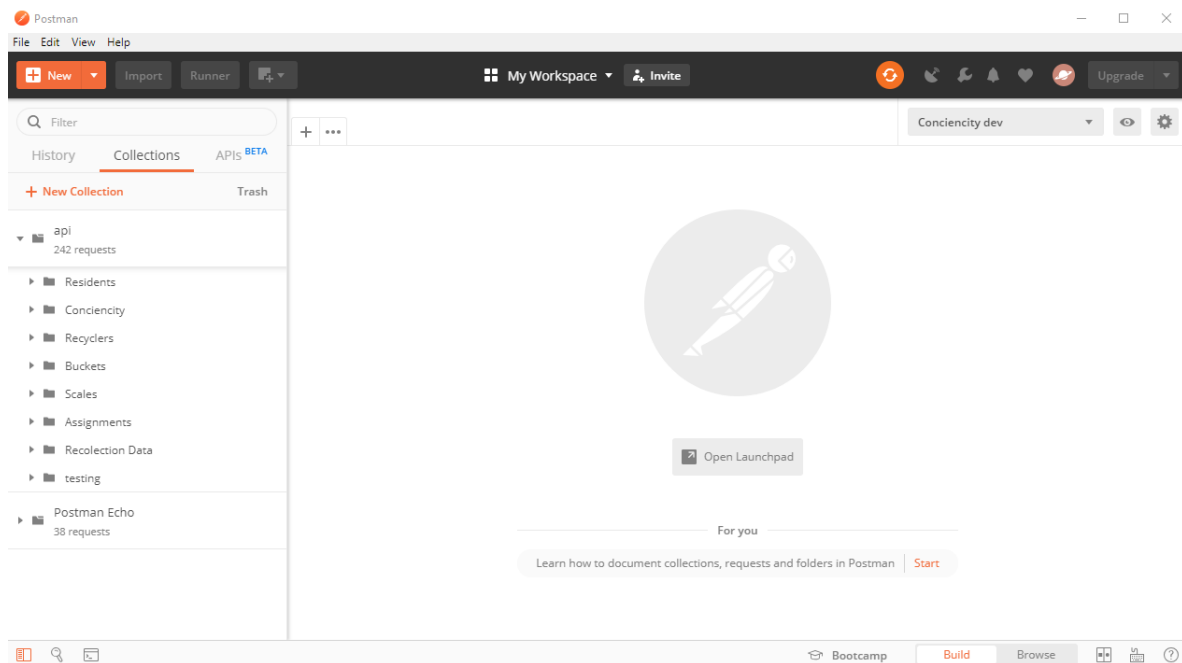


Figura 21. Vista principal de Postman. Fuente: Elaboración propia.

En la parte superior izquierda se visualiza un panel donde se encuentran todos los modelos de la *API*, los cuales contienen todos los métodos creados para cada modelo. Es importante visualizar la carpeta “testing”, que es en la cual se escriben los *test* para cada historia de usuario. En la parte superior derecha se visualiza un *select box* que contiene el entorno *Conciency dev*, el cual se encarga de guardar variables que serán usadas solo por dicho entorno, tales como la *url* base y los *tokens* de autenticación.

Se hará el testing local en una primera instancia.

4.2. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 1

4.2.1. HISTORIA DE USUARIO 1

Como usuario Residente, quiero registrarme en el Sistema ingresando mis datos personales para comenzar a gestionar mis residuos orgánicos.

Para esta historia de usuario, el test que valida la funcionalidad de crear un Residente en el sistema es chequear la respuesta que provee la API, en particular su *statusCode*, el cual debe corresponder al valor 200, que asegura que se ha creado correctamente dicho usuario (ver figura 22).

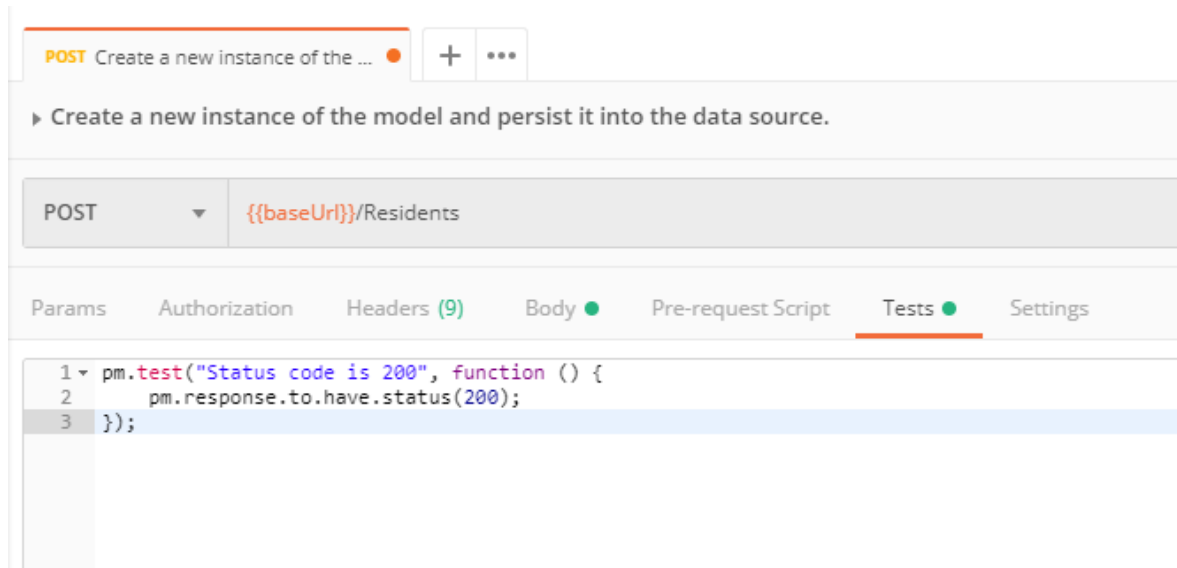


Figura 22. Test asociado a la U.S. 1 de la Iteración 1. Fuente: Elaboración propia.

Una vez enviada la petición POST a /Residents con los parámetros indicados (nombre, email y password), se puede corroborar que funciona sin problemas (Ver figura 23).

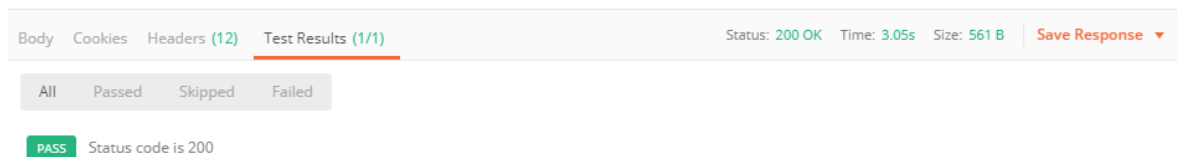


Figura 23. Respuesta para el test de la U.S.1. Fuente: Elaboración Propia.

4.2.2. HISTORIA DE USUARIO 2

Como usuario Reciclador, quiero poder registrarme en el Sistema ingresando mis datos personales para comenzar a gestionar los residuos orgánicos de los residentes.

Para esta historia de usuario, el procedimiento de validación es igual al de la historia de usuario anterior. Lo único que cambia es la URL a la cual se le hace el *POST*, la que debiera ser *POST /Recyclers* con los mismos parámetros (ver figura 24).

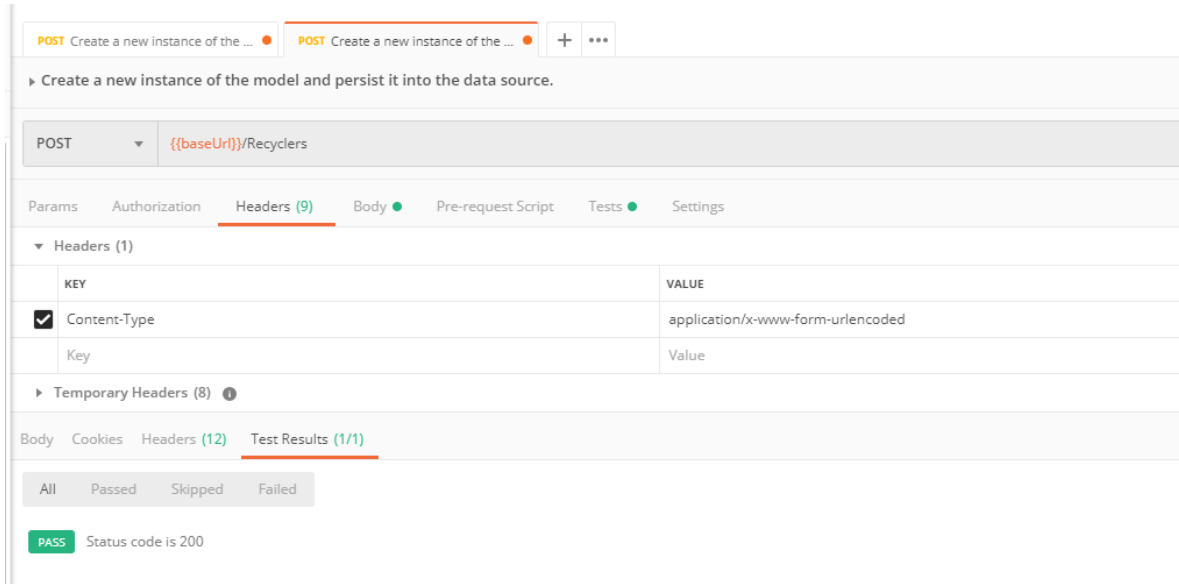


Figura 24. Respuesta para el test de la U.S.2. Fuente: Elaboración propia.

4.2.3. HISTORIA DE USUARIO 3

Como usuario Residente y Reciclador, quiero visualizar mis datos del sistema tales como mi nombre, mi correo, mi nombre de usuario, dirección, etc.

Para esta historia de usuario, lo que se debe hacer es comprobar la funcionalidad del endpoint `GET /Residents/{id}`, el cual retornará los datos del Residente con `id` igual al que se le solicita (que será el del Residente que esté logueado en el sistema). Para esto es necesario previamente estar logueado.

Para esto se debe crear un flujo en POSTMAN, el flujo consta de lo siguiente:

1. Realizar un `POST` a `/Resident/login` con los parámetros `username` o `email` y `password`.
2. Si las credenciales son válidas, se crean las variables `residentId` y `residentToken`, y se guardan en las variables globales de POSTMAN para el entorno `Conciencity-dev`.
3. Usar esas variables para realizar la petición a `/Residents/{id}` y obtener la información requerida.

Si recordamos del capítulo anterior, la información para `GET /Residents/{id}` está protegida y sólo podrá ser accedida al usuario `"Owner"` de dicha información.

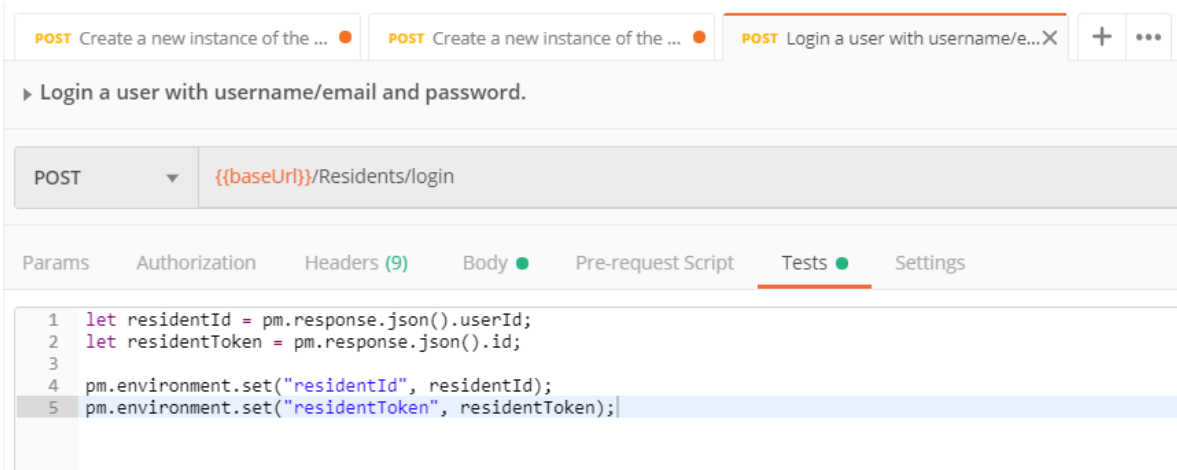


Figura 25. Código para almacenar el residentId y residentToken al momento de hacer el login. Fuente: Elaboración propia.

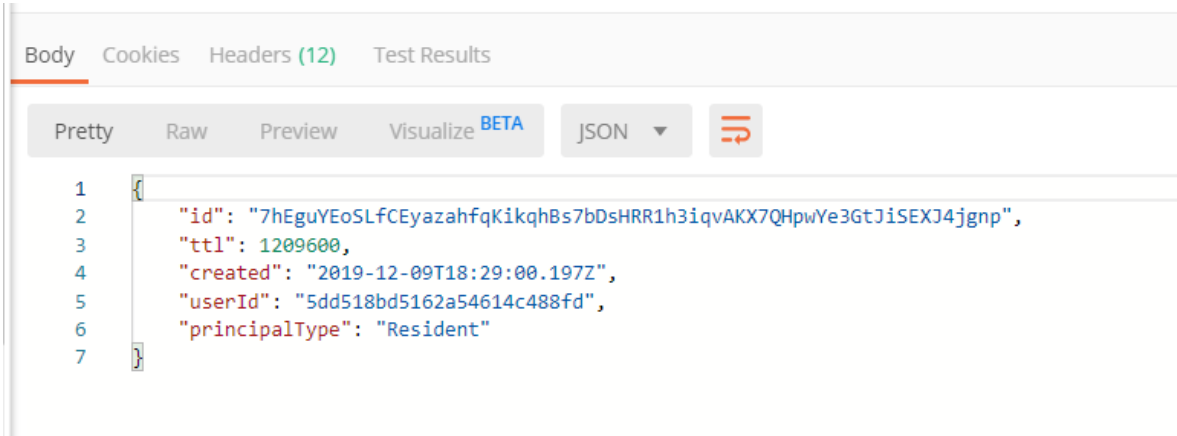


Figura 26. Respuesta de la API para el login de Resident. Fuente: Elaboración propia.

Luego de dicha respuesta, y a través del código de la figura 26, se almacenan las variables en el entorno efectivamente.

Conciencity dev			Edit
VARIABLE	INITIAL VALUE	CURRENT VALUE	
baseUrl	http://localhost:3000/api	http://localhost:3000/api	
residentId	5dd518bd5162a54614c488fd	5dd518bd5162a54614c488fd	
residentToken	7hEguYEoSLfCEyazahfqKikqhBs7bDsHRR1h3iqvAKX7QHpwYe3GtJiSEXJ4jgnp	7hEguYEoSLfCEyazahfqKikqhBs7bDsHRR1h3iqvAKX7QHpwYe3GtJiSEXJ4jgnp	

Figura 27. Variables del entorno Conciencity dev. Fuente: Elaboración propia.

Finalmente, para obtener la información requerida, se realiza la petición al endpoint GET `/Residents/{id}` usando el `accessToken` y el `id` del Resident.

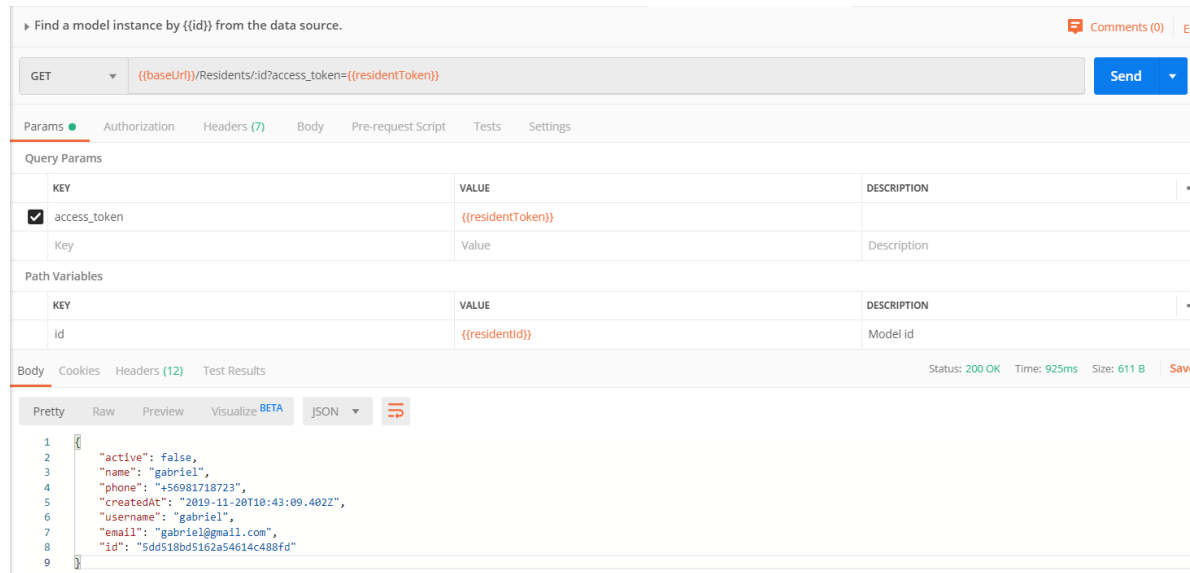


Figura 28. Respuesta para la historia de usuario 3.

Y en la respuesta se obtiene un JSON con toda la información del usuario registrada en el sistema.

4.2.4. HISTORIA DE USUARIO 4

Como usuario Residente y Reciclador, quiero editar mis datos conforme lo necesite, para poder contar con un dinamismo que me permita, por ejemplo, cambiar mi ubicación del sistema y que esto no interfiera en la funcionalidad de la gestión.

Como se menciona en los requerimientos funcionales para esta iteración, un usuario sólo podrá modificar sus propios datos (rol Owner del sistema), por lo tanto, al igual que en la historia de usuario anterior, necesitaremos que el *id* del usuario que esté editando los datos sea el mismo *id* al que pertenecen dichos datos, además de proveer el `accessToken` para una correcta validación de autenticidad.

Una vez que se obtienen esos valores, la respuesta corresponde a lo que se visualiza en la figura 29.

The screenshot shows a REST client interface with the following details:

- Method:** PATCH
- URL:** `{{(baseUrl)}/Residents/{id}?access_token={{(residentToken)}}`
- Content-Type:** x-www-form-urlencoded (selected)
- Request Body (JSON):**

```

1 {
2   "active": false,
3   "name": "Gabriel Salvador",
4   "phone": "+56981718723",
5   "createdAt": "2019-11-20T10:43:09.402Z",
6   "username": "gabriel",
7   "email": "gabriel@gmail.com",
8   "id": "5dd518bd5162a54614c488fd"
9 }

```
- Response:** Status: 200 OK, Time: 2.63s, Size: 620 B

Figura 29. Respuesta para PATCH /Residents/{id}. Fuente: Elaboración propia.

Donde lo que se envía para ser modificado es el *name* del usuario, el cual se cambia satisfactoriamente luego de validar el rol **Owner** y la correcta autenticación.

4.2.5. HISTORIA DE USUARIO 5

Como usuario Residente, quiero eliminar mi cuenta del sistema, en caso de que no quiera seguir participando de la gestión de residuos orgánicos.

Para esta historia de usuario, como se definió anteriormente, lo que ocurre es un borrado lógico del sistema, esto se hace con el fin de darle la opción a un usuario de que pueda volver al sistema, y de esta forma *recuperar* los datos que él tenía guardados durante su participación.

Para esto se creó un endpoint propio llamado DELETE /Residents/{id}/CustomDelete, el cual requiere que pueda ser ejecutado sólo por el rol **Owner**.

Resident.custom Delete

DELETE `{{(baseUrl)}/Residents/id/customDelete?access_token={{(residentToken)}}` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	<code>{{(residentToken)}}</code>	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	<code>{{(residentid)}}</code>	[object Object]

Body Cookies Headers (10) Test Results (1/1) Status: 204 No Content Time: 1571ms Size: 380 B Save

All Passed Skipped Failed

PASS Status code is 204

Figura 30. Respuesta para el método Resident Custom Delete. Fuente: Elaboración propia.

El código de estado para esta petición/respuesta es 204, lo cual asegura de que el recurso fue exitosamente modificado.

4.2.6. HISTORIA DE USUARIO 6

Como usuario Reciclador quiero eliminar mi cuenta del sistema, en caso de que no quiera o no pueda continuar realizando las labores de Reciclador.

El procedimiento es análogo al anterior.

Recycler.custom Delete

DELETE `{{(baseUrl)}/Recyclers/id/customDelete?access_token={{(recyclerToken)}}` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	<code>{{(recyclerToken)}}</code>	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	<code>{{(recyclerid)}}</code>	[object Object]

Body Cookies Headers (10) Test Results (1/1) Status: 204 No Content Time: 1719ms Size: 380 B Save

All Passed Skipped Failed

PASS Status code is 204

Figura 31. Respuesta para el método DELETE /Recyclers/{id}/CustomDelete. Fuente: Elaboración propia.

Con lo anterior, queda validado que los métodos de la iteración 1 cumplen con su funcionalidad.

4.3. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 2

4.3.1. HISTORIA DE USUARIO 1

Como usuario Conciency, quiero crear Bucket en el sistema y asignarlo a un Residente, para que el Residente tenga los implementos necesarios para comenzar a gestionar sus residuos orgánicos.

Como se menciona en el capítulo anterior, Conciency será el encargado de crear en el sistema los objetos *Bucket* y *Scale*. Para esto, es necesario validar la funcionalidad de los siguientes endpoints:

- POST /Buckets

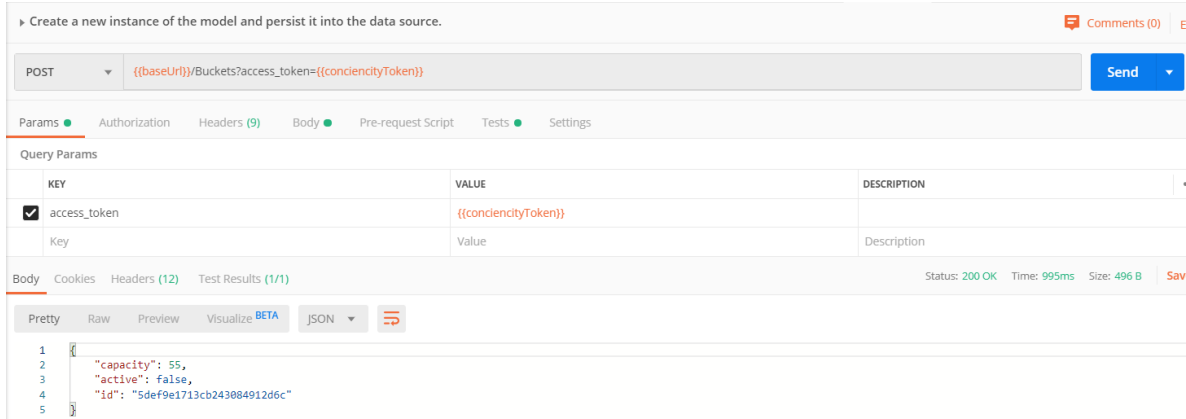


Figura 32. Respuesta para el método POST /Buckets. Fuente: Elaboración propia.

Este método requiere que el usuario que lo ejecute sea *Conciency*, por lo cual se le pasa el `accessToken` asociado a este usuario.

La prueba valida que el código de estado sea 200.

► Create a new instance of the model and persist it into the data source.

POST `{{baseUrl}}/Buckets?access_token={{conciencyToken}}`

Params ● Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

Query Params

	KEY	VALUE
☒	access_token	<code>{{conciencyToken}}</code>
	Key	Value

Body Cookies Headers (12) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 200

Figura 33. Respuesta del test POST /Buckets. Fuente: Elaboración propia.

4.3.2. HISTORIA DE USUARIO 2

Como usuario Conciency, quiero crear Scales en el sistema, y asignarlo a un Reciclador, para que el Reciclador tenga los implementos necesarios y pueda comenzar a recolectar los residuos orgánicos de los Residentes.

- POST /Scales

Análogo al anterior, cumple con los mismos prerequisites y las respuestas son similares:

► Create a new instance of the model and persist it into the data source. Comments (0) E

POST `{{baseUrl}}/Scales?access_token={{conciencyToken}}` Send

Params ● Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

Query Params

	KEY	VALUE	DESCRIPTION
☒	access_token	<code>{{conciencyToken}}</code>	
	Key	Value	Description

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK Time: 999ms Size: 482 B Save

Pretty Raw Preview Visualize BETA JSON

```

1 {
2   "active": false,
3   "id": "5def9ed213cb243884912d6e"
4 }

```

Figura 34. Respuesta para el método POST /Scales. Fuente: Elaboración propia.

► Create a new instance of the model and persist it into the data source.

POST `{{baseUrl}}/Scales?access_token={{conciencyToken}}`

Params ☒ Authorization Headers (9) Body ☒ Pre-request Script Tests ☒ Settings

Query Params

	KEY	VALUE
☒	access_token	{{conciencyToken}}
	Key	Value

Body Cookies Headers (12) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 200

Figura 35. Respuesta para el test POST /Scales. Fuente: Elaboración propia.

4.3.3. HISTORIA DE USUARIO 3

Como usuario Residente, quiero activar mi perfil para poder gestionar los residuos orgánicos de mi hogar.

Para esta funcionalidad, se debe probar el endpoint POST /Residents/{id}/Assignments, la cual debe crear un registro de solicitud de un Bucket para el Residente. Se valida también que el que ejecute el método tenga el rol de **Owner**.

- POST /Residents/{id}/Assignments

► Crea una nueva instancia en assignments de este modelo.

POST `{{baseUrl}}/Residents/:id/assignments?access_token={{residentToken}}` Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	{{residentToken}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{residentId}}	Resident id

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK Time: 1167ms Size: 584 B Save

Pretty Raw Preview Visualize BETA JSON ≡

```

1 {
2   "active": true,
3   "createdAt": "2019-12-10T13:40:03.083Z",
4   "id": "5defa03313cb243084912d71",
5   "userId": "5dd518bd5162a54614c488fd",
6   "principalType": "Resident"
7 }

```

Figura 36. Respuesta para el método POST /Residents/{id}/Assignments. Fuente: Elaboración propia.

► Crea una nueva instancia en assignments de este modelo.

POST `{{baseUrl}}/Residents/:id/assignments?access_token={{residentToken}}`

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE
☒ access_token	{{residentToken}}
Key	Value

Path Variables

KEY	VALUE
id	{{residentId}}

Body Cookies Headers (12) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 200

Figura 37. Test para el endpoint POST /Residents/{id}/Assignments. Fuente: Elaboración propia.

Hasta este momento, el usuario Residente ha registrado una solicitud de asignación de Bucket para que se active su cuenta una vez que se asigne efectivamente dicho elemento al usuario. Por lo tanto, ahora hay que validar que se resuelve efectivamente dicha asignación. Esto se hace a través del endpoint PUT /Residents/{id}/Assignments/{fk}, el cual sólo puede ser ejecutado por el usuario administrador Conciency.

- **PUT /Residents/{id}/Assignments/{fk}**

Actualizar un elemento relacionado por id para assignments.

PUT `{{(baseUrl)}/Residents/{id}/assignments/{fk}?access_token={{(conciencyToken)}}` Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	<code>{{(conciencyToken)}}</code>	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	<code>{{(residentid)}}</code>	Resident id
fk	<code>{{(residentAssignmentid)}}</code>	Clave foránea para assignments

Body Cookies Headers (12) Test Results Status: 200 OK Time: 2.04s Size: 668 B Save

Pretty Raw Preview Visualize BETA JSON

```

1
2  "conciencyId": "5dc2b59d5569d61848776f6d",
3  "active": false,
4  "createdAt": "2019-12-10T14:03:39.910Z",
5  "resolvedAt": "2019-12-10T14:05:04.000Z",
6  "id": "5defa5bb13cb243084912d74",
7  "userId": "5dd510bd5162a54614c488fd",
8  "principalType": "Resident"
9

```

Figura 38. Asignación de un Bucket para el Residente por parte del Administrador. Fuente: Elaboración propia.

4.3.4. HISTORIA DE USUARIO 4

Como usuario Reciclador, quiero activar mi perfil para poder comenzar a recolectar y reciclar los residuos orgánicos de los residentes.

Análogo a la historia de usuario anterior, el procedimiento es el mismo sólo que la diferencia es que el usuario cambia por el Recycler.

- **POST /Recyclers/{id}/Assignments**

► Crea una nueva instancia en assignments de este modelo. Comments (0) | E

POST ▼ `{{baseUrl}}/Recyclers/{id}/assignments?access_token={{recyclerToken}}` Send ▼

Params ● Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	<code>{{recyclerToken}}</code>	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	<code>{{recyclerId}}</code>	Recycler id

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK Time: 1160ms Size: 584 B Savi

Pretty Raw Preview Visualize BETA JSON ▼ ≡

```

1 {
2   "active": true,
3   "createdAt": "2019-12-10T13:44:58.957Z",
4   "id": "5defa15a13cb243084912d73",
5   "userId": "5dee8dd3ed01d087508279b0",
6   "principalType": "Recycler"
7 }
```

Figura 39. Respuesta para el método POST /Recyclers/{id}/Assignments. Fuente: Elaboración propia.

► Crea una nueva instancia en assignments de este modelo.

POST ▼ `{{baseUrl}}/Recyclers/{id}/assignments?access_token={{recyclerToken}}`

Params ● Authorization Headers (9) Body ● Pre-request Script Tests ● Settings

Query Params

KEY	VALUE
☒ access_token	<code>{{recyclerToken}}</code>
Key	Value

Path Variables

KEY	VALUE
id	<code>{{recyclerId}}</code>

Body Cookies Headers (12) Test Results (1/1)

All Passed Skipped Failed

PASS Status code is 200

Figura 40. Respuesta para el test POST /Recyclers/{id}/Assignments. Fuente: Elaboración propia.

Y finalmente, para asignar efectivamente un Scale por parte del usuario Administrador (Conciency) a un Reciclador, se ejecuta el endpoint:

- PUT /Recyclers/{id}/Assignments/{fk}

► Actualizar un elemento relacionado por id para assignments. Comments (0) | E

PUT `{{(baseUrl)}/Recyclers/id/assignments/fk?access_token={{(conciencyToken)}}` Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	<code>{{(conciencyToken)}}</code>	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	<code>{{(recyclerid)}}</code>	Recycler id
fk	<code>{{(recyclerAssignmentid)}}</code>	Clave foránea para assignments

Body Cookies Headers (12) Test Results Status: 200 OK Time: 1666ms Size: 668 B Save

Pretty Raw Preview Visualize BETA JSON ≡

```

1 {
2   "conciencyId": "5dc2b59d5569d61848776f6d",
3   "active": false,
4   "createdAt": "2019-12-10T14:18:43.379Z",
5   "resolvedAt": "2019-12-10T14:19:01.000Z",
6   "id": "5defa94313cb243084912d7a",
7   "userId": "5dee8dd3ed01d057500279b0",
8   "principalType": "Recycler"
9 }

```

Figura 41. Respuesta para PUT /Recyclers/{id}/Assignments/{fk}. Fuente: Elaboración propia.

De esta forma se valida la asignación de recursos y activación de cuentas tanto para el usuario Residente como para el usuario Reciclador.

4.4. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 3

4.4.1. HISTORIA DE USUARIO 1

Como usuario Reciclador, quiero poder censar el peso de un Bucket de un Residente mediante la balanza, con el fin de poder registrar el detalle de la recolección del residuo orgánico para el Residente, así como el peso y la hora del registro.

Para esta funcionalidad, debemos usar el endpoint creado POST /Residents/{id}/Data, el cual sólo puede ser ejecutado por el usuario con rol Recycler. El endpoint se encarga de persistir un registro de recolección asociado al residente y al reciclador.

El flujo para esta historia de usuario sería:

1. Iniciar sesión como Recycler.
2. Censar el peso del residuo del Resident y enviar la información al endpoint POST /Residents/{id}/Data.
3. Validar que se haya escrito correctamente el registro.

Resident. post data

POST `{{baseUrl}}/Residents/{id}/data?access_token={{recyclerToken}}` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	{{recyclerToken}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{residentid}}	[object Object]

Body Cookies Headers (12) Test Results (1/1) Status: 200 OK Time: 2.42s Size: 469 B Sav

All Passed Skipped Failed

PASS Status code is 200

Figura 42. Validación del método POST /Residents/{id}/Data. Fuente: Elaboración propia.

4.4.2. HISTORIA DE USUARIO 2

Como usuario Reciclador, quiero poder ver mi historial de recolecciones realizadas, con el fin de visualizar estadísticas relacionadas a mi participación en el sistema, tales como la cantidad de recolecciones que he realizado, y la cantidad de kilos totales que he recolectado.

Gracias al modelo de datos, al estar persistida la colección Data (y también embebida dentro de ReleccionData), podemos consultar directamente los registros de recolección que ha recogido un Reciclador en particular. Esto se hace simplemente enviando una petición al método GET /Recyclers/{id}/Data

El test asociado verifica el código de respuesta, además de devolver la información requerida.

data consultas de Recycler.

GET `{{(baseUrl)}/Recyclers/{id}/data?access_token={{(recyclerToken)}}` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☒ access_token	{{(recyclerToken)}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{(recyclerId)}}	Recycler id

Body Cookies Headers (13) Test Results (2/2) Status: 200 OK Time: 1175ms Size: 3.78 KB

Pretty Raw Preview Visualize BETA JSON

```

1 {
2   {
3     "bucketId": "5def9dd813cb243084912d6a",
4     "scaleId": "5defa95113cb243084912d7b",
5     "registerAt": "2019-12-14T17:54:56.000Z",
6     "weight": 4,
7     "id": "5df521f04220b2418c3a3bf7",
8     "recyclerId": "5dee8dd3ed01d057508279b0"
9   },
10  {
11    "bucketId": "5def9dd813cb243084912d6a",
12    "scaleId": "5defa95113cb243084912d7b",
13    "registerAt": "2019-12-14T17:56:52.000Z",
14    "weight": 5,
15    "id": "5df522654f848c189448ec55",
16    "recyclerId": "5dee8dd3ed01d057508279b0"
17  },
18 }

```

Figura 43. Respuesta para GET /Recyclers/{id}/Data. Fuente: Elaboración propia.

De la respuesta se puede obtener información como el número de recolecciones que ha hecho un Recycler y, además, mediante el uso de filtros podemos saber la data que se ha recolectado en cierto intervalo de fechas entre otras informaciones interesantes.

4.4.3. HISTORIA DE USUARIO 3

Como usuario Residente, quiero poder ver mi historial de recolección, con el fin de visualizar estadísticas relacionadas a la gestión de mis residuos, tales como la cantidad de veces que me han ido a recolectar los residuos y la cantidad de residuos que he gestionado en el sistema.

Esta historia de usuario es más directa, lo que se debe hacer aquí es un GET /Residents/{id}/RecolectionData

Esto nos retorna un documento con la *data* embebida (ver figura 44).

recolectionData consultas de Resident.

GET `{{(baseUrl)}/Residents/{id}/recolectionData?access_token={{(residentToken)}}` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
filter	<JSON>	
access_token	{{(residentToken)}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{(residentid)}}	Resident id

Body Cookies Headers (13) Test Results (1/1) Status: 200 OK Time: 1211ms Size: 3.4 KB Sav

Pretty Raw Preview Visualize BETA JSON

```

1 {
2   "id": "5de5666132c89d1a849022d8",
3   "residentId": "5dd518bd5162a54614c488fd",
4   "data": [
5     {
6       "bucketId": "5dd51e968fdebd366044546e",
7       "scaleId": "5de41172fa7f763058870396",
8       "registerAt": "2019-12-03T12:09:09.000Z",
9       "weight": 1,
10      "id": "e02f77f3c244c409b187f046",
11      "recolectionDataId": null,
12      "recyclerId": "5dc6b11786eada3c607e2092"
13    },
14    {
15      "bucketId": "5dd51e968fdebd366044546e"
16    }
17  ]
18 }

```

Figura 44. Respuesta para el endpoint GET /Residents/{id}/RecolectionData. Fuente: Elaboración propia.

De la respuesta, podemos inferir información con respecto a su historial de recolección, tales como la cantidad de veces que ha reciclado sus residuos, qué reciclador ha sido el que ha ido a recolectarlos, y las fechas de recolección.

4.5. VALIDACIÓN DE FUNCIONALIDADES PARA LA ITERACIÓN 4

4.5.1. HISTORIA DE USUARIO 1

Como usuario **Administrador**, quiero ver estadísticas de la gestión de los residuos orgánicos, tales como:

- Ver el total de kilos de residuos recolectados en el tiempo.
- Ver los lugares en los cuales se están recolectando los residuos en un mapa.
- Ver la cantidad de residentes que están usando el sistema.
- Ver la cantidad de recicladores que están gestionando los residuos.
- Por cada residente debo ser capaz de ver sus estadísticas de recolección, así como por cada Reciclador también.

Para el primer requerimiento se tiene que hacer una petición a la colección **Data**, consultando todos los registros de la colección y haciendo una sumatoria del campo *weight*, esto permite obtener el total de kilos de residuos recolectados en el tiempo. El endpoint habilitado que hace dicho trabajo es GET /Data/totalWaste y puede ser ejecutado sólo por el Administrador.

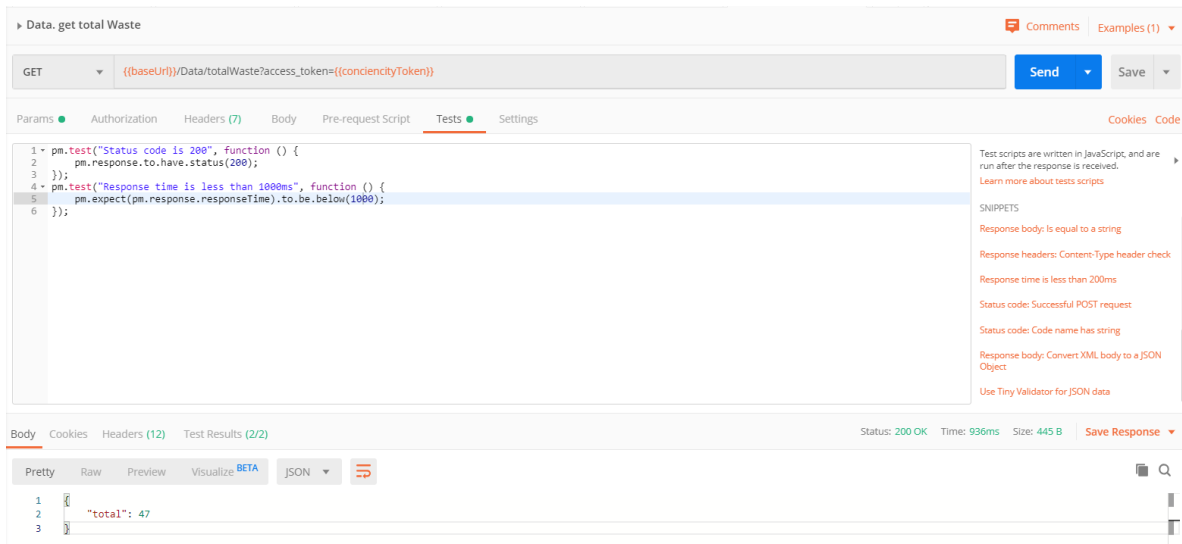


Figura 45. Tests y respuesta para el endpoint GET /Data/totalWaste. Fuente: Elaboración propia.

Para la siguiente petición, ver las ubicaciones geográficas en un mapa, hay que hacer una petición a la colección Residents y traer el campo address de cada Resident, luego con dicho campo y una vez que se tenga el atributo *“location”*, se podrá poner en un mapa la información requerida.

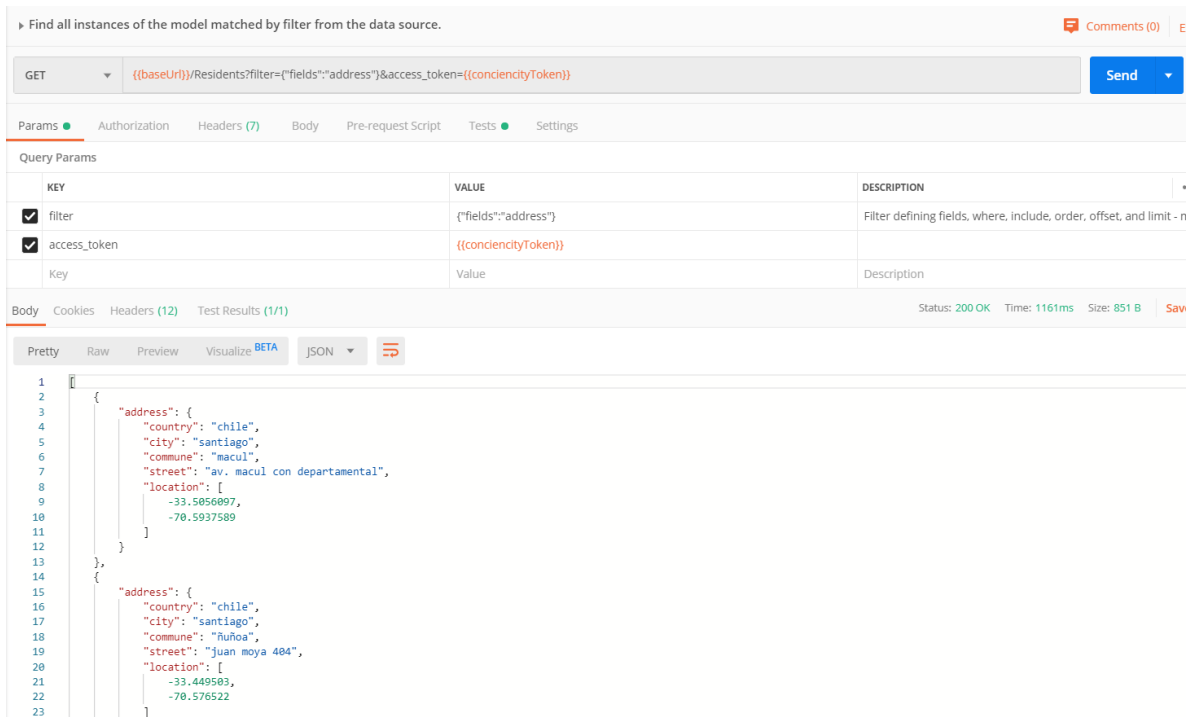


Figura 46. Respuesta para GET /Residents. Fuente: Elaboración propia.

Es importante el filtro que se le realiza a esta consulta, donde en este caso lo que se obtiene es el objeto entero *address*, y la respuesta es un arreglo de estos objetos.

Para ver la cantidad de residentes que están utilizando el sistema, basta con llamar al endpoint GET /Residents/count, es importante acá el campo *where* que se aplica a la consulta, este campo permite filtrar sólo los residentes que están activos.

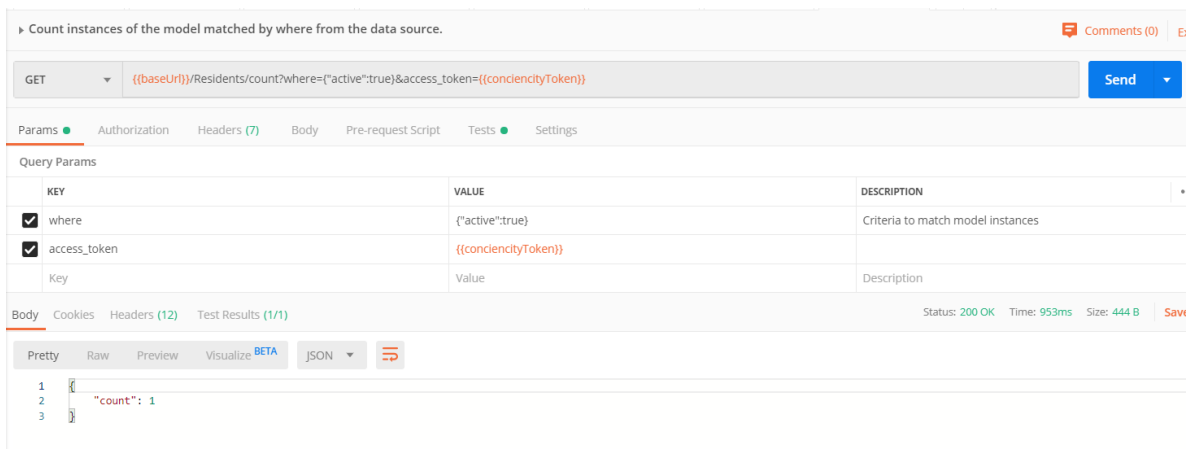


Figura 47. Respuesta para GET /Residents/count. Fuente: Elaboración propia.

Para ver la cantidad de Recicladores que están utilizando el sistema, el endpoint habilitado es el mismo que para el caso anterior, sólo que el modelo a consultar es para el Recycler.

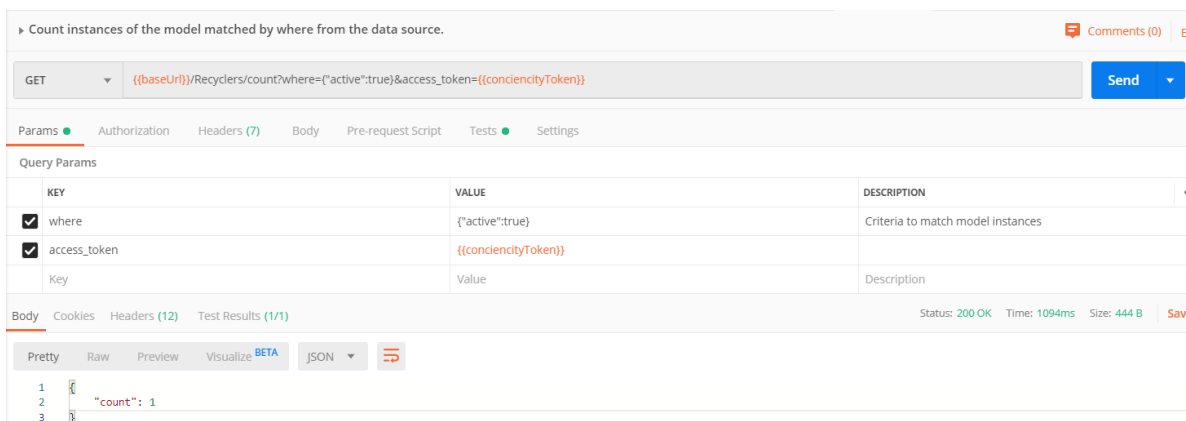


Figura 48. Respuesta para GET /Recyclers/count. Fuente: Elaboración propia.

Finalmente, para poder ver el detalle de recolección para cada residente y reciclador, se debe hacer una petición al endpoint GET /Residents/{id}/RecolectionData

recolectionData consultas de Resident.

GET `{{(baseUrl)}/Residents/id/recolectionData?access_token={{(conciencyToken)}}` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☐ filter	<JSON>	
☒ access_token	{{(conciencyToken)}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{(residentId)}}	Resident id

Body Cookies Headers (13) Test Results (1/1) Status: 200 OK Time: 1150ms Size: 3.4 KB Save

Pretty Raw Preview Visualize BETA JSON ≡

```

1 {
2   {
3     "id": "5de5666132c89d1a848022d8",
4     "residentId": "5dd518bd5162a54614c488fd",
5     "_data": [
6       {
7         "bucketId": "5dd51e968fdebd366044546e",
8         "scaleId": "5de41172fa7f763058870396",
9         "registerAt": "2019-12-03T12:09:09.000Z",
10        "weight": 1,
11        "id": "e02f77f3c244c409b187f046",
12        "recolectionDataId": null,
13        "recyclerId": "5dc0b11786eada3c007e2092"
14      },
15      {
16        "bucketId": "5dd51e968fdebd366044546e",
17        "scaleId": "5de41172fa7f763058870396",

```

Figura 49. Respuesta para GET /Residents/{id}/RecolectionData. Fuente: Elaboración propia.

Y mediante esta respuesta, se pueden obtener las estadísticas solicitadas en la historia de usuario.

Para obtener las estadísticas del Recycler, se debe hacer una petición al endpoint GET /Recyclers/{id}/Data, dicho endpoint devolverá los registros de recolección asociados al Recycler, mediante los cuales se puede obtener las estadísticas solicitadas.

data consultas de Recycler.

GET `{{(baseUrl)}/Recyclers/{id}/data?access_token={{(conciencyToken)}}` Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
☐ filter	<JSON>	
☒ access_token	{{(conciencyToken)}}	
Key	Value	Description

Path Variables

KEY	VALUE	DESCRIPTION
id	{{(recyclerId)}}	Recycler id

Body Cookies Headers (13) Test Results (1/1) Status: 200 OK Time: 1274ms Size: 3.78 KB Sav

Pretty Raw Preview Visualize BETA JSON

```

1  {
2    "bucketId": "5def9dd813cb243084912d6a",
3    "scaleId": "5defa95113cb243084912d7b",
4    "registerAt": "2019-12-14T17:54:56.000Z",
5    "weight": 4,
6    "id": "5df521f04220b2418c3a3bf7",
7    "recyclerId": "5dee8dd3ed01d057508279b0"
8  },
9  {
10   "bucketId": "5def9dd813cb243084912d6a",
11   "scaleId": "5defa95113cb243084912d7b",
12   "registerAt": "2019-12-14T17:56:52.000Z",
13   "weight": 5,
14   "id": "5df522654f848c189448ec55",
15   "recyclerId": "5dee8dd3ed01d057508279b0"
16 },
17 }
```

Figura 50. Respuesta para la petición GET /Recyclers/{id}/Data. Fuente: Elaboración propia.

4.6. VALIDACIÓN DE SEGURIDAD BASADA EN ROLES

Otro punto importante de validar es la seguridad de los endpoints. Como se mencionó en capítulos anteriores, *Loopback* permite tener un control sobre los Roles de los usuarios y a qué métodos pueden acceder dichos roles o no. La validación siguiente se hará para validar una historia de usuario a modo de ejemplo, pero esa validación cumple para todas las historias restantes.

Lo que se hace es intentar acceder al método GET /Recyclers/{id}/Data, usando el rol Resident, pero dicho método está protegido y sólo puede ser accedido por el tipo de usuario Conciency o Recycler. Por lo tanto, muestra un error 401, el test falla y no se muestra la información requerida.

Página 84 de 91

CAPÍTULO 5: CONCLUSIONES

5.1. CONCLUSIONES GENERALES

En esta memoria se estudió la gestión de residuos en Chile, tema bastante extenso en el que se detectan diversas formas de contribuir desde la Informática. Sin embargo, para poder aportar efectivamente, fue necesario comprender bien cómo funciona el sistema de gestión de residuos actualmente tanto en Chile como en el extranjero. Se logró recoger enriquecedoras experiencias de países que nos llevan años de ventaja, lo que permitió extraer ideas para diseñar un sistema de gestión de residuos que se adapte a nuestro país.

Todo lo anterior está estrictamente acompañado de una característica que como humanos debemos desarrollar a lo largo de la vida: la conciencia.

Es importante que la sociedad a través de su gobierno y/o municipios trabajen en concientizar a las personas para que todos podamos entender la gravedad de tener sistemas como el que tenemos actualmente en Chile, puesto que, si bien el sistema de gestión de residuos funciona, no es para nada eficiente, y termina provocando serios daños medioambientales.

El diseño e implementación de un Software como Servicio para la gestión integral de residuos en Chile, surge como una solución moderna para un sistema que podría hacerse de una mejor manera, apoyado en la tecnología. Usando componentes como *IoT*, se hace posible obtener datos que permitan tomar decisiones o tener una visión más concreta de cuán eficiente está siendo la gestión de los residuos en el país, entre otras informaciones interesantes.

El uso de la tecnología en estos tiempos es imprescindible, sin embargo, para un sistema que comprende muchos procesos que dependen de humanos, se hace un poco más complejo llevarlo a la operación misma.

Este prototipo puede ser utilizado para apoyar emprendimientos que se encarguen de gestionar los residuos orgánicos, ya que está hecho para cubrir todos los roles que en un sistema de este tipo puedan surgir (como el administrador, el residente y el reciclador), y cumple con ser fácilmente ampliable y escalable para otros tipos de residuos también, basta con hacer cambios en los procesos del sistema.

5.2. CONCLUSIONES ESPECÍFICAS

Para el desarrollo de este módulo del proyecto se utilizaron las tecnologías *Loopback (NodeJS)*, *MongoDB*, *Nginx*, *Docker*, *Postman* y las plataformas *DigitalOcean* y *Mongodb Atlas*, las cuales sin problemas permitieron levantar la solución diseñada, por lo que se concluye que a nivel de prototipo estas tecnologías son suficientes. Cabe mencionar que como se documentó en un entorno de desarrollo, el uso de *Docker-Compose* es suficiente, pero que, para entornos de producción, es necesario usar *Docker Swarm* [33] el cual está diseñado para ser implementado en entornos productivos.

En conclusión, las tecnologías escogidas fueron las correctas.

5.3. RECOMENDACIONES

Hay diversos problemas que podemos abordar como ingenieros en informática y como ingenieros en otras áreas también, sin embargo, es importante que se piense en equipos de desarrollo como un todo, en el cual

cada integrante posee ciertas capacidades e intereses, que debemos saber aprovechar para lograr el desarrollo efectivo de proyectos multidisciplinarios.

Para enfrentar problemas grandes y complejos, se requieren de equipos de personas con distintos perfiles, que puedan proponer soluciones innovadoras que busquen mejorar los sistemas actuales. En Chile se necesita mucho trabajar en equipos así, somos un país en vías de desarrollo donde gran parte de los sistemas que nos gobiernan dejan bastante que desear, en cuanto al diseño e implementación y su funcionamiento actual. Por lo tanto, según mi punto de vista hay una oportunidad de mejorar o derechamente reemplazar algunos sistemas por otros más modernos, donde se tenga como prioridad hacerlo de la mejor manera.

Otra recomendación importante es creer en el equipo que está abordando el problema, y comprometerse realmente con el proyecto y su desarrollo. Saber elegir a los compañeros de equipo también es algo que se da a veces de manera orgánica, sin embargo, hay que saber identificar a quienes no estén tan realmente comprometidos ya que eso afectará el desarrollo del proyecto entero.

El liderazgo y el trabajo en equipo son habilidades fundamentales que quizás no se desarrollan tanto en la universidad, pero con oportunidades de realizar este tipo de proyectos multidisciplinarios uno puede adoptar y pulir estas habilidades, que sin duda son de las más importantes que he aprendido en esta carrera y etapa de mi vida. Ahora ya en un ámbito más profesional puedo ver que todos los proyectos siguen esa misma mecánica de ser equipos multidisciplinarios en el cual cada uno aporta con su conocimiento dentro del rubro que es especialista.

La recomendación es a que se involucren en proyectos multidisciplinarios y ojalá con problemas que tengan un impacto social directo, que ayuden a tener una mejor calidad de vida para las personas y un menor impacto para el planeta.

5.4. TRABAJO FUTURO

Este proyecto puede sin duda ser reutilizado, ampliado, extendido y escalado. Además, al ser un componente de un sistema grande, se puede usar como una pieza dentro de un rompecabezas, donde junto al desarrollo e integración de las otras piezas se puede armar un sistema integral de gestión de residuos.

Si bien este proyecto multidisciplinario comprende el desarrollo de una interfaz gráfica validada por usuarios, también sería interesante proponer un desarrollo para otros tipos de dispositivos como los móviles.

El trabajo que personalmente me gustaría que se desarrollase sería el de escalar esta solución y pensarla como una solución que sirva para todos los emprendimientos que están gestionando de una manera *independiente* los residuos de los ciudadanos, están aquellos (y que son a los que apuntaríamos en primera instancia) que gestionan los residuos orgánicos y los transforman en otros tipos de energía como es el *Compost* y *Humus*, productos que se utilizan para potenciar la calidad de las tierras de cultivo, o de la tierra en general. Por lo tanto, este sistema podría escalarse a una arquitectura *Multi-Tenant* [34], donde cada cliente podrá hacer uso del sistema y tener el control sobre sus propios usuarios, de esta forma se diversifica la toma de datos del país en la gestión de residuos y se aprovecha este desarrollo para poder servir a muchos emprendimientos. Sin duda sería un desarrollo que podría tener grandes frutos.

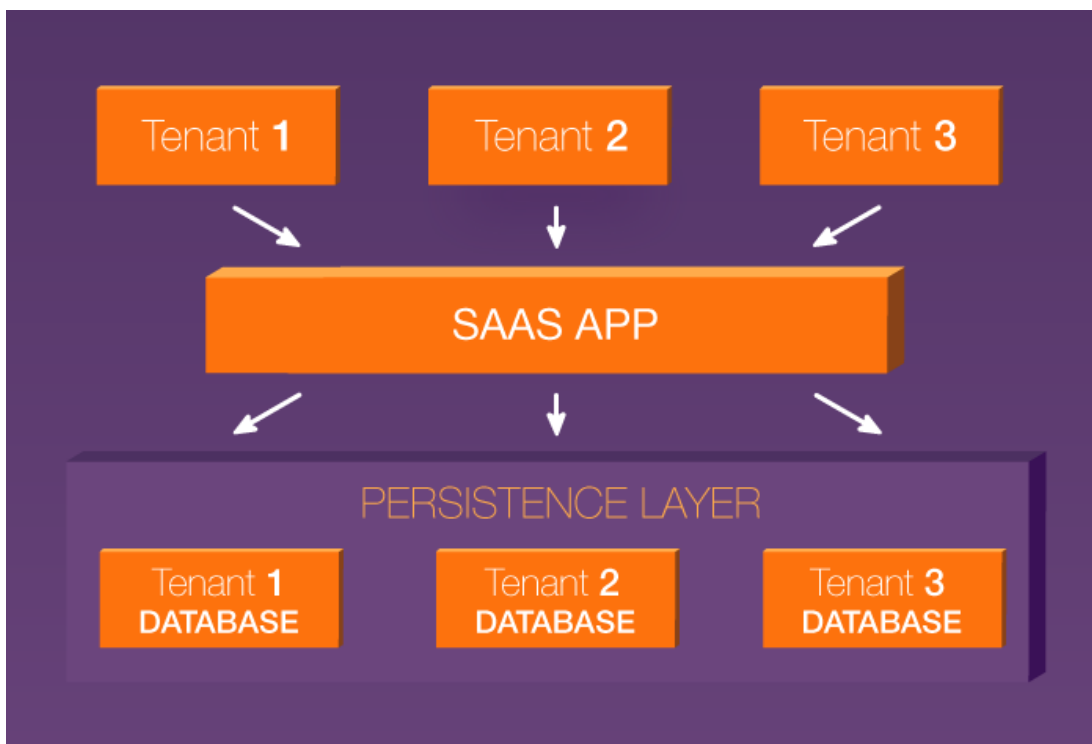


Figura 52. Arquitectura multi Tenant. Fuente: <https://adrianalonso.es/arquitectura-del-software/enfoques-de-arquitecturas-multitenant-para-aplicaciones-saas/>

5.5. PALABRAS FINALES DEL AUTOR

Durante el transcurso de este trabajo de título aprendí mucho acerca de sistemas de gran envergadura como es el caso del sistema de gestión de residuos, donde más que tomarlo como un trabajo de título, en un principio y hasta el día de hoy, lo tomé como un desafío para poder proponer soluciones a un problema con múltiples variables, que involucra diversas disciplinas y habilidades de trabajo en equipo, para lograr diseñar, desarrollar e implementar una solución lo más efectiva posible.

Si bien el sistema no pudo llegar a su fase de implementación puesto que, como se mencionó en apartados anteriores, requiere de muchos procesos y muchas personas y permisos que gestionar, quedo completamente motivado y capacitado para continuar con este proyecto a lo largo del tiempo, desarrollando e integrando los siguientes módulos como los que se mencionan en el trabajo futuro (apartado anterior). Sin duda sistemas de este estilo son estrictamente necesarios de estudiar e implementar, puesto que, en países como Chile, la tecnología si bien avanza, la implementación de sistemas modernos que utilicen o apoyen sus procesos en la tecnología, aún no logra impactar contextos que podrían considerarse *básicos*, como es el caso de la basura, entre muchos otros.

Finalmente, insto y motivo a los futuros ingenieros a unirse a este tipo de iniciativas, donde hoy más que nunca son necesarias y totalmente posibles de implementar.

BIBLIOGRAFÍA

- [1] J. C. J. R. a. S. R. A. Allamand, «Primer Reporte del Manejo de Residuos Sólidos en Chile,» Departamento de Comunicaciones Comisión Nacional de Medio Ambiente (CONAMA), Santiago, 2010.
- [2] F. Pacheco, R. Sánchez y M. Villena, «Eficiencia de los Gobiernos Locales y sus Determinantes. Un Análisis de Fronteras Estocásticas en Datos de Panel para Municipalidades Chilenas,» Dirección de presupuestos, 2013.
- [3] «Big Belly,» 2018. [En línea]. Available: <https://www.bigbelly.com/>.
- [4] «Enevo,» 2019. [En línea]. Available: <https://www.enevo.com/>.
- [5] «Ecube Labs,» 2019. [En línea]. Available: <https://www.ecubelabs.com/>.
- [6] «Compology,» 2019. [En línea]. Available: <https://compology.com/>.
- [7] «Sensoneo,» 2019. [En línea]. Available: <https://sensoneo.com/>.
- [8] D. Elorza, «socialab,» 2016. [En línea]. Available: <https://comunidad.socialab.com/challenges/InnovaLaReina/idea/61069>.
- [9] Urbanatika, 2019. [En línea]. Available: <http://urbanatika.com/#next>.
- [10] lun, «lun,» 2017. [En línea]. Available: http://www.lun.com/Pages/NewsDetail.aspx?dt=2017-05-18&NewsID=373223&BodyID=0&PaginaId=36&strNameFile=maquina_reciclaje_lun.flv&iMoveScroll=461.
- [11] Ecoven, «maquinas-de-reciclaje,» 2019. [En línea]. Available: <https://ecoven.cl/maquinas-de-reciclaje/>.
- [12] ReciclApp, «3ie,» 2018. [En línea]. Available: <http://www.ib.3ie.cl/wp-content/uploads/2018/11/Ficha-IB-Reciclapp.pdf>.
- [13] J. Dávila, «marketing4ecommerce,» 2019. [En línea]. Available: <https://marketing4ecommerce.cl/asi-funciona-chat-la-apuesta-de-google-para-sustituir-a-los-sms-y-derrotar-a-whatsapp/>.
- [14] Karubag, «karubag,» 2019. [En línea]. Available: <https://www.karubag.cl/planes>.
- [15] Mivita, «tarjetamivita,» 2019. [En línea]. Available: <http://www.tarjetamivita.cl/beneficios/karubag/>.
- [16] srcompost, «srcompost,» 2019. [En línea]. Available: <http://srcompost.cl/>.

- [17] Wikipedia, «Transferencia_de_Estado_Representacional,» Wikipedia, 28 09 2019. [En línea]. [Último acceso: 15 12 2019].
- [18] «Software como servicio,» 2017. [En línea]. Available: https://es.wikipedia.org/wiki/Software_como_servicio#targetText=Software%20como%20un%20Servicio%2C%20abreviado,v%C3%ADa%20Internet%20desde%20un%20cliente..
- [19] «Internet of things,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_of_things.
- [20] «Big data,» 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Big_data.
- [21] «NoSQL,» 2019. [En línea]. Available: <https://es.wikipedia.org/wiki/NoSQL>.
- [22] «Docker,» 2019. [En línea]. Available: [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software)).
- [23] Docker, «Docker Compose,» Docker Inc., 2019. [En línea]. Available: <https://docs.docker.com/compose/>.
- [24] «Backend y Frontend,» [En línea]. Available: <https://platzi.com/blog/que-es-frontend-y-backend/#targetText=Algunos%20de%20los%20lenguajes%20de,el%20proyecto%20que%20est%C3%A1s%20desarrollando..>
- [25] «Loopback,» 2018. [En línea]. Available: <https://loopback.io/lb3>.
- [26] «Cross Platform Software,» 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Cross-platform_software.
- [27] «MongoDB,» 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Cross-platform_software.
- [28] «Database Index,» 2019. [En línea]. Available: https://en.wikipedia.org/wiki/Database_index#targetText=An%20index%20is%20a%20copy,indexes%20on%20functions%20or%20expressions..
- [29] «what-container,» 2019. [En línea]. Available: <https://www.docker.com/resources/what-container>.
- [30] nginx, «nginx.com,» Nginx Inc., 2019. [En línea]. Available: nginx.com.
- [31] A. López, «qué es postman,» 03 Junio 2019. [En línea]. Available: <https://openwebinars.net/blog/que-es-postman/>. [Último acceso: 17 12 2019].
- [32] S. W. Ambler, «Agile Modeling,» Amblysoft Inc, 2019. [En línea]. Available: <http://agilemodeling.com/>.
- [33] Docker, «Docs Docker,» Docker Inc, 2019. [En línea]. Available: <https://docs.docker.com/engine/swarm/>.

- [34] D. Alexander, «platzi,» Platzi, 01 12 2017. [En línea]. Available: <https://platzi.com/blog/multi-tenant-que-es-y-por-que-es-importante/>. [Último acceso: 21 12 2019].
- [35] «Sensoneo,» Sensoneo, 2019. [En línea]. Available: <https://sensoneo.com/>.
- [36] Urbanatika, «urbanatika,» [En línea]. Available: <http://urbanatika.com/#next>.
- [37] Urbanatika, «urbanatika,» 2019. [En línea]. Available: <http://urbanatika.com/#next>.
- [38] mongodb, «atlas,» 2019. [En línea]. Available: <https://www.mongodb.com/cloud/atlas>.
- [39] «docker,» Docker, 2019. [En línea]. Available: <https://www.docker.com/resources/what-container>.
- [40] «docker,» Docker Inc, 2019. [En línea]. Available: <https://www.docker.com/resources/what-container>. [Último acceso: 22 October 2019].
- [41] Loopback, «Inside a Loopback application,» Strongloop, 2019. [En línea]. Available: <https://loopback.io/doc/en/lb4/Inside-LoopBack-Application.html>.