ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

DEMOSTRACIÓN DE PROCESO DEVOPS PARA EL DESARROLLO DE UNA APLICACIÓN WEB: SISTEMA DE DONACIÓN DE RECURSOS.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO ESPECIALISTA EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN

IVÁN VLADIMIR TOROMORENO SARANGO

ivan.toromoreno@epn.edu.ec

DIRECTOR: MSc. REGINA MARITZOL TENEMAZA VERA

maritzol.tenemaza@epn.edu.ec

CODIRECTOR: MSc. Vicente Adrián Egüez Sarzosa

adrian.eguez@epn.edu.ec

Quito, octubre 2021

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Iván Vladimir Toromoreno Sarango, bajo nuestra supervisión.

MSc. Maritzol Tenemaza

DIRECTOR DE PROYECTO

MSc. Adrián Egüez

CODIRECTOR DE PROYECTO

DECLARACIÓN

Yo, Iván Vladimir Toromoreno Sarango, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Iván Vladimir Toromoreno Sarango

DEDICATORIA

Mi tesis la dedico con todo mi corazón a todas las víctimas de la pandemia causada por el virus COVID-19, especialmente a mi padre quien siempre supo guiarme por el camino correcto y hasta el último de sus días demostró su amor y admiración hacia mi persona.

A mi madre quien es mi mayor ejemplo de lucha y quién siempre ha encontrado una forma de sacar a toda mi familia adelante.

A mis hermanos quienes con su amor y paciencia han sido pilares fundamentales para ser el hombre que soy hoy en día.

A Anahí, quien me ha acompañado en los momentos buenos y malos de mi carrera y ha sido un apoyo incondicional.

A mis amigos, especialmente a Jorge, César y Daniel, a quienes siempre consideraré parte de mi familia.

A mis maestros, quienes con su sabiduría me han formado como un profesional al servicio de la comunidad.

Iván Toromoreno

AGRADECIMIENTO

Agradezco a Dios, por darme la sabiduría y la fortaleza para sobreponerme a los desafíos de la vida y con ello poder desarrollar este proyecto de la mejor manera. A mis padres, quienes con su amor y excelente ejemplo de rectitud me han formado como un adulto justo y responsable. A mis directores Maritzol Tenemaza y Adrián Eguez por brindarme sus conocimientos a lo largo de mi desarrollo profesional; y a mis hermanos y amigos por su apoyo y buenos consejos.

Iván Toromoreno

CONTENIDO

1 INTRODUCCIÓN 21

1.1 Planteamiento del problema 21

1.2 Objetivos 23

1.2.1 Objetivo General 23

1.2.2 Objetivos específicos 23

1.3 Marco Teórico 24

1.3.1 Metodología de desarrollo de software 24

1.3.2 Scrum 25

1.3.3 Control de versiones 28

1.3.4 DevOps 30

1.3.5 CI/CD 31

1.3.6 Computación en la nube 32

1.3.7 Mínimo producto viable 33

1.3.8 Aplicación web 33

1.3.9 Patrón arquitectónico de software 34

1.3.10 MVC – Patrón de arquitectura: Modelo - Vista - Controlador 35

1.3.11 Pruebas unitarias 36

1.3.12 Jest 37

1.4 Herramientas, frameworks y librerías de desarrollo 37

1.4.1 Azure DevOps 37
1.4.2 Firebase 38
1.4.3 Angular 38
1.4.4 Heroku 39
1.4.5 Figma 39
1.4.6 Trello 40
1.4.7 Nicepage 40
2 Metodología 40
2.1 Sprint 0 40
2.1.1 Arquitectura del sistema 41
2.1.2 Roles de usuario 42
2.1.3 Diseño de UI/UX del sistema 43
2.1.4 Diccionario de datos 43
2.1.5 Diagrama entidad relación 44
2.1.6 Estructura de los identificadores de los ítems de trabajo 45
2.1.7 Roles de trabajo 45
2.1.8 Épicas 45
2.1.9 Features 47
2.1.10 Historias de usuario 47
2.1.11 Definición de sprints 48

2.1.12 Configuración del entorno de trabajo 49
2.2 Sprint 1 50
2.2.1 Sprint Planning 50
2.2.2 Ejecución del Sprint 52
2.2.3 Sprint Review 56
2.2.4 Sprint Retrospective 59
2.3 Sprint 2 61
2.3.1 Sprint Planning 61
2.3.2 Ejecución del Sprint 62
2.3.3 Sprint Review 64
2.3.4 Sprint Retrospective 66
2.4 Sprint 3 68
2.4.1 Sprint Planning 68
2.4.2 Ejecución del Sprint 73
2.4.3 Sprint Review 75
2.4.4 Sprint Retrospective 79
2.5 Sprint 4 82
2.5.1 Sprint Planning 82
2.5.2 Ejecución y cierre del sprint 83
3 RESULTADOS Y DISCUSIÓN 83

3.1 Producto final 83
3.1.1 Modulo general de la aplicación 84
3.1.2 Módulo de donadores 89
3.1.3 Módulo de donatarios 92
3.2 Documentación en Azure DevOps 96
3.2.1 Centro de documentos del proyecto (Wiki) 97
3.2.2 Dashboards 98
3.2.3 Boards 99
3.2.4 Repos 102
3.2.5 Pipelines 103
3.2.6 Discusión de esquema de trabajo 104
3.3 Pruebas funcionales con actores externos 105
3.3.1 Fase de pruebas sobre la versión alfa 106
3.3.2 Fase de pruebas sobre prototipo 107
4 CONCLUSIONES Y RECOMENDACIONES 108
4.1 Conclusiones 108
4.2 Recomendaciones 109
5 REFERENCIAS BIBLIOGRÁFICAS 110
6 110
7 ANEXOS 112

7.1 Anexo I: Épicas del sistema 112

7.2 Anexo II: Features del sistema 114

7.3 Anexo III: Historias de usuario Sprint 1 117

7.4 Anexo IV: Historias de usuario Sprint 2 119

7.5 Anexo V: Historias de usuario Sprint 3 120

7.6 Anexo VI: Diseño de las interfaces de usuario final 126

7.7 Anexo VII: Pruebas funcionales 133

7.8 Anexo VIII: Encuesta de usabilidad 137

TABLA DE ILUSTRACIONES

Ilustración 2 - Esquema de trabajo de Scrum [8] 25

Ilustración 1 - Organización de ramas definida por gitflow [19] 30

Ilustración 3 - Arquitectura del sistema 42

Ilustración 4 - Diagrama entidad relación 44

Ilustración 5 - Épica EC01 46

Ilustración 6 - Épica ED01 46

Ilustración 7 - Épica ER01 46

Ilustración 8 - Historia de usuario USC010201 50

Ilustración 9 - Historia de usuario USC010202 51

Ilustración 10 - Historia de usuario USR010101 51

Ilustración 11 - Historia de usuario USR010101 51

Ilustración 12 - Commit inicial en el árbol de cambios 53

Ilustración 13 - Ejecución de pipeline del proyecto 53

Ilustración 14 - Configuración de conexión en proyecto firebase 54

Ilustración 15 - Configuración de conexión firebase en archivos de entorno Angular 54

Ilustración 16 - Árbol de versionamiento del proyecto posterior a la integración con firebase 55

Ilustración 17 - UI Pantalla de creación de casos 55

Ilustración 18 - Pantalla de bienvenida 57

Ilustración 19 - Pantalla de inicio de sesión 58

Ilustración 20 - Pantalla de creación de cuenta 58

Ilustración 21 - Pantalla de solicitud de cuenta de usuario donatario 59

Ilustración 22 - Burndown Chart - Sprint 1 60

Ilustración 23 - Historia de usuario USR010101 61

Ilustración 24 - Historia de usuario USD010204 61

Ilustración 25 - Historia de usuario USD010101 62

Ilustración 26 - Pantalla principal del sistema al finalizar el sprint 2 63

Ilustración 27 - Pantalla de visualización de casos al finalizar el sprint 2 63

Ilustración 28 - Pantalla de registro y edición de casos 65

Ilustración 29 - Registros de casos ingresados en la base de datos firestore 65

Ilustración 30 - Historia de usuario USC010303 66

Ilustración 31 - Burndown Chart - Sprint 2 67

Ilustración 32 - Historia de usuario USC010201 68

Ilustración 33 - Historia de usuario USD010103 68

Ilustración 34 - Historia de usuario USR010203 69

Ilustración 35 - Historia de usuario USR010204 69

Ilustración 36 - Historia de usuario USD010202 69

Ilustración 37 - Historia de usuario USD010102 69

Ilustración 38 - Historia de usuario USR010102 70

Ilustración 39 - USD010203 70

Ilustración 40 - USD010206 70

Ilustración 41 - USD010205 70

Ilustración 42 - USC010101 71

Ilustración 43 - USC010303 71

Ilustración 44 - Pantalla de administración del sistema 73

Ilustración 45 - Pantalla de donaciones por medio de botón de pagos 74

Ilustración 46 - Pantalla de administración de donaciones - Usuario donatario 75

Ilustración 47 - Pantalla de exploración de casos de donación 79

Ilustración 48 - Burndown Chart - Sprint 3 80

Ilustración 49 - Tableros de desempeño del equipo - Sprint 3 - 1 81

Ilustración 50 - Tableros de desempeño del equipo - Sprint 3 – 2 81

Ilustración 51 - Pruebas funcionales del sistema. 82

Ilustración 52 - Diseño general UI/UX de la aplicación 84

Ilustración 53 - Versión final de la pantalla de inicio 85

Ilustración 54 - Versión final de la pantalla de ayuda 86

Ilustración 55 - Pantalla final de registro 87

Ilustración 56 - Pantalla final de inicio de sesión 87

Ilustración 57 - Pantalla de administración - administración de casos 88

Ilustración 58 - Pantalla de administración - administración de usuarios 88

Ilustración 59 - Pantalla principal del sistema - Usuario donatario 89

Ilustración 60 - Pantalla de exploración de casos 90

Ilustración 61 - Pantalla de detalle de caso de donación - Módulo donadores 91

Ilustración 62 - Pantalla de donaciones - Contacto con donatario 91

Ilustración 63 - Pantalla de monitoreo de donaciones 92

Ilustración 64 - Postulación como usuario donatario - Paso 1 93

Ilustración 65 - Postulación como usuario donatario - Paso 2 93

Ilustración 66 - Postulación como usuario donatario - Paso 3 94

Ilustración 67 - Postulación como usuario donatario - Paso 4 94

Ilustración 68 - Pantalla de edición de casos 95

Ilustración 69 - Pantalla de administración de donaciones 95

Ilustración 70 - Pantalla de exploración de casos creados 96

Ilustración 71 - Wiki - Overview del proyecto 97

Ilustración 72 - Historial de construcciones 98

Ilustración 73 - Desempeño del equipo - Sprint 3 99

Ilustración 74 - Tablero de ítems de trabajo 100

Ilustración 75 - Tablero de historias de usuario al finalizar el proyecto 101

Ilustración 76 - Backlogs de proyecto al finalizar la fase de desarrollo 101

Ilustración 77 - Tablero kanban de tareas - Sprint 3 102

Ilustración 78 - Historial de commits sobre rama master 103

Ilustración 79 - Historial de construcción de la app por medio del pipeline configurado . 104

Ilustración 80 - Resultados de evaluación de percepción de uso del sistema. 107

Ilustración 81 - Épica EC01 112

Ilustración 82 - Épica ED01 113

Ilustración 83 - Épica ER01 113

Ilustración 84 - Feature FC0101 114

Ilustración 85 - Feature FC0102 114

Ilustración 86 - Feature FC0103 115

Ilustración 87 - Feature FD0101 115

Ilustración 88 - Feature FD0102 116

Ilustración 89 - Feature FR0101 116

Ilustración 90 - Feature FR0102 117

Ilustración 91 - Historia de usuario USC010201 - Extendida..... 117

Ilustración 92 - Historia de usuario USC010202 - Extendida..... 118

Ilustración 93 - Historia de usuario USR010101 - Extendida..... 118

Ilustración 94 - Historia de usuario USR010201 - Extendida..... 119

Ilustración 95 - Historia de usuario USD010204 - Extendida..... 119

Ilustración 96 - Historia de usuario USD010101 - Extendida..... 120

Ilustración 97 - Historia de usuario USC010201 - Extendida..... 120

Ilustración 98 - Historia de usuario USD010103 - Extendida..... 121

Ilustración 99 - Historia de usuario USR010203 - Extendida..... 121

Ilustración 100 - Historia de usuario USR010204 - Extendida..... 122

Ilustración 101 - Historia de usuario USD010202 - Extendida..... 122

Ilustración 102 - Historia de usuario USD010102 - Extendida..... 123

Ilustración 103 - Historia de usuario USR010102 - Extendida..... 123

Ilustración 104 - Historia de usuario USD100203 - Extendida..... 124

Ilustración 105 - Historia de usuario USD010206 - Extendida..... 124

Ilustración 106 - Historia de usuario USD010205 - Extendida..... 125

Ilustración 107 - Historia de usuario USC010101 - Extendida..... 125

Ilustración 108 - Historia de usuario USC010303 - Extendida..... 126

Ilustración 109 - Diseño UI - Pantalla de inicio..... 126

Ilustración 110 - Diseño UI - Pantalla de inicio de sesión 127

Ilustración 111 - Diseño UI - Pantalla de registro 127

Ilustración 112 - Diseño UI - Pantalla principal 128

Ilustración 113 - Diseño UI - Pantalla de detalle de casos 128

Ilustración 114 - Diseño UI - Pantalla genérica de exploración 129

Ilustración 115 - Diseño UI - Pantalla general de administración 129

Ilustración 116 - Diseño UI - Pantalla de donaciones 130

Ilustración 117 - Diseño UI - Pantalla de registro de donatarios 1 130

Ilustración 118 - Diseño UI - Pantalla de registro de donatarios 2 131

Ilustración 119 - Diseño UI - Pantalla de registro de donatarios 3 131

Ilustración 120 - Diseño UI - Pantalla de registro de donatarios 4 132

Ilustración 121 - Diseño UI - Pantalla de registro de casos 132

Ilustración 122 - Prueba funcional - Login de usuario 133

Ilustración 123 - Prueba funcional - Probar registro de un usuario 133

Ilustración 124 - Prueba funcional - Creación de caso 134

Ilustración 125 - Prueba funcional - Pantalla de ayuda 134

Ilustración 126 - Prueba funcional - Preguntas frecuentes 135

Ilustración 127 - Prueba funcional - Ingresos sin verificación de correo 135

Ilustración 128 - Prueba funcional - Solicitud para ser usuario donatario 135

Ilustración 129 - Prueba funcional - Búsqueda de casos específicos 136

Ilustración 130 - Prueba funcional - Creación de donación 136

Ilustración 131 - Prueba funcional - Casos disponibles (pantalla principal) 136

Ilustración 132 - Prueba funcional - Lista de donaciones (realizadas) 137

Ilustración 133 - Prueba funcional - Lista de donaciones (recibidas) 137

Ilustración 134 - Encuesta - Lista de usuarios encuestados 138

Ilustración 135 - Encuesta - Pregunta 1 138

Ilustración 136 - Encuesta - Pregunta 2 139

Ilustración 137 - Encuesta - Pregunta 3 139

Ilustración 138 - Encuesta - Pregunta 4 140

Ilustración 139 - Encuesta - Pregunta 5 140

Ilustración 140 - Encuesta - Pregunta 6 141

Tabla 1 - Roles de trabajo 45

Tabla 2 - Features del sistema 47

Tabla 3 - Historias de usuario 48

Tabla 4 - Fechas de iteraciones del proyecto 49

Tabla 5 - Backlog Sprint 1 52

Tabla 6- Evaluación de criterios de aceptación - Sprint 1 57

Tabla 7 - Backlog - Sprint 2 62

Tabla 8 - Evaluación de criterios de aceptación – Sprint 2 64

Tabla 9 – Backlog – Sprint 3 72

Tabla 10 - Evaluación de criterios de aceptación - Sprint 3 78

Tabla 11 - Sujetos de prueba de fase beta del aplicativo 106

RESUMEN

El presente trabajo busca demostrar la implementación de la cultura de desarrollo denominada DevOps para la construcción de un sistema web de donación de recursos, el cual está enfocado en la colaboración entre los miembros de la comunidad. La aplicación resultante del presente proyecto es un prototipo que será entregado a la Escuela Politécnica Nacional y podrá ser llevado a producción por una entidad gubernamental realizando las modificaciones que se consideren necesarias.

El proyecto hace uso de la herramienta Azure DevOps como eje central de desarrollo. La herramienta es utilizada durante todo el ciclo de vida del proyecto, abarcando las tareas de administración (haciendo uso del framework Scrum), implementación del sistema (haciendo uso del control de versiones git) y finalmente la integración de pipelines CI/CD que permitan la fácil inclusión de cambios progresivos que se hagan en la aplicación, demostrando así la interconexión entre las tareas de desarrollo y de integración.

Durante la realización de la aplicación se hizo uso del flujo de trabajo gitlfow para la integración continua de los cambios realizados, separando los ciclos de desarrollo de entrega funcional de los ciclos de corrección de errores. Los cambios fueron integrados en una rama principal la cual se encontró configurada con un pipeline de integración y despliegue continuo el cual permitió contar con un ambiente de staging en donde se realizaron las pruebas funcionales del sistema.

Palabras clave: DevOps, Scrum, Control de versiones, Sistema web, Integración continua, Despliegue continuo, Metodologías de desarrollo.

ABSTRACT

The current work tries to demonstrate the implementation of the DevOps development culture for the construction of a web system of resources donation, which is focus on the collaboration of the community members. The resulting application of the current project is a prototype that is going to be delivered to the Escuela Politécnica Nacional and may be put into production for any government entity making the necessary modifications.

The project makes use of Azure DevOps as main axis of development. The tool is used the whole life cycle of the project, encompassing the administration tasks (using Scrum framework), system implementation (using git versions control system) and finally the CI / CD pipelines integration that allow the easy inclusion of progressive changes that were making on the application, demonstrating in that way the interconnection between the development and integration tasks.

During the realization of the application the gitflow workflow was used for the continuous integration of the changes made, separating the cycles of functional delivery development from the bugs corrections cycles. The changes were integrated on a main branch which was configured with a continuous integration and continuous delivery pipeline that allowed to have a staging environment where the functional tests of the system were made.

Key words: DevOps, Scrum, Versions control, Web system, Continuous integration, Continuous delivery, Development methodologies.

1 INTRODUCCIÓN

1.1 Planteamiento del problema

El concepto de cultura organizativa fue desarrollado a mediados del siglo 20, la idea gira en torno a que un grupo de personas trabajando juntos en un ambiente corporativo pueden crear una cultura aparte de la cultura social en general. Las personas crean estas culturas en base a valores y comportamientos compartidos [1].

En la industria de la construcción de software siempre se ha intentado buscar una metodología de desarrollo que permita tener un equilibrio entre el bienestar de los desarrolladores (no solamente en el ámbito económico sino también personal y psicológico) y los plazos de entrega dispuestos por los clientes, los cuales, muchas veces al no tener conocimientos técnicos suelen ser irreales con respecto a los requerimientos establecidos en los sistemas a desarrollar.

Previamente las metodologías de desarrollo utilizadas solían ser sumamente rígidas en la definición de requerimientos y en las fases del ciclo de vida de la construcción de un sistema, lo cual conllevaba a tener costos altos de retrabajo, pruebas que debían esperar a que toda la fase de implementación esté terminada y un bajo nivel de interacción con el cliente [2].

Las nuevas metodologías de desarrollo conocidas como metodologías ágiles han sido altamente adoptadas por las empresas encargadas de construir sistemas informáticos, ya que permiten una mayor comunicación con los clientes y una mayor flexibilidad para poder construir sistemas que cumplan con todas las necesidades que los usuarios finales esperen solventar con las aplicaciones entregadas [3].

Las metodologías ágiles como XP permiten a los desarrolladores entender de mejor manera los requerimientos a implementar brindando una mayor rapidez en la entrega de valor a los clientes, sin embargo, el equipo de operaciones continúa buscando una estabilidad en el sistema, haciendo que los cambios implementados se retrasen en su despliegue a un ambiente productivo. El roce entre ambos equipos suele generar retrasos en los tiempos de entrega y frustración entre todos los involucrados en el proceso de desarrollo [4].

El movimiento DevOps consiste en un cambio en la cultura organizativa y comienza a formarse entre los años 2007 y 2008 como respuesta a la confrontación entre los intereses de los equipos de desarrollo y operaciones, tratando de evitar tanto los cuellos de botella provocados por el equipo de operaciones como los cambios que generan fallos en el producto realizados por los equipos de desarrollo, la propuesta de DevOps nace como una contra parte al enfoque tradicional de desarrollo (en donde cada equipo era un silo aislado) intentando facilitar la interconexión de ambos frentes haciendo uso herramientas y principios compartidos [5].

DevOps es una cultura organizativa que se apoya en workflows y herramientas que permiten a equipos de trabajo adaptarse a su filosofía. Sin embargo, es crucial entender los ideales detrás del movimiento DevOps para que los grupos de trabajo tengan claro cuáles son los valores y comportamientos esperados por cada miembro del equipo [1].

DevOps puede ser visto como la interacción entre el desarrollo y las operaciones de software desde distintas categorías, las cuales son:

- **Cultura:** Crear una cultura en donde el desarrollo y las operaciones de software interactúen de forma regular.
- **Automatización:** Automatizar pasos que requieran interacción entre el personal de desarrollo y operaciones.
- **Medición** / **Métricas:** Implementar mediciones que abarquen las disciplinas de desarrollo y operaciones, o las interacciones entre el personal de desarrollo y operaciones, o dar acceso al personal de desarrollo a métricas usadas por el personal de operaciones y viceversa.
- **Apoyo:** Optimizar la interacción entre el personal de desarrollo y operaciones.
- **Compartir:** Tener un espacio en donde el personal de desarrollo y operaciones interactúen, o mantener sistemas usados por el personal de desarrollo y operaciones para interactuar, o la compartición entre el personal de desarrollo y operación sobre su intervención en diferentes partes de un mismo sistema [2].

En la actualidad existe una deficiencia en el entendimiento común de lo que DevOps constituye tanto en la academia como las comunidades que lo practican. La investigación en el fenómeno de DevOps es necesaria para poder examinar el impacto en el desarrollo y operaciones de software [6].

Por lo antes descrito se busca realizar el desarrollo de un sistema web haciendo uso de la filosofía de DevOps para sentar de forma clara los procesos, herramientas, roles y relaciones que involucran adaptarse a este cambio cultural en el ciclo de vida de un proyecto con el fin de estudiar la mejora que significa contar con una sinergia entre los equipos de desarrollo y operaciones además de la automatización de procesos que ayudan a la entrega rápida de valor y a la transparencia entre los actores que interactúan para la implementación de un sistema informático.

Se busca generar un portal web que permita la recaudación de fondos para personas de escasos recursos. Durante el ciclo de vida se hará uso de DevOps para el desarrollo y despliegue continuo de funcionalidad del portal.

Se busca adaptarse a las mejores prácticas relacionadas a la filosofía DevOps. La utilización de esta metodología permitirá hacer énfasis en las relaciones entre desarrollo y operaciones durante el ciclo de vida de desarrollo del proyecto.

Finalmente se espera entregar un portal web funcional junto con la documentación que respalde la forma en la que el sistema fue desarrollado bajo la filosofía DevOps, recolectando las mejores prácticas, herramientas y recomendaciones a seguir para implantar dicha filosofía.

1.2 Objetivos

1.2.1 Objetivo General

Desarrollar una aplicación web utilizando herramientas y cultura DevOps con métricas.

1.2.2 Objetivos específicos

- Definir las prácticas, herramientas y métricas DevOps a ser utilizadas y medidas durante el proyecto.
- Establecer los pipelines DevOps de trabajo dentro del ciclo de vida del proyecto y el proceso de medición de métricas.
- Diseñar e implementar un portal web que permita interconectar a personas interesadas en realizar donaciones con personas en situación de vulnerabilidad acoplándose a las prácticas y pipelines DevOps.

- Evaluar las métricas obtenidas dentro de los flujos y procesos DevOps

1.3 Marco Teórico

1.3.1 Metodología de desarrollo de software

Una metodología de desarrollo de software es la forma en la que se administra un proyecto de construcción de software. Los elementos de los que se compone una metodología de software son los siguientes:

- **Roles:** Consisten en los papeles que pueden tener de los miembros de un equipo y de la descripción del trabajo que debe realizar cada persona para lograr un objetivo común. Los ejemplos más comunes son: arquitecto de software, diseñador, desarrollador, administrador del proyecto.
- **Habilidades:** Es el conjunto de conocimientos que se espera que tengan los miembros de un equipo. Por ejemplo: diseño de interfaces de usuario, programación en un framework específico o administración de un proyecto de software.
- **Equipos:** Es la manera en la que los equipos se agrupan y a la vez son asignados a los roles determinados dentro de la organización. Por ejemplo: el equipo de desarrollo, el equipo de seguridad, etc.
- **Técnicas:** La forma en la que se realiza el trabajo asignado.
- **Actividades:** El conjunto de acciones y ceremonias que realizan los roles del equipo. Por ejemplo: Reunión de revisión de avances, desarrollo de entregables, etc.
- **Productos del trabajo:** Son el conjunto de resultados obtenidos al seguir el proceso de construcción del software y vienen definidos por los requerimientos, la calidad, el contenido e incluso por los receptores.
- **Estándares:** Lo que es y no es permitido en un producto de trabajo. Por ejemplo: guías de estilos, estilos de escritura de código, la forma en la que se escriben los requerimientos del sistema, etc.
- **Calidad:** Es definida por las expectativas que se tienen del producto a ser entregado. Por ejemplo: cantidad de fallos que el sistema puede tener al año.

- **Valores:** Son las normas sociológicas que comparten y promueven los miembros del equipo. Por ejemplo: Scrum tiene como valores: coraje, foco, compromiso, respeto y apertura. [7]

1.3.2 Scrum

Scrum es un marco de trabajo que busca facilitar el desarrollo de productos complejos que generen alto valor de negocio de formas ágiles y adaptativas. El marco de trabajo fue creado por Ken Schwaber y Jeff Sutherland con la finalidad de brindar un esquema de organización a los equipos que trabajan en el desarrollo de sistemas de software altamente complejos. Scrum hace uso del empirismo con la finalidad de reemplazar una perspectiva de construcción algorítmica con una heurística. La forma en la que Scrum organiza el trabajo se puede ver reflejada en la ilustración 2 en donde se inicia definiendo un backlog del producto y por medio de iteraciones se realiza la implementación de nuevas funcionalidades que entreguen valor al producto enfocados en los valores del negocio. [8]

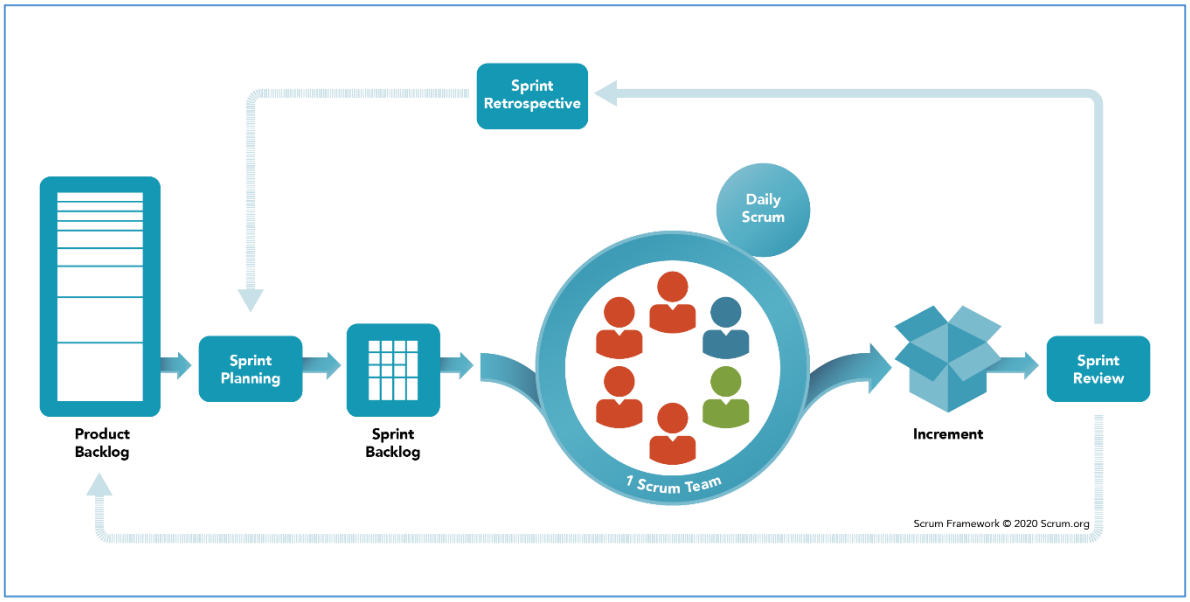

Ilustración 1 - Esquema de trabajo de Scrum [8]

1.3.2.1 Valores de Scrum

Los valores en los que está fundamentado el marco de trabajo Scrum son [9]:

- **Coraje:** el cual consiste en tener el suficiente coraje para asumir los desafíos y la incertidumbre inherentes al proyecto.
- **Foco:** para mantener al equipo enfocado en la meta de cada iteración y del proyecto a nivel global.
- **Compromiso:** las personas del equipo se comprometen a resolver sus tareas y a buscar ayuda cuando lo consideren necesario.
- **Respeto:** los miembros del equipo scrum deben tener respeto por su trabajo y por el trabajo realizado por sus compañeros, así como los conocimientos y capacidades que cada uno puede aportar.
- **Apertura:** El equipo de trabajo y los interesados deben mostrar apertura a los cambios de requerimientos y de estimaciones que puedan surgir a lo largo del desarrollo del proyecto.

1.3.2.2 Equipo Scrum

Los miembros de un equipo Scrum pueden desempeñar los siguientes roles [10]:

- **Product Owner:** Es la persona responsable de definir las historias de usuario del proyecto y de priorizarlas dentro del backlog para garantizar que las entregas se alineen a las metas del negocio. El PO es responsable de monitorizar y velar por el incremento de valor generado en el producto en cada iteración. Las tareas específicas de un product owner son:
    - Desarrollar y comunicar explícitamente la meta del producto.
    - Crear y comunicar claramente los ítems del backlog.
    - Ordenar según su prioridad los ítems creados.
    - Asegurarse de que el backlog sea visible, transparente y entendible.
- **Scrum Master:** El Scrum master es el responsable de que el equipo aplique la metodología de forma correcta como está descrita en la guía oficial de Scrum, es decir que ayudan al equipo a entender la teoría y la práctica que todos los miembros del equipo requieren conocer para llevar a cabo sus tareas de acuerdo a los roles que desempeñan. Los scrum master sirven a los equipos de la siguiente manera [11]:
    - A los equipos de desarrollo:
        - Entrenando a los miembros en las funcionalidades que deben llevar a cabo.

        - Ayudando a mantener el foco en las metas del negocio.
        - Eliminando los impedimentos que evitan que el equipo avance a un ritmo adecuado.
        - Asegurándose de que los eventos definidos por Scrum sean llevados a cabo y que sean productivos para el equipo.
    - Al product owner:
        - Ayudando a buscar técnicas para administrar el backlog de tareas y definir la meta del producto de forma efectiva.
        - Ayudando al equipo a entender la necesidad de los ítems de trabajo
        - Ayudando a establecer planeación de producto empírica por entornos complejos
        - Facilitando la interacción con los interesados según sea necesario o requerido.
    - A la organización:
        - Liderando, entrenando y entrenando a la organización en la adopción de Scrum.
        - Planificando y notificando las implementaciones de Scrum dentro de la organización.
        - Ayudando a los empleados y a los interesados en abordar el trabajo complejo con una aproximación empírica.
        - Removiendo las barreras entre los interesados y los equipos Scrum.
- **Equipo de desarrollo:** El equipo de desarrollo es el responsable por crear cada aspecto de un producto con incrementos funcionales entregados al final de cada iteración. Los miembros del equipo de desarrollo deben [12]:
    - Crear un plan para cada iteración, es decir un backlog para cada sprint.
    - Buscar la mejor calidad haciendo uso de una definición de terminado.
    - Adaptar su plan de trabajo cada día para conseguir las metas de la iteración.
    - Haciéndose responsables los unos de los otros como profesionales.

1.3.2.3 Ceremonias Scrum

Scrum define las siguientes ceremonias como parte de su marco de trabajo:

- **Sprint:** Los sprints son el núcleo de Scrum y son eventos de larga duración en donde todo el equipo involucrado trabaja en realizar incrementos de valor en el

producto que está siendo implementado. Cada sprint empieza cuando el sprint anterior ha terminado [13].

- **Sprint planning:** La planeación del sprint es la ceremonia que da inicio al sprint y es el evento en el cual se definen la meta del sprint, las historias de usuario que pueden ser realizadas y cómo se realizará el trabajo seleccionado [14].
- **Daily Scrum:** Es una ceremonia que se lleva a cabo todos los días con el fin de monitorizar el progreso del equipo de cara a las metas definidas en la planeación del sprint. La ceremonia se enfoca en establecer el trabajo que ha sido realizado por el equipo el día anterior, el trabajo que se va a llevar a cabo en el día de la ceremonia y determinar impedimentos que eviten el correcto desenvolvimiento de los miembros del equipo [15].
- **Sprint review:** Es la ceremonia que se lleva a cabo al finalizar el sprint y consiste en mostrar el trabajo realizado por el equipo de desarrollo a los demás miembros del equipo Scrum y a los interesados en el proyecto con la intención de inspeccionar el avance del proyecto y realizar adaptaciones al backlog en caso de ser necesario [16].
- **Sprint retrospective:** Es la ceremonia que da cierre al sprint y busca establecer las mejores prácticas dentro del equipo con la finalidad de aumentar su eficacia de cara a la entrega de valor en futuras iteraciones. Los tópicos que son tomados en cuenta en esta ceremonia son: las acciones positivas que han sido realizadas por el equipo, las acciones negativas o mejorables, los compromisos que ayuden a mejorar las acciones negativas encontradas [17].

1.3.3 Control de versiones

El control de versiones es una práctica que consiste en guardar un registro de los cambios realizados en el código de un producto de software con la finalidad de rastrear y gestionar la evolución del sistema a lo largo del tiempo. El control de versiones se ve soportado por herramientas que permiten a los equipos de software gestionar los cambios en el código fuente a lo largo de la construcción de un sistema, permitiendo de esta manera que se tenga una gran herramienta de conocimiento general compartida entre todos los integrantes del grupo de trabajo [18].

Entre las ventajas que tiene el uso del control de versiones dentro de un equipo de desarrollo de software es la capacidad de regresar el sistema a una versión previa en caso

de que se presente un error durante la ejecución del sistema en un ambiente de pruebas o un ambiente productivo. La mayor parte de sistemas de control de versiones permiten además tener un registro de las personas encargadas de realizar cambios dentro del sistema de manera que tengamos un contexto más cercano de quién pudo ser el responsable de una funcionalidad que no cumpla con los requerimientos definidos [18].

Los sistemas de control de versiones más usados como git manejan la estructura de los archivos en una estructura de tipo árbol, lo cual permite a los desarrolladores tener separación entre el trabajo que están realizando, aunque este sea implementado de forma paralela; es decir, en caso de que dos desarrolladores se encuentren trabajando en un mismo sistema, el control de versiones permite que sus cambios no se vean afectados por los cambios realizados por otro miembro del equipo. Esta ventaja del control de versiones viene acompañado junto a la capacidad de resolver conflictos en el código fuente de forma manual entre los implicados en el desarrollo del sistema [18].

Gitflow

Gitflow es un flujo de trabajo que puede ser implementado dentro del control de versiones, permite que los desarrolladores organicen de forma ordenada los cambios que implementan dentro del sistema y define un esquema de trabajo común que debe ser compartido entre todos los involucrados en la escritura del código fuente con la finalidad de evitar solapamiento en el trabajo que se realice, la corrección de forma adecuada de los cambios realizados e incluso la definición de ambientes de trabajo mediante la utilización de diferentes ramas que estén siendo trabajadas por el equipo [19].

La organización de ramas de trabajo definida por gitflow consiste en tener una rama principal (usualmente conocida como main o master) en la cual se almacenará el código productivo, una rama de desarrollo que usualmente es usada para ambientes de pruebas o staging y varias ramas de funcionalidades que derivan de la rama de desarrollo. Adicionalmente se cuenta con ramas que se desprenden de la rama principal orientadas a la solución de bugs que puedan aparecer en ambientes productivos conocidas como “hotfixes”. La ilustración 1 muestra de forma gráfica el esquema definido por gitflow para el manejo de ramas de un proyecto [19].

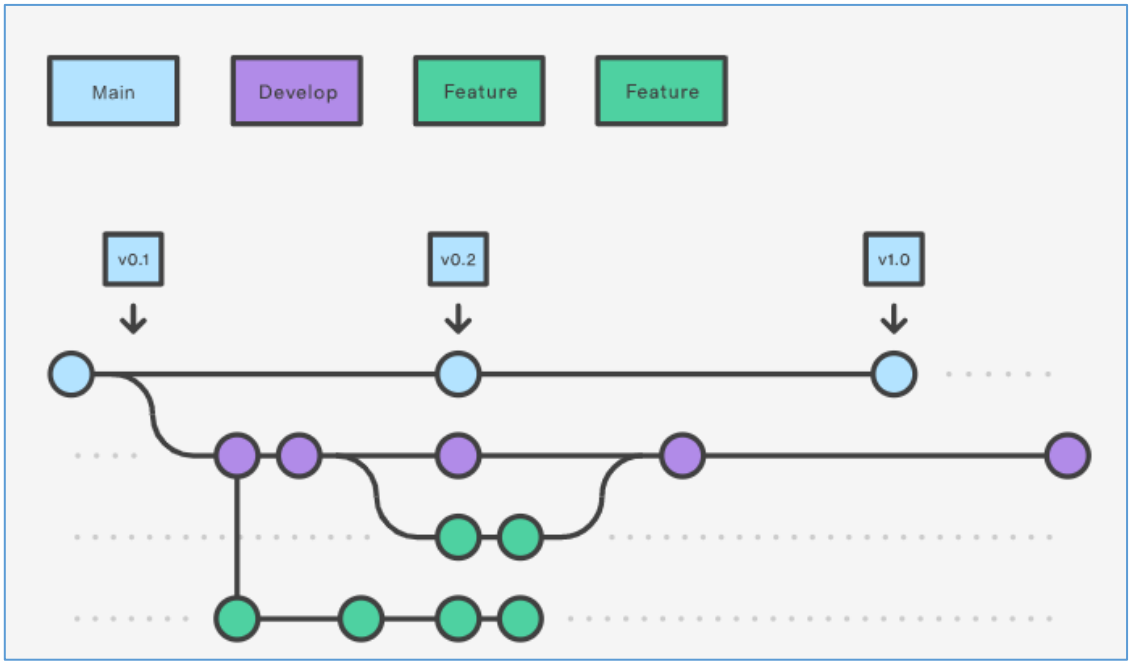

Ilustración 2 - Organización de ramas definida por gitflow [19]

1.3.4 DevOps

DevOps es un concepto que puede tomar varios significados dependiendo el contexto o la organización. Una de las definiciones más acertadas es considerar a DevOps un esquema de trabajo que permite a los miembros de un equipo compartir la responsabilidad sobre las tareas de tal forma que desarrollen empatía por el esfuerzo de los demás, permitiendo de esta manera que todos lleven a cabo su trabajo de forma efectiva y duradera, además de hacer uso de automatización para llevar a cabo tareas repetitivas con menor esfuerzo y costo [20].

DevOps consiste en compartir las metas del equipo entre todos sus miembros, de tal manera que exista una mayor transparencia en el trabajo que se realiza en el día a día, lo cual ayuda a la organización a entregar más valor a sus clientes en menor tiempo, además permite a los equipos de desarrollo y operaciones tener mayor sincronización y confianza en el trabajo que realizan y los resultados que esperan de los demás [20].

Las herramientas de las que hace uso DevOps como docker o chef no son el eje central del marco de trabajo, tampoco las metodologías con las cuales se está realizando la construcción de un producto de software. El eje central de DevOps es la cultura de la

organización, que permite a todos los miembros del equipo automatizar su trabajo de tal manera que todos se sientan conectados entre sí y puedan desarrollar sus tareas de forma más confiable y ágil siendo influenciados principalmente por metodologías ágiles de desarrollo de software [20].

La cultura de las organizaciones que implementan DevOps debe estar comprometida fuertemente con la colaboración entre los equipos de trabajo, de tal manera que el esfuerzo que realiza cada uno ayude a conseguir las metas propuestas por los otros. De igual manera se debe contar con la capacidad de automatizar los procesos repetitivos que se realizan al momento de implementar un producto. El compromiso establecido debe ser a todos los niveles, los desarrolladores y los encargados de los equipos de operaciones deben poner su esfuerzo en desarrollar herramientas de automatización mientras que los gerentes deben permitir a sus equipos desarrollar estas herramientas que no son vistas en una entrega final al cliente pero que son usadas constantemente por los equipos internos [20].

1.3.5 CI/CD

Los sistemas o productos que son implementados por los equipos de desarrollo son puestos a disposición de los clientes mediante procesos de despliegue, los cuales consisten en la planeación, mantenimiento y ejecución de la entrega de un artefacto de software que genera valor al negocio. El proceso de despliegue debe tener en cuenta el entorno de trabajo en el cual se encuentra funcionando el sistema productivo y dependiendo del tipo de sistema que se requiera desplegar, se debe conocer también los pasos necesarios para que los cambios realizados se integren de forma correcta. Dentro de la automatización de estos procesos nacen los conceptos de integración, despliegue y entrega continua, los cuales se definen como [20]:

Integración continua: La integración continua es el proceso que consiste en integrar los cambios realizados por un desarrollador dentro del código fuente del código productivo, al trabajar con esquemas como gitflow. El proceso de integración continua consiste en hacer un merge de las ramas de desarrollo de funcionalidades específicas con la rama principal generalmente conocida como main o máster. Dentro del proceso de integración continua se incluyen procesos como la automatización de pruebas para comprobar que los cambios realizados por los desarrolladores no introduzcan fallos al sistema principal, lo cual genera mayor confianza en las implementaciones realizadas [20].

Entrega continua: La entrega continua es un conjunto de principios de ingeniería de software que ayudan a los desarrolladores a establecer reglas que permitan determinar que los cambios realizados a un sistema pueden ser desplegados a un ambiente productivo sin realizar daños en los procesos de pruebas automatizadas o construcción de artefactos funcionales, usualmente estos principios son integrados dentro del proceso de integración continua ya que ambos se encuentran fuertemente relacionados [20].

Despliegue continuo: El proceso de despliegue continuo consiste en desplegar los cambios realizados por el equipo de desarrollo en un ambiente productivo para el cliente con la ayuda de pruebas y validaciones que permitan reducir el riesgo de provocar fallos en el producto. La diferencia con los procesos previamente listados consiste en que tanto la integración como la entrega continua permiten que los cambios estén listos para ser desplegados, sin embargo, el proceso de despliegue continuo asegura que dichos cambios hayan sido integrados con el producto en el ambiente de producción. La integración de un proceso de despliegue continuo en el proyecto trae una gran cantidad de ventajas entre las cuales están: mayor cantidad de entregas al cliente, menor tiempo de integración de los cambios realizados, mayor tiempo de respuesta en caso de fallos en el sistema y mayor aprendizaje por parte de todo el equipo involucrado en la construcción de un sistema de software [20].

1.3.6 Computación en la nube

La nube se refiere a un entorno de TI diseñado para acceder a recursos escalables y medibles de forma remota. La nube nace como un concepto paralelo al internet, el cual es básicamente un conjunto de redes interconectadas entre sí. El concepto de computación en la nube se refiere a una nueva forma de construcción y alojamiento de artefactos de software que juntos forman un sistema informático; el esquema convencional de alojamiento de aplicaciones es el uso de servidores “on premise” el cual consiste en tener un entorno de trabajo dentro de servidores físicos que son propiedad de la organización encargada de administrar los sistemas construidos; por otra parte el esquema de computación en la nube consiste en hacer uso de servicios entregados por proveedores externos para alojar distintas funcionalidades del sistema, las cuales pueden ir desde un proveedor de alojamiento de bases de datos hasta complejas arquitecturas basadas completamente en proveedores externos a la organización [21].

1.3.7 Mínimo producto viable

El concepto de mínimo producto viable (MVP por sus siglas en inglés) proviene de las metodologías lean y consiste básicamente en un prototipo de software el cual está enfocado en crear una propuesta de valor haciendo uso de pocos recursos, tanto económicos como humanos para su realización con la finalidad de evaluar si una propuesta de negocio tiene valor en el mercado en el cual está pensado su funcionamiento. El enfoque que defiende la realización de un mínimo producto viable es evitar la pérdida de recursos en entregables que no generen valor o que no contengan el enfoque correcto dentro de las metas del negocio [22].

El inicio de la definición de un mínimo producto viable consiste en establecer un prototipo con las mínimas funcionalidades que pueda ser utilizable por los usuarios de los cuales se espera sea utilizado el sistema, a gran nivel los pasos que se deben seguir para desarrollar con MVP son [22]:

1. Establecer el mercado al cual se busca vender el producto, este puede ir desde un grupo de estudiantes hasta los habitantes de una región determinada.
2. Establecer el valor del producto que se va a realizar, es fundamental que antes de comenzar el desarrollo del prototipo se tenga clara la meta que se desea conseguir.
3. Desarrollar los caminos de usuario, los cuales son casos reales de uso del prototipo con la finalidad de entender los puntos de fricción en los cuales el usuario pueda tener problemas de usabilidad con el sistema.
4. Establecer el diseño y realizar pruebas de usabilidad para validar que los usuarios potenciales del sistema se sientan cómodos con el producto que utilizaran incluso antes de que este sea desarrollado.
5. Construir el MVP realizando iteraciones de forma continua que entreguen valor al mercado seleccionado en el paso inicial de la realización del prototipo [22].

1.3.8 Aplicación web

El término aplicación web es definido por la “Java Servlet Specificacion v2.3” como una colección de servlets, páginas html, clases y otros recursos que pueden ser empaquetados en diferentes contenedores de diferentes proveedores y puede ser referenciado por rutas específicas en un entorno web generalmente expuesto al internet. Las aplicaciones

establecen reglas de conexión en cada una de sus rutas con la finalidad de asegurar los recursos expuestos y de mapear las funcionalidades esperadas por los usuarios [23].

SPA (Aplicaciones de una sola página)

Las aplicaciones web pueden ser construidas en distintos esquemas de representación, los más utilizados son el esquema de aplicación multi páginas y el esquema de aplicación de una sola página, el cual consiste en tener una aplicación que funcione dentro del navegador y obtenga todos los recursos necesarios para su funcionamiento al hacer la primera llamada web al servidor. Las llamadas al navegador realizadas por las aplicaciones SPA son únicamente con el objetivo de obtener los datos relevantes para el usuario que deben ser desplegados en los recursos previamente obtenidos en la llamada inicial. Los principales beneficios obtenidos al desarrollar aplicaciones SPA son [24]:

- Mejor velocidad de ejecución y una experiencia de usuario más fluida.
- Mayor facilidad de depuración con herramientas de desarrollo provistas por los navegadores.
- Facilidad y simpleza de desarrollo al evitar implementar lógica de representación en el lado del servidor.
- Mayor modularización de los componentes que confirman la aplicación [24].

1.3.9 Patrón arquitectónico de software

Los patrones de arquitectura de software consisten en soluciones reutilizables a problemas recurrentes de forma similar a como trabajan los patrones de diseño, pero con un alcance más amplio dentro del diseño de un sistema. Los patrones arquitectónicos más comunes se pueden observar a detalle en la tabla 1 [25].

Patrón arquitectónico	Características	Aplicaciones
Capas	El sistema se estructura en capas con diferentes responsabilidades que se comunican entre sí.	Aplicaciones de escritorio, aplicaciones web de comercio electrónico.
Maestro esclavo	Consiste en un componente maestro que distribuye el trabajo del sistema a los componentes esclavos para la obtención de un resultado final.	Replicación de base de datos, periféricos conectados a un bus de un sistema informático.
Filtro de tubería	El sistema se organiza en componentes que procesan la información de forma secuencial hasta obtener un resultado.	Compiladores, flujos de trabajo de bioinformática.
Agente	Consiste en varios componentes que se comunican y coordinan entre sí por medio de un componente intermediario.	Software de message bróker.
Modelo vista controlador	Consiste en tres componentes principales (modelo, vista y controlador) con funciones específicas dentro del sistema.	Arquitectura de aplicaciones web, marcos de trabajo web.

Tabla 1 - Patrones arquitectónicos de software [25]

En base a los patrones estudiados previamente se ha optado por implementar el patrón modelo vista controlador como base para el diseño del aplicativo, ya que este se encuentra por definición dentro del marco de trabajo angular gracias a su gran manejo de responsabilidades dentro de los componentes que conforman una aplicación web.

1.3.10 MVC – Patrón de arquitectura: Modelo - Vista - Controlador

El patrón de arquitectura MVC ha tenido una gran relevancia en las ciencias de la computación desde el año 1979 (cuando era conocido como TMVE o Cosa – Modelo – Vista

– Editor) por la solución elegante pero poderosa que ofrece al separar las responsabilidades entre los diferentes componentes que conforman un sistema, permitiendo la separación entre las implementaciones de la lógica de acceso a datos y la lógica de representación de los datos en la interfaz de usuario final. La separación de conceptos ayuda a los desarrolladores a establecer un esquema de trabajo claro y a reducir la incertidumbre de la implementación realizada por los demás miembros del equipo, aun cuando el patrón añade cierta complejidad al desarrollo [25].

Los componentes que conforman al patrón MVC son:

- **Modelo:** Es la representación de los datos almacenados en la aplicación junto con las reglas de negocio que definen cómo van a cambiar en el tiempo.
- **Controlador:** Son los interceptores de la comunicación proveniente del usuario y ayudan a la interconexión entre los componentes de la arquitectura.
- **Vista:** Es la definición de cómo la interfaz gráfica será representada a los usuarios [25].

1.3.11 Pruebas unitarias

Las pruebas unitarias dentro de un sistema de software consisten en piezas de código encargadas de comprobar las funcionalidades que se esperan de una parte atómica de la aplicación. La principal utilidad que tienen las pruebas unitarias al momento de desarrollar un sistema es la de comprobar que los requerimientos se están cumpliendo de forma correcta al nivel de funciones. El enfoque de trabajar con pruebas unitarias al momento de desarrollar un sistema informático permite verificar que a medida que la aplicación crece en el tiempo, las funcionalidades no se han vuelto obsoletas o han dejado de funcionar de la forma que es esperada. Las fases en las cuales consiste una prueba unitaria son [27]:

1. **Arrange:** es la fase en donde son definidas las variables y requerimientos para la función que será probada.
2. **Act:** es la fase en donde se ejecuta la acción que se desea probar y se almacenan los resultados obtenidos.
3. **Assert:** es la fase en la cual se comprueba que los resultados obtenidos en la función ejecutada coincidan con los resultados esperados según las especificaciones.

1.3.12 Jest

Jest es un framework de pruebas para javascript cuya filosofía está enfocada en mantener la simplicidad en las pruebas que son escritas dentro del sistema. El marco de trabajo es compatible con distintos tipos de proyectos que hagan uso de: babel, typescript, node, react, angular, vue entre otros. Las principales características del framewerk son [28]:

- No requiere configuración previa para poder funcionar sobre un proyecto.
- Capacidad de probar grandes cantidades de código en línea o dentro del proyecto.
- Las pruebas son ejecutadas de forma paralela para maximizar el desempeño.
- Gran documentación del api de pruebas.
- Informe de pruebas y cobertura de código detallados.
- Facilidad de realizar mocks que reemplacen servicios reales.

1.4 Herramientas, frameworks y librerías de desarrollo

1.4.1 Azure DevOps

Azure Devops es una herramienta de administración de proyectos de software que posee varias funcionalidades para los equipos de desarrollo, operaciones y gerencia. Las funcionalidades utilizadas para el desarrollo del presente proyecto son [26]:

- **Summary:** esta funcionalidad permite mostrar un resumen general del proyecto alojado en la plataforma.
- **Dashboards:** permite mostrar en tableros información relevante del desempeño del equipo.
- **Wiki:** son archivos compartidos por todos los miembros del equipo con información relevante del proyecto.
- **Work items:** es el repositorio de los ítems de trabajo del equipo, dependiendo la configuración puede mostrar las épicas, funcionalidades e historias de usuario del proyecto.
- **Boards:** consiste en un tablero kaban en el cual se refleja el estado de las historias de usuario definidas en el proyecto.
- **Backlogs:** permite gestionar las historias de usuario en las distintas iteraciones definidas por el proyecto.

- **Sprints:** es una funcionalidad que refleja el avance del equipo dentro de las iteraciones definidas por el equipo haciendo uso de un tablero kanban de tareas, un backlog de historias de usuario, la configuración de la capacidad de los miembros del equipo en el tiempo y análisis por medio de charts de desempeño.
- **Repos:** es un conjunto de repositorios de tipo git en donde se almacenará el código fuente del proyecto.
- **Pipelines:** son plantillas de automatización de procesos de integración y despliegue continuo.

1.4.2 Firebase

Firebase es un servicio provisto por google que permite almacenar distintas funcionalidades relacionadas con computación en la nube. La plataforma de firebase tiene su núcleo en Google Cloud Services por lo que sus métodos de integración son altamente confiables y robustos. Las funcionalidades de firebase utilizadas en el presente proyecto son [27]:

- **Firestore:** es un servicio de base de datos no relaciones que permite almacenar los datos del proyecto en forma de documentos similares a una estructura json.
- **Authentication:** es una herramienta que permite la integración del proyecto con proveedores de autenticación externos como correo electrónico, Facebook o google. Firebase authentication permite también gestionar la sesión de los usuarios dentro del aplicativo.
- **Storage:** la funcionalidad de storage permite almacenar recursos de la aplicación tales como archivos multimedia o documentos pdf o docx, en una organización de directorios.

1.4.3 Angular

Angular es una plataforma de desarrollo que está construida sobre typescript e incluye las siguientes funcionalidades [28]:

- Un framework de construcción de aplicaciones basado en componentes para el desarrollo de aplicaciones web robustas.

- Una colección de librerías que integran funcionalidades comúnmente requeridas por desarrolladores de aplicaciones web como son: el manejo de rutas y formularios, comunicación de lado del cliente y más.
- Un conjunto de herramientas de desarrollo que ayudan a desarrollar, construir, probar y actualizar las aplicaciones implementadas.

Angular basa el desarrollo de las aplicaciones en las siguientes ideas centrales:

- **Componentes:** los componentes son bloques de construcción que componen la aplicación final. Los componentes contienen un archivo typescript con la lógica de necesaria para el funcionamiento, un archivo html que sirve de plantilla y un archivo de estilos que se encarga de estilizar el componente implementado.
- **Plantillas:** las plantillas son el html contenido dentro de los componentes de la aplicación. Una plantilla puede ser definida dentro del archivo typescript o puede ser referenciada por su ruta relativa.
- **Inyección de dependencias:** la inyección de dependencias permite declarar las dependencias de las clases componentes del proyecto sin tener que lidiar con la instanciación de estas, facilitando de esta manera el desarrollo y las pruebas del sistema.

1.4.4 Heroku

Heroku es una plataforma que ofrece servicios en la nube en un esquema de plataforma como servicio (PaaS) y es usada para el despliegue, administración y escalamiento de aplicaciones web. La plataforma contiene soporte para distintas tecnologías que son utilizadas en el ámbito laboral y académico como nodejs, ruby, spring, entre otras. La plataforma es sumamente robusta y es altamente mantenida dando a los desarrolladores confianza de la estabilidad del servicio que es provisto y de la experiencia de desarrollo que heroku ofrece [30].

1.4.5 Figma

Figma es una herramienta de diseño de interfaces gráficas que permite definir tanto la experiencia como la interfaz de usuario (UI/UX) con un conjunto amplio de funcionalidades como son: organización páginas y capas de las vistas de usuario, diseño de componentes

los cuales contienen los atributos que pueden ser implementados posteriormente en una hoja de estilos y prototipos que permiten definir flujos de usuario dentro de la aplicación, de tal manera que los desarrolladores conozcan a detalle lo que se espera mostrar en la implementación final del producto [31].

1.4.6 Trello

Trello es una herramienta de organización de tickets de trabajo en formato “Kanban” el cual consiste en clasificar los ítems creados en distintos estados dependiendo de los parámetros definidos por el equipo de trabajo. La forma de organizar el trabajo bajo la metodología kanban consiste en cuatro principios los cuales son: empezar con el estado actual o lo que se está realizando, comprometerse a buscar cambios incrementales y evolutivos, respetar los procesos, responsabilidades y cargos actuales y animar el liderazgo en todos los niveles [32].

1.4.7 Nicepage

Nicepage es una herramienta de construcción de plantillas de sitios web que permite diseñar de forma fácil interfaces de usuario y generar archivos que puedan ser usados en distintos entornos de trabajo. La ventaja de usar herramientas como nicepage consiste en la agilidad de construcción de páginas que manualmente construidas requerirían mucho tiempo y esfuerzo en su implementación pero que al ser generadas automáticamente contienen funcionalidades como responsividad y estilos dinámicos [33].

2 Metodología

2.1 Sprint 0

El sprint 0 fue utilizado para la definición de esquemas de trabajo, estándares en el nombramiento de historias de usuario, diseño de experiencia de usuario y un repaso breve sobre la arquitectura del aplicativo. La duración de esta iteración fue de aproximadamente dos semanas y los resultados obtenidos se verán listados en la presente sección. Adicionalmente el sprint 0 sirvió para configurar las herramientas y entorno de trabajo que se utilizarán a lo largo del desarrollo del proyecto.

2.1.1 Arquitectura del sistema

La arquitectura del sistema toma como base la separación de responsabilidades entre los componentes que conforman el aplicativo. Los componentes que interactúan para proveer las funcionalidades del sistema se muestran en la ilustración 3 y se listan a continuación.

- **Ambiente de pruebas:** consiste en un servidor alojado en el servicio Heroku el cual contendrá los cambios que se realicen en cada sprint.
- **Aplicación frontend:** consiste en una aplicación desarrollada con el framework Angular y separada en distintos componentes que permitan visualizar la separación de responsabilidades entre cada uno de ellos. Los componentes serán agrupados en los siguientes grandes grupos:
    - **Donantes:** el grupo de donantes contiene los componentes enfocados en la funcionalidad de los usuarios donantes del sistema como son la creación de usuarios donantes, la exploración de casos y la administración de donaciones.
    - **Donatarios:** el grupo de donatarios contiene los componentes enfocados en la funcionalidad de los usuario donatarios del sistema como son la administración de casos, la creación de casos y la administración de donaciones.
    - **Común:** el grupo común contiene los componentes y clases que son de uso común por los demás grupos del sistema como son la administración de cuenta, el almacenamiento y la autenticación.

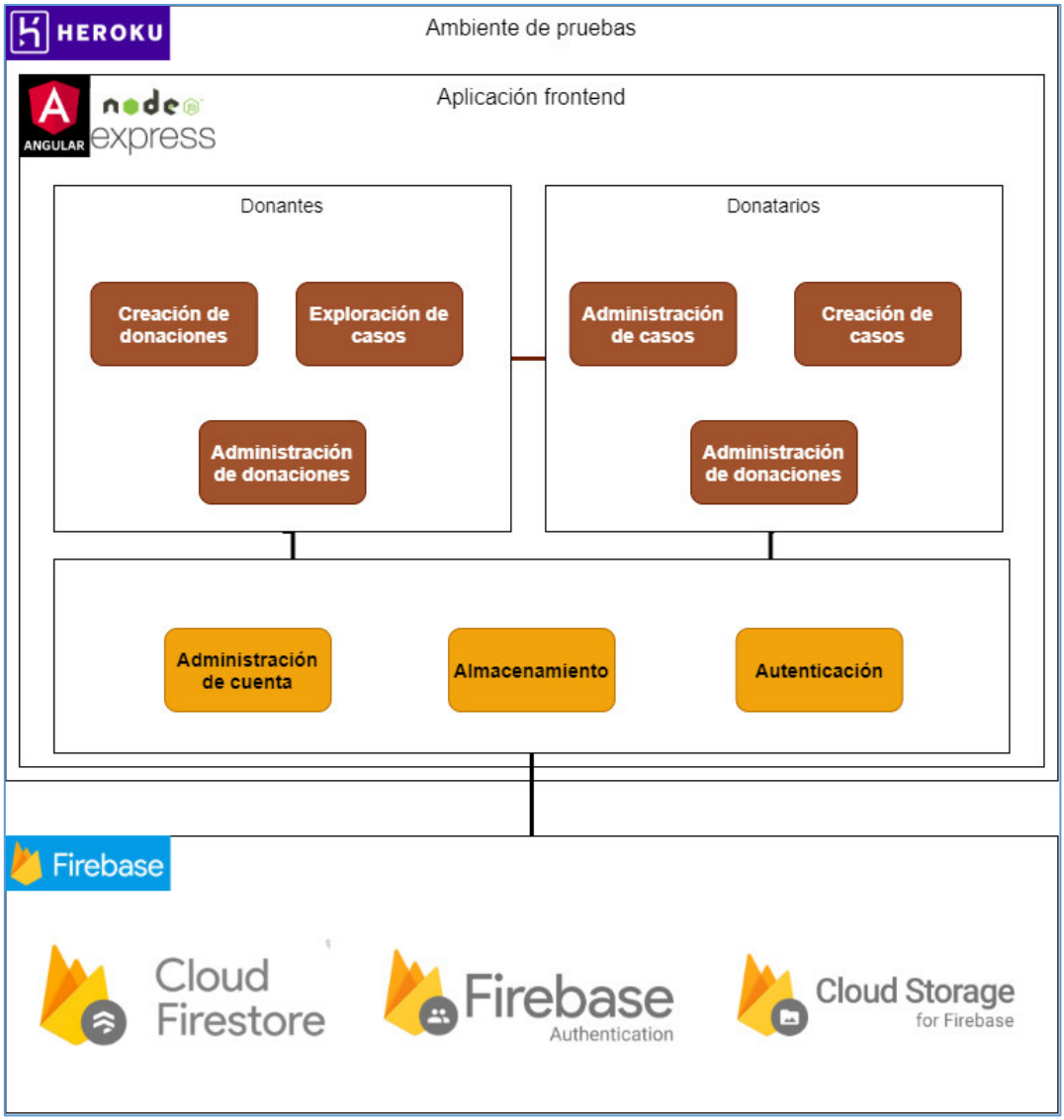

Ilustración 3 - Arquitectura del sistema

2.1.2 Roles de usuario

El sistema contará con tres roles de usuario los cuales tendrán las siguientes funcionalidades:

- **Donador:** Es el usuario encargado de realizar donativos a casos de donación creados en el sistema, puede:
    - Explorar casos existentes
    - Filtrar casos existentes
    - Buscar entre los casos existentes

    - Realizar donaciones
    - Administrar las donaciones realizadas
    - Explorar las donaciones realizadas
- **Donatario:** Es un usuario donador que es aprobado para poder crear casos de donación dentro del sistema, puede:
    - Realizar las acciones descritas para un usuario donador
    - Crear casos de donación
    - Administrar donaciones recibidas
    - Explorar donaciones recibidas
- **Administrador:** Es un usuario especial que sirve como apoyo a los demás usuarios con las siguientes funcionalidades:
    - Aprobación de solicitudes de usuarios donatarios
    - Administración de usuarios donatarios
    - Administración de casos creados.

2.1.3 Diseño de UI/UX del sistema

El diseño de las interfaces de usuario se llevó a cabo en la herramienta Figma y está basado en el diseño utilizado por páginas altamente utilizadas como son Facebook y Platzi. El objetivo principal del diseño es tener una interfaz de usuario amigable de cara al usuario final que permita llevar a cabo las acciones del sistema de una forma fácil y entendible. El diseño puede ser revisado a detalle en el Anexo I del presente documento.

2.1.4 Diccionario de datos

El diccionario de datos busca estandarizar la forma en la que los distintos documentos almacenados en firebase son creados y modificados al hacer uso del sistema. Los datos que contienen un diccionario son los siguientes:

- **Estado de casos de donación:** cuyo valor puede ser:
    - Activo
    - Cerrado
    - Bloqueado
- **Estado de donaciones:** cuyo valor puede ser:
    - Pendiente

- Confirmada
- Negada
- En revisión

2.1.5 Diagrama entidad relación

El diagrama mostrado en la ilustración 4 representa la estructura de los datos en una base de datos SQL, sin embargo, al hacer uso del servicio firestore de firebase para el almacenamiento de los datos, se seguirá un esquema NoSQL por lo cual se realizarán las siguientes adaptaciones:

- Las entidades donante y donatario serán almacenadas dentro de una misma colección llamada 'usuarios'.
- IFO será almacenado como un atributo de la colección 'usuarios'
- Las imágenes de casos de donaciones serán atributos de la colección 'casosDeDonacion'

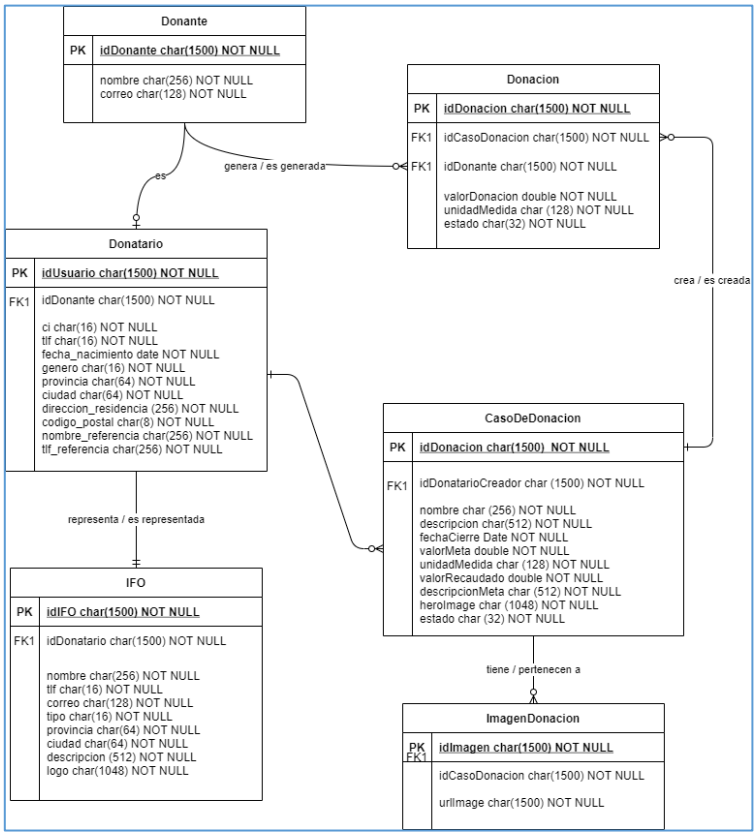

Ilustración 4 - Diagrama entidad relación

2.1.6 Estructura de los identificadores de los ítems de trabajo

Las siguientes especificaciones definirán la forma en la cual los identificadores de usuario deben ser establecidos para los ítems de trabajo.

- **Épicas:** Las épicas serán identificadas con la siguiente estructura:
  - E[Inicial del Módulo][Número de épica]
- **Features:** Los features serán idenficiados con la siguiente estructura:
  - F[Inicial del módulo][Número de épica madre][Número de feature]
- **Historias de usuario:** Las historias de usuario serán identificadas con la siguiente estructura:
  - US[Inicial del módulo][Número de épica madre de feature][Número de feature padre][Número historia de usuario]

2.1.7 Roles de trabajo

El equipo Scrum está conformado por dos integrantes, los cuales ocuparán los siguientes roles de trabajo dentro del proyecto:

Equipo Scrum	
Rol	**Integrante**
Product owner	Maritzol Tenemaza
Scrum master	Iván Toromoreno
Desarrollador	Iván Toromoreno

Tabla 2 - Roles de trabajo

2.1.8 Épicas

Las ilustraciones 5, 6 y 7 son una representación de épicas definidas para desarrollar todas las funcionalidades del sistema. La lista de épicas puede ser explorada a detalle en el Anexo II del presente documento.

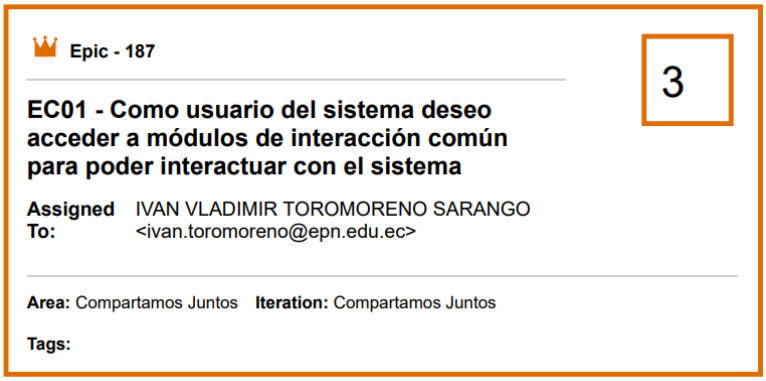

Ilustración 5 - Épica EC01

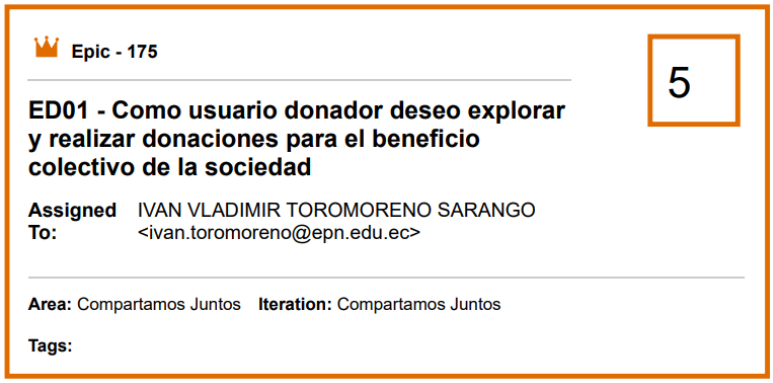

Ilustración 6 - Épica ED01

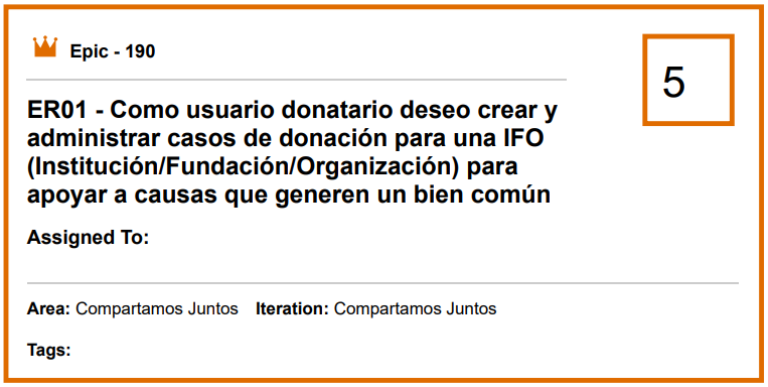

Ilustración 7 - Épica ER01

2.1.9 Features

La tabla 3 muestra una lista resumida de las funcionalidades que se han establecido para el sistema. La lista completa se muestra forma detallada en el Anexo III.

Title	Business Value
FC0101 - Como usuario del sistema deseo acceder a un módulo de ayuda y de preguntas frecuentes para conocer el funcionami...	2
FC0102 - Como usuario del sistema deseo poder registrarme y autenticarme dentro del sistema para poder interactuar con otro...	3
FD0101 - Como usuario donador deseo realizar donaciones a los creadores de casos para contribuir con sus proyectos sociales	5
FD0102 - Como usuario donador deseo explorar los casos de donación disponibles para poder colaborar con ellos	5
FR0101 - Como usuario donatario deseo postular en el sistema para acceder a la creación de casos	5
FR0102 - Como usuario donatario deseo crear y administrar casos de donación para evaluar su estado y darlos a conocer a los d...	5

Tabla 3 - Features del sistema

2.1.10 Historias de usuario

La tabla 4 muestra de forma resumida las historias de usuario que serán implementadas en el sistema de forma iterativa. La lista detallada de historias de usuario se muestra en la sección de anexos desglosadas por las iteraciones en las cuales fueron implementadas.

Title	Priority
USC010101 - Como usuario del sistema deseo ver una lista de preguntas frecuentes para conocer más de cómo funciona el siste...	1
USC010102 - Como usuario del sistema deseo acceder a un módulo de ayuda para conocer más sobre las funcionalidades del si...	1
USC010201 - Como usuario del sistema deseo poder registrarme para acceder a las funcionalidades ofrecidas	4
USC010202 - Como usuario deseo autenticarme en el sistema para poder acceder a las funcionalidades disponibles	4
USD010101 - Como usuario donador deseo ver los detalles de un caso específico para conocer más información sobre él.	4
USD010102 - Como usuario donador deseo explorar el estado de las donaciones que he realizado a un caso para administrar su...	4
USD010103 - Como usuario donador deseo ingresar los detalles de mi donación para que pueda ser llevada a cabo	4
USD010202 - Como usuario donador deseo explorar todos los casos a los cuales he contribuido para revisar su estado o realizar...	3
USD010203 - Como usuario donador deseo ver los casos activos en los cuales he donado anteriormente para poder realizar una...	2
USD010204 - Como usuario donador deseo ver los casos de donación activos en la pantalla principal para poder acceder a ellos ...	4
USD010205 - Como usuario donador deseo buscar un caso de donación específico para poder acceder a él de forma más eficie...	2
USD010206 - Como usuario donador deseo explorar las categorías de casos existentes para seleccionar uno de mi preferencia	2
USR010101 - Como usuario donatario deseo registrarme en el sistema para poder crear casos de donación	4
USR010102 - Como usuario donatario deseo ser notificado cuando mi cuenta haya sido habilitada para poder crear casos de do...	3
USR010201 - Como usuario donatario deseo registrar y editar mis casos de donación para que los demás usuarios puedan realiz...	4
USR010202 - Como usuario donatario deseo acceder a los casos de donación activos para poder administrarlos	2
USR010203 - Como usuario donatario deseo acceder a todos los casos que he creado para poder administrarlos	4
USR010204 - Como usuario donatario deseo administrar las donaciones de los casos que he creado para actualizar su el estado ...	4

Tabla 4 - Historias de usuario

2.1.11 Definición de sprints

Los sprints se definieron con una duración de 2 semanas (10 días laborales) y acorde a las circunstancias en las cuales se ha elaborado el actual proyecto se han establecido las fechas mostradas en la tabla 5 para la ejecución de cada iteración.

Fechas de ejecución de iteraciones		
Iteración	**Fecha de inicio**	**Fecha de finalización**
Iteración 1	20/02/2021	05/03/2021
Iteración 2	05/04/2021	16/04/2021
Iteración 3	31/07/2021	06/08/2021
Iteración 4	07/08/2021	13/08/2021

Tabla 5 - Fechas de iteraciones del proyecto

2.1.12 Configuración del entorno de trabajo

El sprint 0 fue usado para la configuración de las herramientas necesarias para el desarrollo del presente proyecto. Las herramientas utilizadas junto con su configuración se listan a continuación.

- Firebase: El proyecto de firebase ha sido configurado con una cuenta de Gmail y hace uso de funcionalidades sin costo de la plataforma. Las configuraciones específicas de la herramienta son:
    - Integración con angular mediante la habilitación en el proyecto y haciendo uso de la librería npm llamada angularfire.
    - Habilitación de autenticación por medio de correo electrónico
    - Creación de base de datos firestore en la región us-central
- Proyecto angular: El proyecto angular ha sido configurado con el cliente oficial del framework llamado ng-client. Las configuraciones específicas de la herramienta son:
    - Versión de node: 12.14.0
    - Versión de npm: 6.13.4
    - Versión de cliente angular: 12.1.1
    - Librerías instaladas en el proyecto:
        - Bootstrap: 4.3.1
        - Firebase: 5.5.7
        - Ngx-spinner: 8.1.0
- Azure DevOps: El proyecto en Azure DevOps ha sido configurado con las siguientes especificaciones:
    - Tipo de repositorio: Git
    - Tipo de administración del proyecto: Scrum

- Configuraciones adicionales:
  - Configuración de pipeline para CI/CD e integración con heroku. En el Anexo IV se podrá observar a detalle el script configurado.
  - Configuración del repositorio para integración con proyecto angular.

2.2 Sprint 1

2.2.1 Sprint Planning

Objetivo del Sprint

Implementar las funcionalidades relacionadas a la autenticación dentro del sistema junto al registro de casos de donación por parte de los usuarios donatarios. Probar las configuraciones de realizadas en el sprint 0 sobre los proyectos de Azure DevOps y heroku para comprobar su correcta integración y funcionamiento.

Historias de usuario

Las historias de usuario que fueron definidas para el Sprint 1 se muestran en las ilustraciones 8, 9, 10 y 11 y podrán ser observadas a detalle en la sección de anexos.

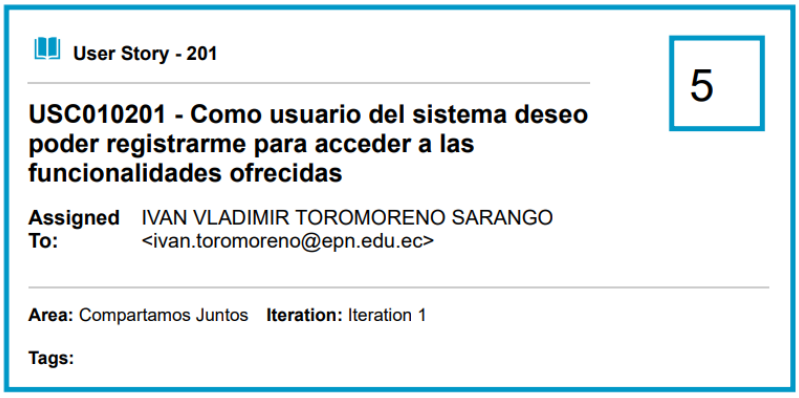

Ilustración 8 - Historia de usuario USC010201

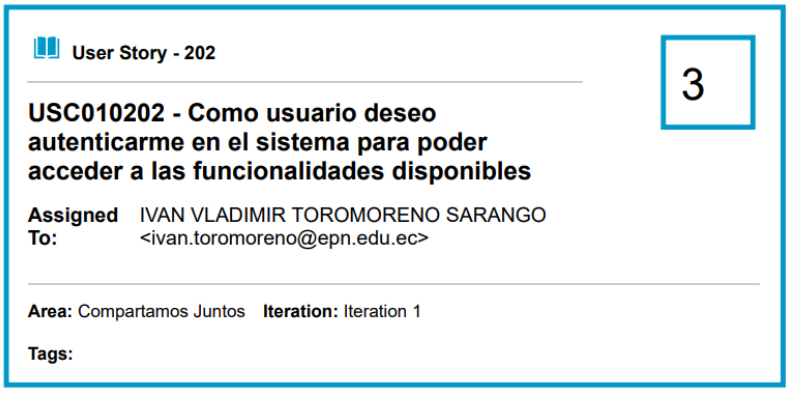

Ilustración 9 - Historia de usuario USC010202

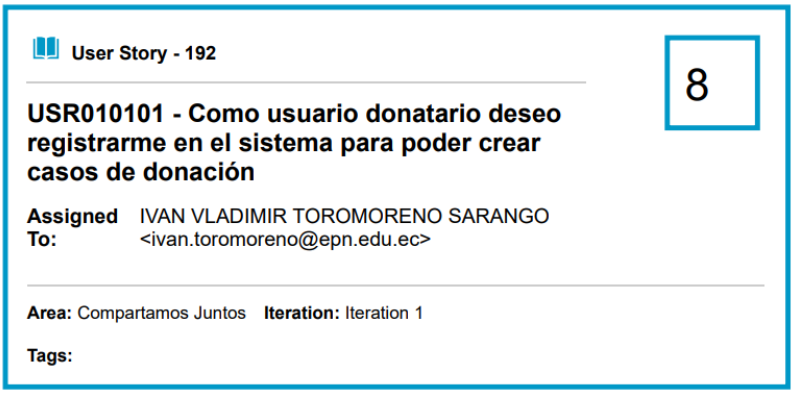

Ilustración 10 - Historia de usuario USR010101

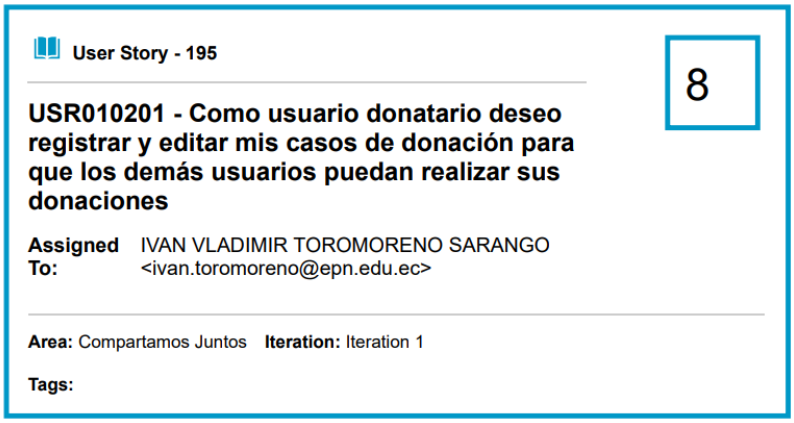

Ilustración 11 - Historia de usuario USR010101

Sprint backlog

La tabla 6 muestra las tareas definidas para las historias previamente definidas:

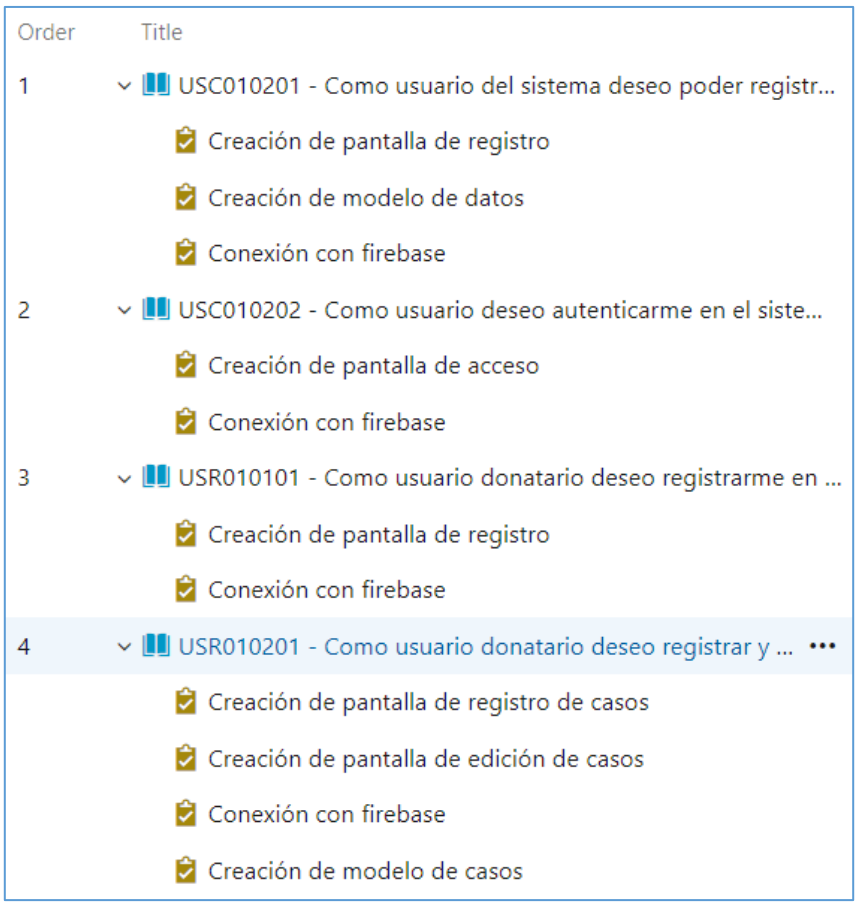

Tabla 6 - Backlog Sprint 1

2.2.2 Ejecución del Sprint

Durante la ejecución del sprint las siguientes actividades fueron relevantes:

Commit inicial del proyecto al repositorio Azure Git

El proyecto creado durante la ejecución del sprint 0 se integra al repositorio de Azure Git y se comprueba su correcto funcionamiento en el ambiente local de desarrollo. El commit inicial aún no hace uso del pipeline ejecutado con la finalidad de evitar fallos de carga. La ilustración 12, muestra el commit dentro del árbol del repositorio.

Ilustración 12 - Commit inicial en el árbol de cambios

Configuración CI/CD y despliegue de la aplicación a heroku

El repositorio del proyecto es configurado con pipelines de DevOps haciendo uso de un template predeterminado de construcción y pruebas, la ilustración 13 muestra la ejecución inicial del pipeline y se comprueba el correcto funcionamiento de la app en un enlace público.

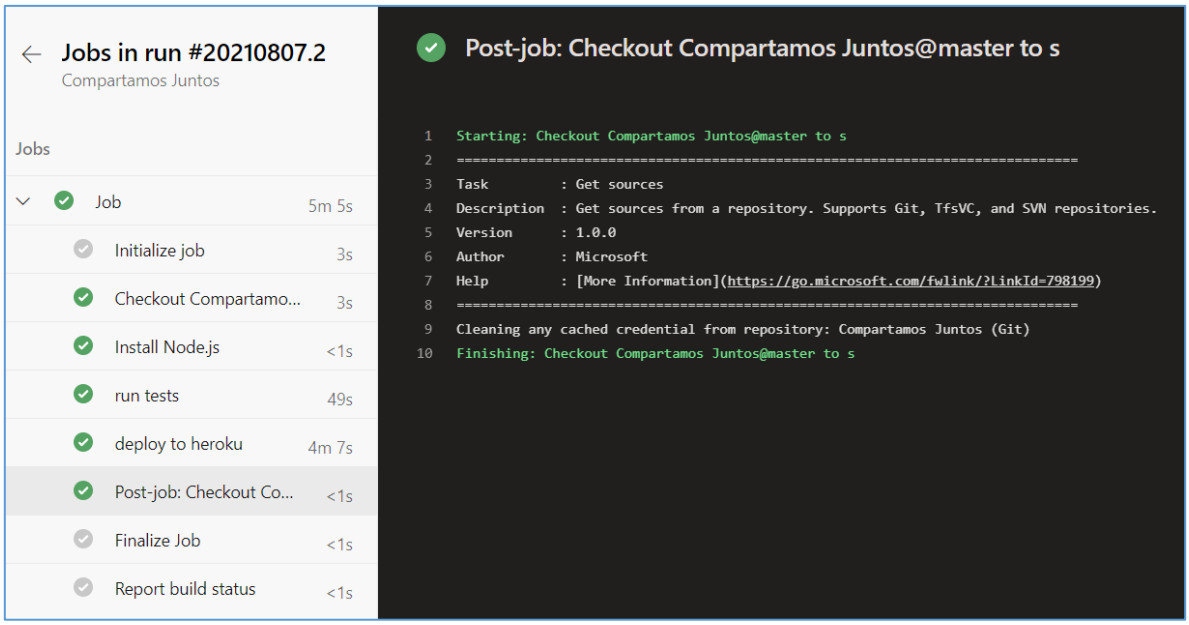

Ilustración 13 - Ejecución de pipeline del proyecto

Adicionalmente se realizan las ramificaciones necesarias para el desarrollo del proyecto junto con cambios en el archivo de CI para la ejecución de pruebas unitarias del proyecto haciendo uso de jest.

Conexión de firebase con proyecto Angular

La conexión del proyecto alojado en firebase con la aplicación implementada en el framework se realizó por medio de la librería angularfire. La ilustración 14 muestra la

configuración del proyecto, lo cual es el primer paso para la correcta integración entre las dos plataformas.

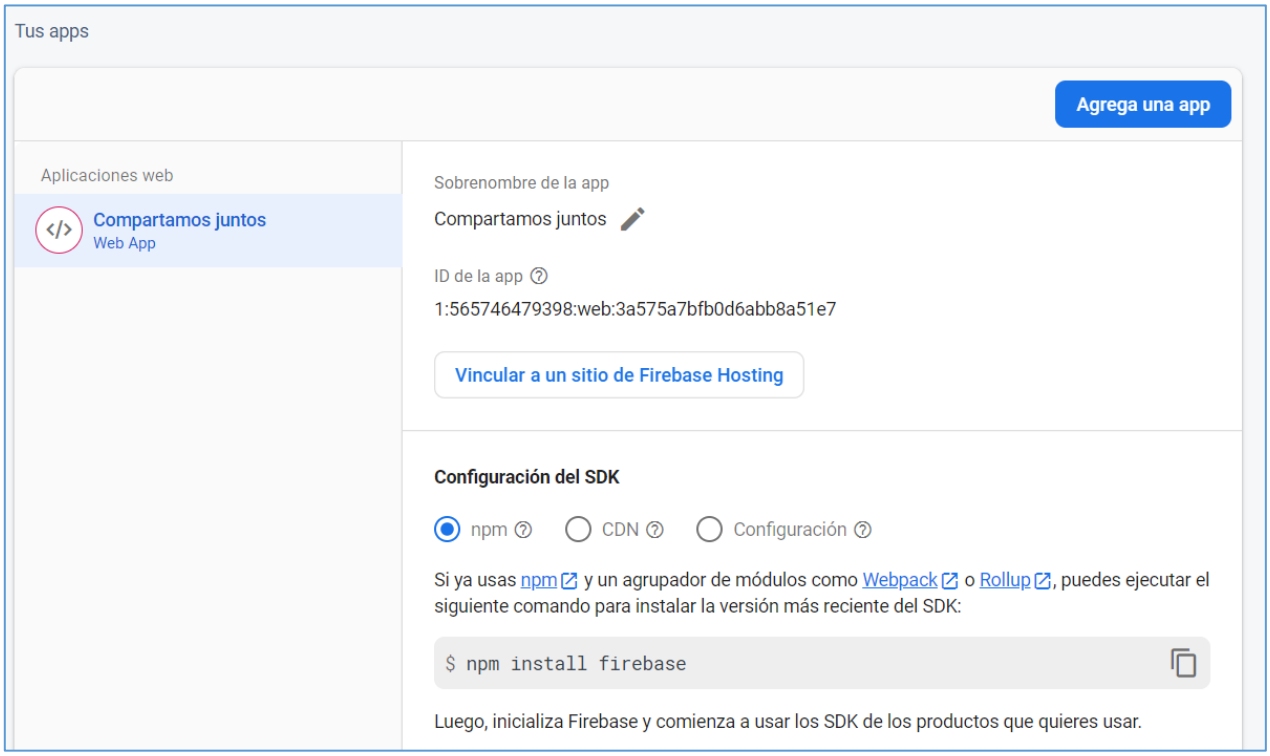

Ilustración 14 - Configuración de conexión en proyecto firebase

El proyecto de Angular fue configurado por medio de los archivos de entorno para poder realizar la conexión de forma automática al momento de desplegar la aplicación a Heroku, la figura 15 muestra los archivos de configuración para los ambientes de desarrollo local y producción.

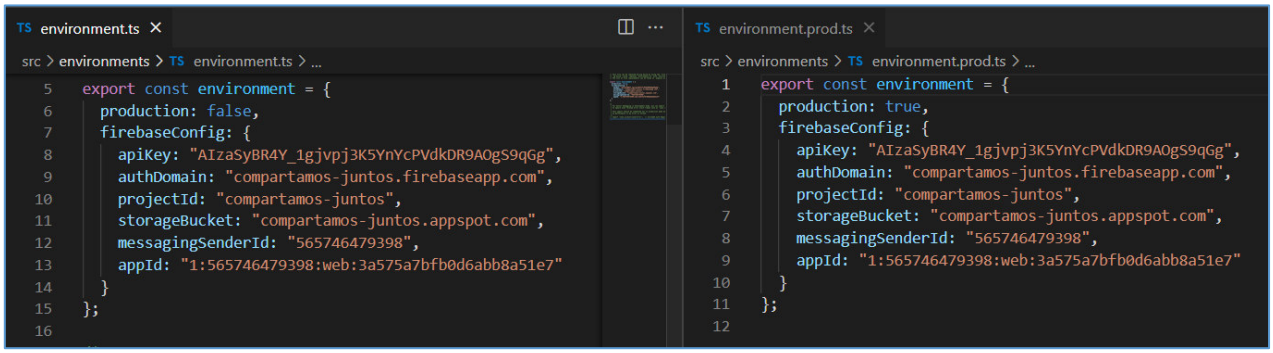

Ilustración 15 - Configuración de conexión firebase en archivos de entorno Angular

Finalmente se realizaron las pruebas de integración entre el proyecto Angular con los pipelines CI/CD que integran los cambios en el entorno de pruebas alojado en Heroku. La ilustración 16 muestra el árbol de versionamiento del proyecto hasta el estado funcional de la configuración previamente realizada.

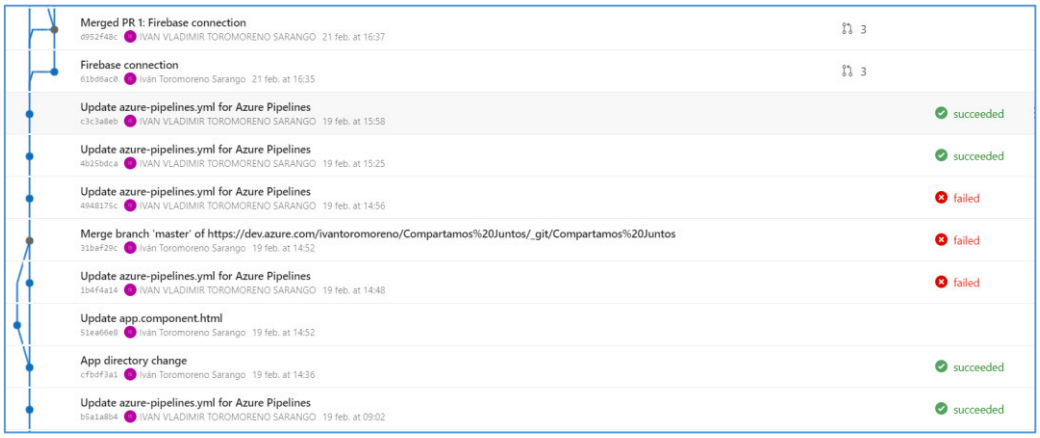

Ilustración 16 - Árbol de versionamiento del proyecto posterior a la integración con firebase

Programación de componentes

Posterior a la implementación de las configuraciones previamente realizadas se comenzó a realizar la programación de los componentes que respondan a las historias de usuario y tareas detalladas en el backlog del sprint. La ilustración 17 muestra la interfaz de usuario correspondiente al registro de casos dentro del sistema, la cual fue la última implementación realizada en el sprint.

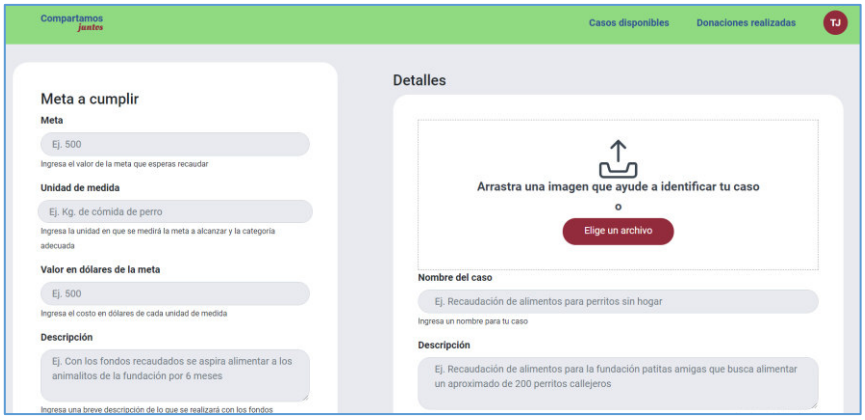

Ilustración 17 - UI Pantalla de creación de casos

2.2.3 Sprint Review

El objetivo planteado en el primer sprint no fue conseguido ya que se subestimó la historia relacionada a la creación y administración de casos, sin embargo, la mayor parte de las historias fueron validadas con respecto a los criterios de aceptación definidos. La tabla 7 muestra la evaluación realizada a las historias de usuarios implementadas en el sprint.

Historia de usuario	Criterio de aceptación	Cumple	Comentarios
USC010201	Se deberá mostrar el formulario de registro al dar click en la opción de registrarse en la pantalla inicial del sistema.	Sí	
	Se deberá validar que la información ingresada corresponda a lo detallado en la descripción	Sí	
	Se deberá indicar al usuario que su cuenta ha sido creada	Sí	
	Se deberá notificar al usuario por correo electrónico que su cuenta debe ser validada para poder acceder	Sí	
USC010202	Se mostrará el formulario de autenticación al usuario	Sí	
	Se validará que la información ingresada sea correcta	Sí	
	Se ingresará al sistema en caso de que las credenciales sean correctas	Sí	
	Se mostrará un mensaje indicando que las credenciales ingresadas son incorrectas en caso de que no lo sean	Sí	
USR010101	Se presentará un formulario de varios pasos en los cuales el usuario deberá llenar información personal y de la entidad a la cual brindará ayuda, en la última página del formulario se presentarán los términos y condiciones del sistema para poder crear casos de donación.	Sí	
USR010201	El usuario donatario podrá visualizar el formulario de creación de casos.	Sí	
	El usuario deberá ser informado de la correcta creación del caso y podrá visualizar los datos ingresados	Sí	
	El caso será visualizado por los usuarios donantes una vez que el caso haya sido terminado de crear	Sí	
	El usuario donatario podrá editar un caso previamente creado	No	La funcionalidad no ha sido implementada por lo

			cual no se ha marcado a la historia como terminada.
	El usuario donante podrá visualizar las actualizaciones del caso en el tiempo	No	La funcionalidad no alcanzó a ser implementada a lo largo de la iteración.

Tabla 7- Evaluación de criterios de aceptación - Sprint 1

Incremento obtenido

El incremento que se obtuvo en el transcurso del sprint está relacionado a la implementación de las pantallas de inicio del sitio, autenticación, creación de cuenta de usuario donatario y creación de casos. Las ilustraciones de la 17 a la 21 muestran a breves rasgos las interfaces de usuario de las pantallas implementadas.

Ilustración 18 - Pantalla de bienvenida

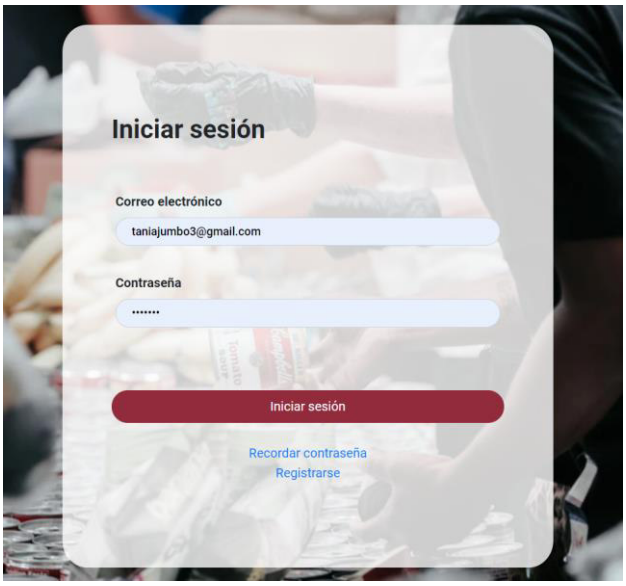

Ilustración 19 - Pantalla de inicio de sesión

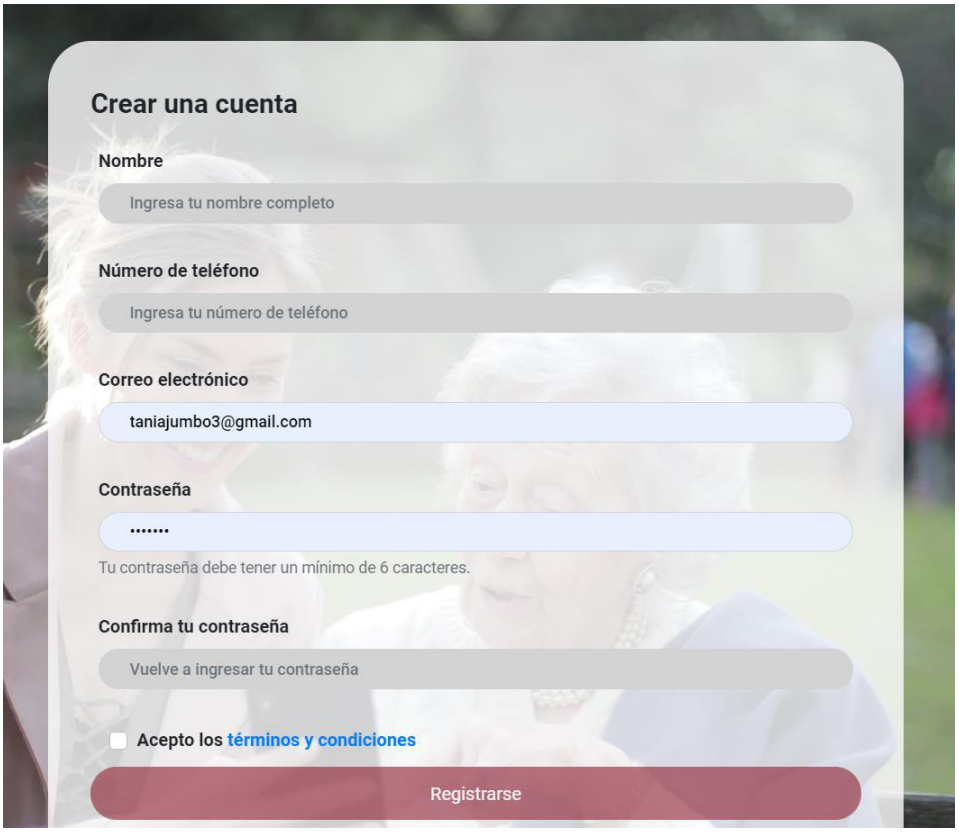

Ilustración 20 - Pantalla de creación de cuenta

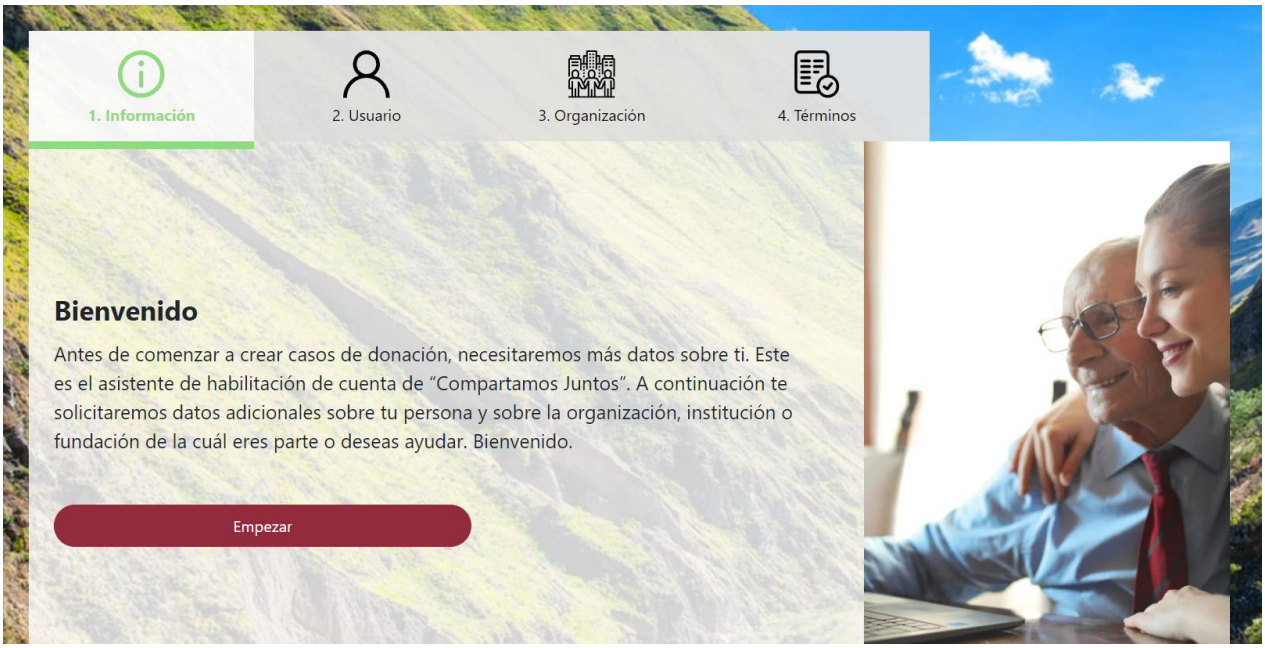

Ilustración 21 - Pantalla de solicitud de cuenta de usuario donatario

Adaptación del Product Backlog

El product backlog no tuvo cambios, la historia que no se terminó a lo largo del sprint 1 fue resuelta en el transcurso del sprint 2.

2.2.4 Sprint Retrospective

Puntos positivos: se identificaron los siguientes puntos positivos en el transcurso del sprint:

- Gran conocimiento de las herramientas utilizadas
- Enfoque a las metas del proyecto
- Compromiso con los criterios de aceptación

Puntos de mejora: se identificaron los siguientes puntos de mejora en el transcurso del sprint:

- Estimación del esfuerzo requerido para la implementación de las historias de usuario

Compromisos de mejora: se ha establecido los siguientes compromisos de mejora de cara a mejorar el desempeño:

- Mejorar la estimación del esfuerzo que conlleva implementar cada historia de usuario.

Burndown Chart: La ilustración 22 muestra el burndown chart en base al desempeño a lo largo del sprint:

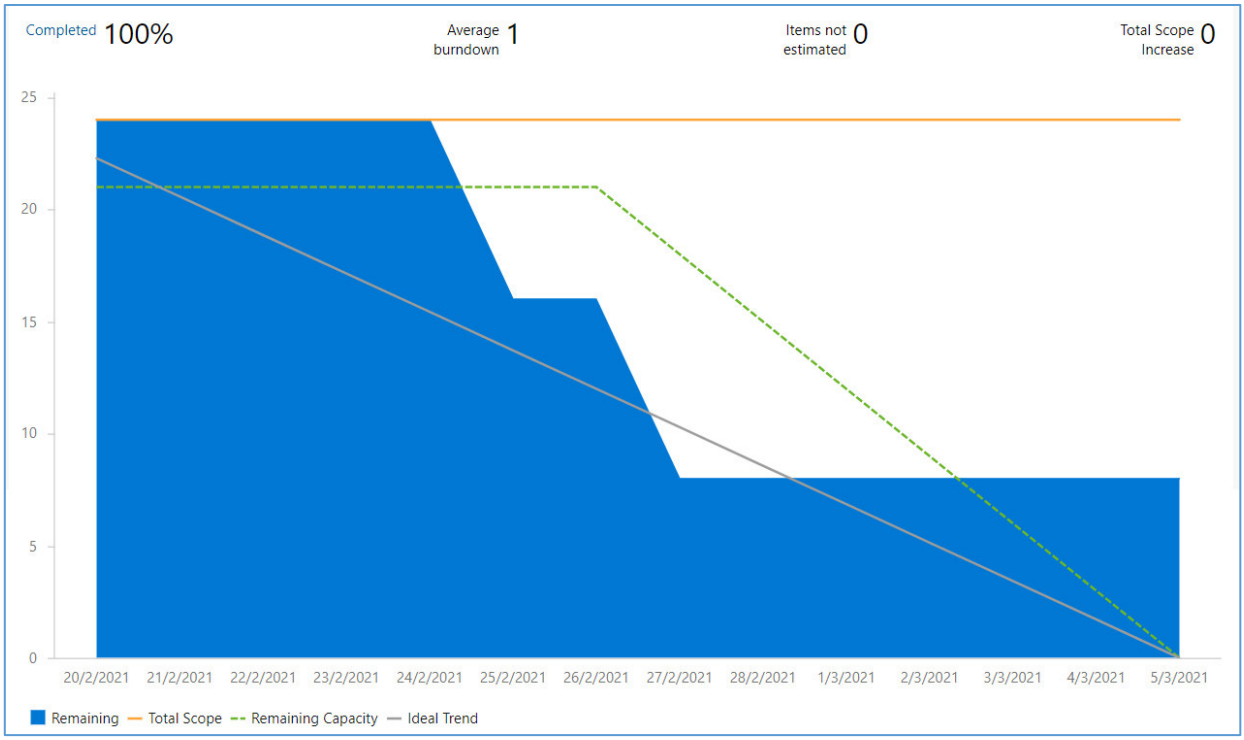

Ilustración 22 - Burndown Chart - Sprint 1

Análisis de tableros del equipo: como parte de la configuración del entorno DevOps se han establecido los siguientes tableros para analizar el desempeño del equipo:

- Cantidad de commits realizados: 27
- Cantidad de ejecuciones correctas de pipelines: 6
- Cantidad de ejecuciones incorrectas de pipelines: 6
- Porcentaje de pipelines exitosos: 50%

Lo cual refleja un porcentaje aceptable de construcciones por medio de pipelines teniendo en consideración la poca madurez del equipo al ser la primera iteración.

2.3 Sprint 2

2.3.1 Sprint Planning

Objetivo del Sprint

Implementar las funcionalidades relacionadas a la inspección de casos por parte del usuario donador. Completar la tarea de registro y administración de casos correspondientes a la historia USR010101 originalmente asignada al sprint 1. Al no existir errores registrados en las tareas completadas en la anterior iteración el objetivo del sprint no contempla reparación de funcionalidades defectuosas.

Historias de usuario

Las historias de usuario que fueron definidas para el Sprint 1 se muestran en las ilustraciones 23, 24 y 25 y podrán ser observadas a detalle en la sección de anexos.

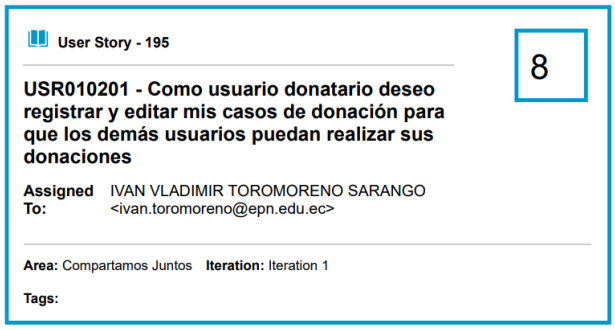

Ilustración 23 - Historia de usuario USR010101

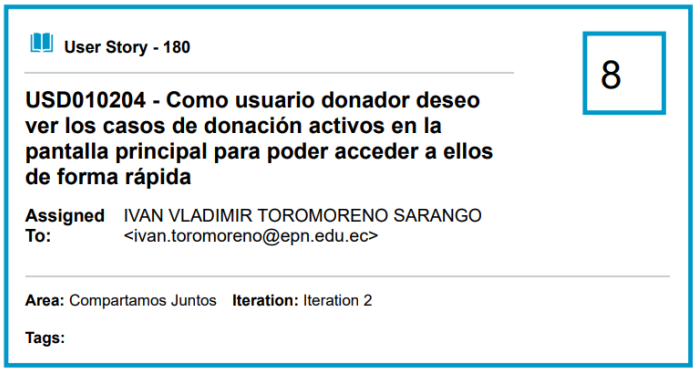

Ilustración 24 - Historia de usuario USD010204

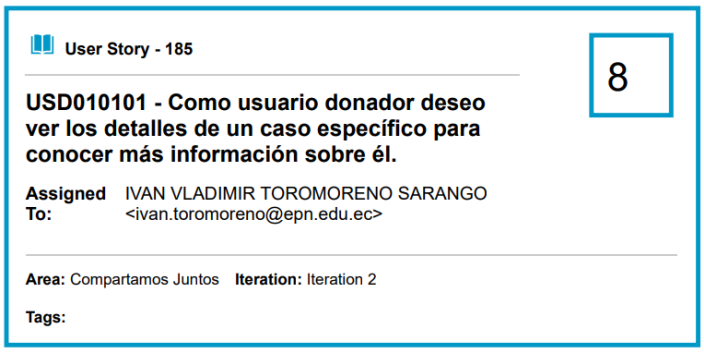

Ilustración 25 - Historia de usuario USD010101

Sprint backlog

La tabla 8 muestra las tareas definidas para las historias previamente definidas:

Order	Title
1	USD010204 - Como usuario donador deseo ver los casos de donación activos en la pan...
	Realizar diseño de la interfaz de usuario
	Conectar con firebase
2	USD010101 - Como usuario donador deseo ver los detalles de un caso específico ...
	Adecuar pantalla de detalle de caso para usuario donador

Tabla 8 - Backlog - Sprint 2

2.3.2 Ejecución del Sprint

Durante la ejecución del sprint las siguientes actividades fueron relevantes:

Creación de pantalla principal del sistema

La pantalla principal del sistema fue implementada en el transcurso del sprint como parte de la historia USD010204, cabe recalcar que, aunque la pantalla principal ya era funcional no contaba con todas las funcionalidades que se mostraban en pantalla, puesto a que estas serían implementadas en la siguiente iteración del producto. La ilustración 26 muestra de forma general la pantalla principal del sistema en dónde se pueden explorar los casos registrados por los usuarios donatarios.

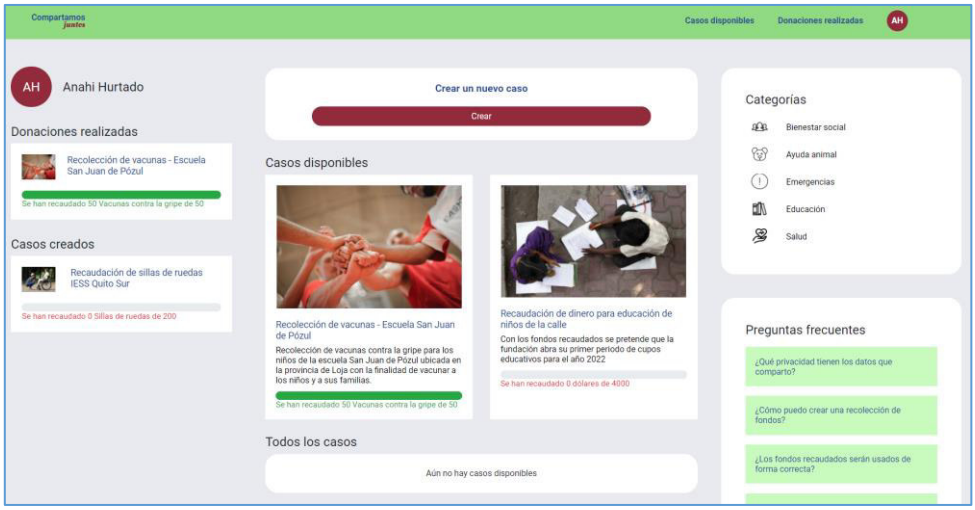

Ilustración 26 - Pantalla principal del sistema al finalizar el sprint 2

Visualización de casos por parte de los usuarios donantes y donatarios

La visualización de los casos por parte de los usuarios fue implementada haciendo uso de un mismo componente el cual modificaba su comportamiento en base a los parámetros recibidos en la ruta del controlador. La ilustración 27 muestra de forma general la pantalla de visualización de casos creados.

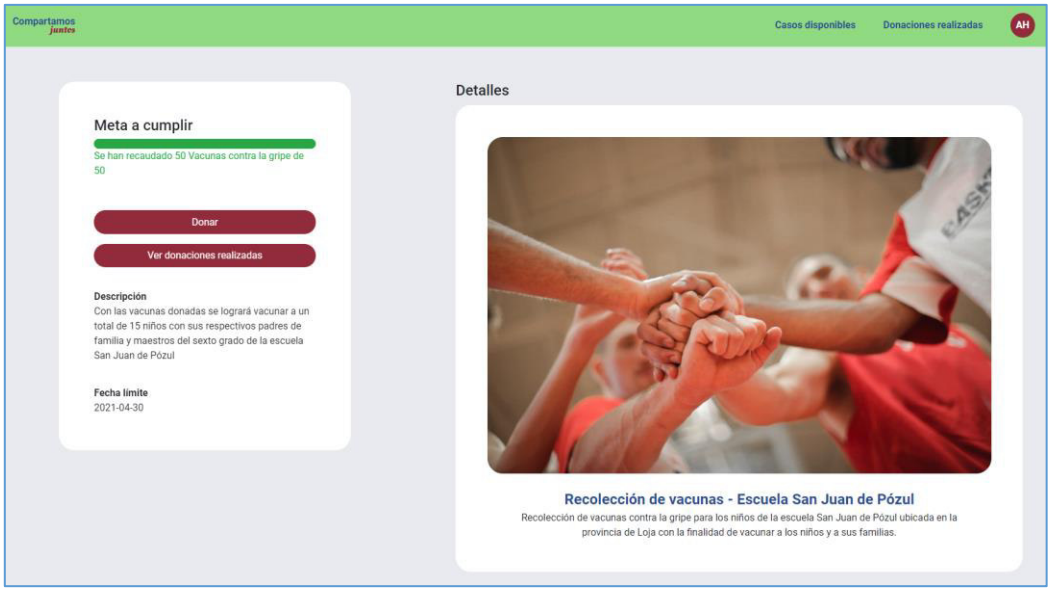

Ilustración 27 - Pantalla de visualización de casos al finalizar el sprint 2

2.3.3 Sprint Review

El objetivo planteado en el segundo sprint fue conseguido gracias al aprendizaje en la estimación de tareas que se tomó como compromiso en la iteración anterior, las historias definidas para el sprint fueron cumplidas en base al diseño planteado inicialmente y contaron con los criterios de aceptación necesarios para ser aceptadas. La tabla 9 muestra la evaluación realizada a las historias de usuarios implementadas en el sprint.

Historias de usuario	Criterios de aceptación	Cumple	Comentarios
USR010201	El usuario donatario podrá visualizar el formulario de creación de casos.	Sí	
	El usuario deberá ser informado de la correcta creación del caso y podrá visualizar los datos ingresados	Sí	
	El caso será visualizado por los usuarios donantes una vez que el caso haya sido terminado de crear	Sí	
	El usuario donatario podrá editar un caso previamente creado	Sí	
	El usuario donante podrá visualizar las actualizaciones del caso en el tiempo	Sí	
USD010204	El usuario observará en la pantalla principal un carrusel con los casos de donación activos en el sistema de la manera detallada en la descripción.	Sí	
USD010101	El usuario podrá acceder a la pantalla de detalles de un caso dando click en un caso que no haya donado previamente o en la opción de ver detalles en la pantalla de administración de un caso en el cual previamente haya contribuido.	Sí	
	Se mostrará una pantalla con los detalles detallados en la descripción.	Sí	

Tabla 9 - Evaluación de criterios de aceptación – Sprint 2

Incremento obtenido

El incremento obtenido resultante del sprint 2 fue la aceptación de las historias planteadas, dando como resultado las siguientes funcionalidades:

La administración de casos por parte del usuario donatario se terminó en su totalidad. El resultado es la interfaz de usuario mostrada en la ilustración 28 completamente funcional e interconectada con firebase.

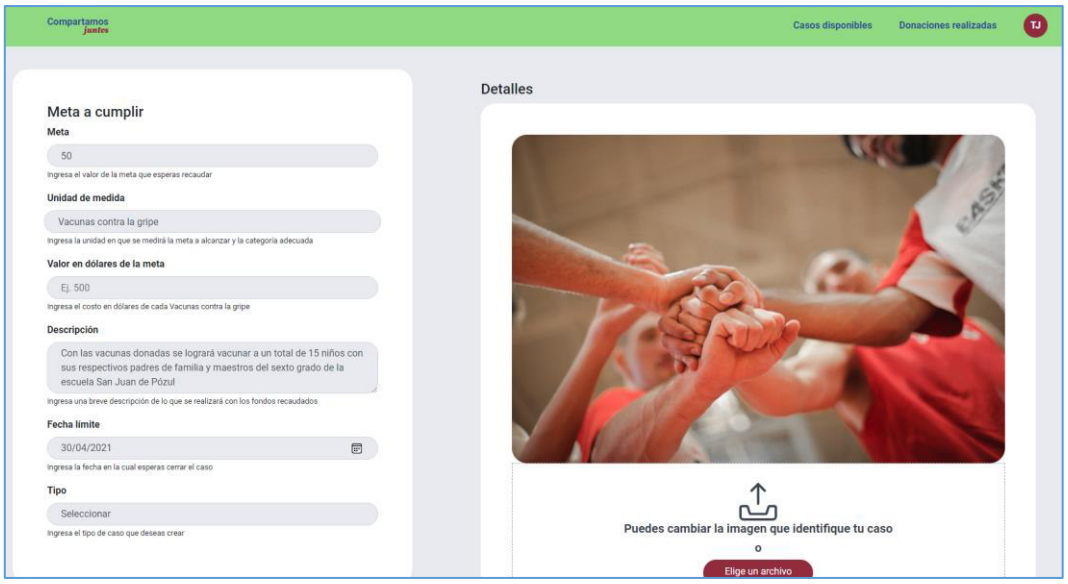

Ilustración 28 - Pantalla de registro y edición de casos

Adicionalmente se crearon las funcionalidades mostradas en las ilustraciones 26 y 27 correspondientes a la exploración de casos por parte del usuario donador. La ilustración 29 muestra los registros ingresados en la base de datos firestore como resultado de las pruebas realizadas en la etapa de desarrollo.

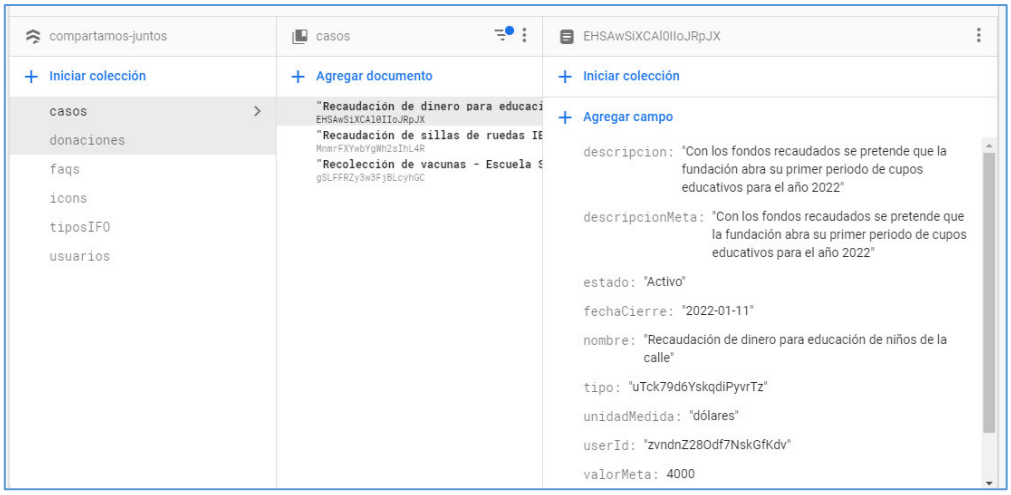

Ilustración 29 - Registros de casos ingresados en la base de datos firestore

Adaptación del Product Backlog

El product backlog cambió producto de la necesidad de contar con una pantalla de administración en el sistema, se ingresó una nueva historia de usuario que fue ejecutada en el sprint 3.

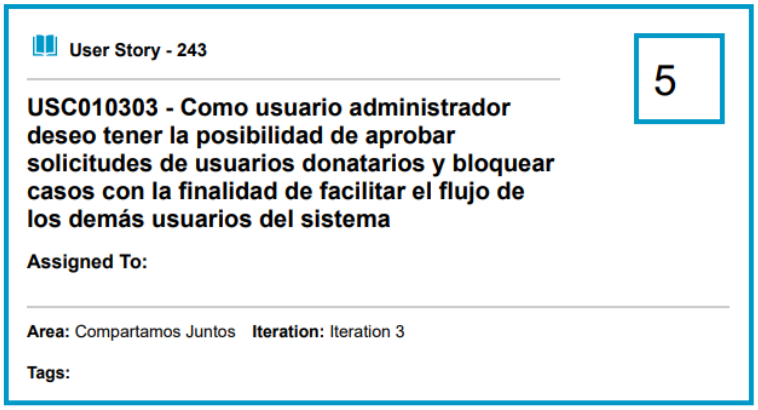

Ilustración 30 - Historia de usuario USC010303

2.3.4 Sprint Retrospective

Puntos positivos: se identificaron los siguientes puntos positivos en el transcurso del sprint:

- Mejor análisis en la complejidad de las tareas.
- Organización de tiempo para el cumplimiento de las metas.
- Compromiso para no arrastrar tareas a la siguiente iteración.

Puntos de mejora: se identificaron los siguientes puntos de mejora en el transcurso del sprint:

- Planificación de incidentes que pueda tener el equipo a lo largo del sprint.
- Revisión constante de pruebas unitarias del sistema.

Compromisos de mejora: se ha establecido los siguientes compromisos de mejora de cara a mejorar el desempeño:

- Contar con un plan de estimación que contemple las posibles incidencias que pueda tener el equipo de desarrollo a lo largo de la iteración.
- Diseñar un esquema de revisión de pruebas unitarias del sistema para evitar construcciones fallidas en el pipeline.

Burndown Chart: La ilustración 31 muestra el burndown chart en base al desempeño a lo largo del sprint:

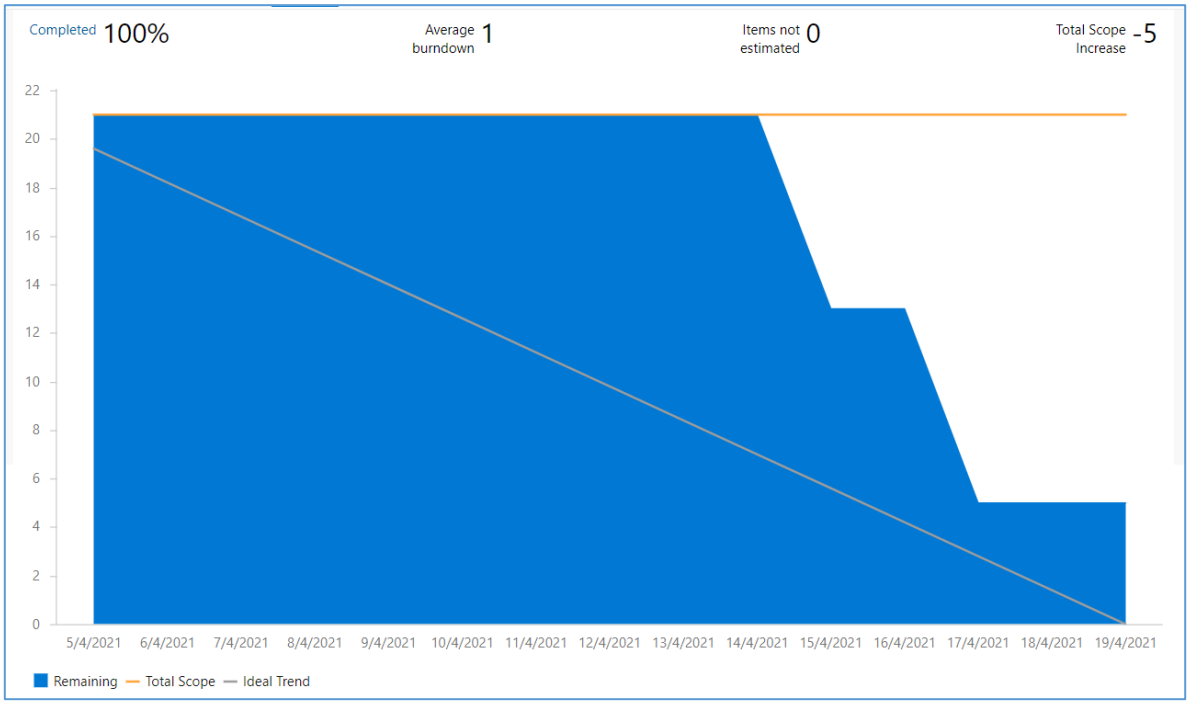

Ilustración 31 - Burndown Chart - Sprint 2

Análisis de tableros del equipo: los tableros de DevOps mostraron los siguientes resultados en base al desempeño del equipo:

- Cantidad de commits realizados: 21
- Cantidad de ejecuciones correctas de pipelines: 3
- Cantidad de ejecuciones incorrectas de pipelines: 6
- Porcentaje de pipelines exitosos: 33%

Esta información refleja las carencias en el esquema de pruebas unitarias que produjeron un bajo porcentaje de CI/CD exitoso a lo largo de la iteración.

2.4 Sprint 3

2.4.1 Sprint Planning

Objetivo del Sprint

Terminar la primera fase del prototipo de la aplicación, sus funcionalidades restantes y todos los requerimientos definidos para ser un mínimo producto viable. El tercer sprint fue el último sprint en integrar cambios funcionales al sistema y fue retrasado debido a complicaciones con el equipo de desarrollo.

Historias de usuario

Las historias de usuario que fueron definidas para el Sprint 1 se muestran en las ilustraciones de la 32 a la 43 y podrán ser observadas a detalle en la sección de anexos.

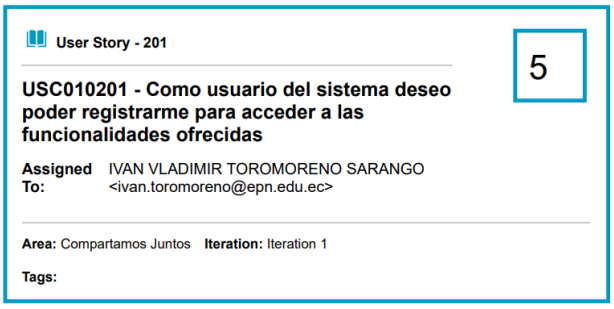

Ilustración 32 - Historia de usuario USC010201

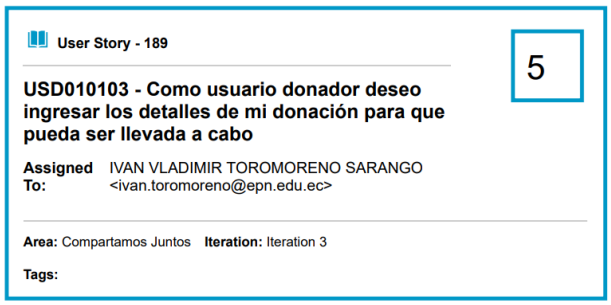

Ilustración 33 - Historia de usuario USD010103

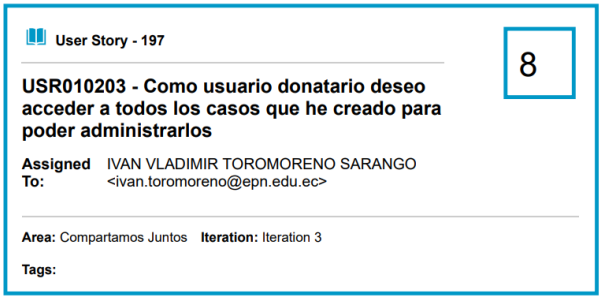

Ilustración 34 - Historia de usuario USR010203

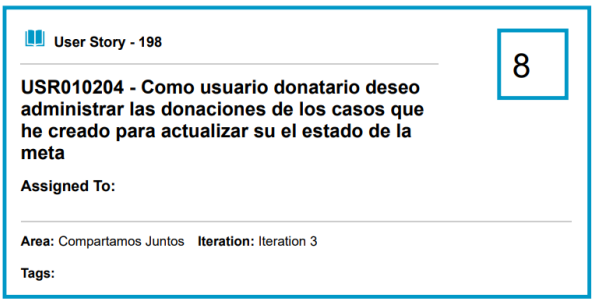

Ilustración 35 - Historia de usuario USR010204

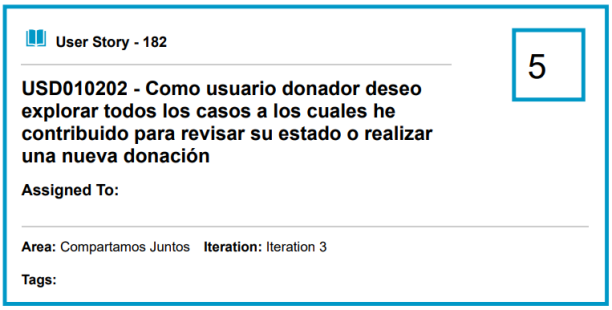

Ilustración 36 - Historia de usuario USD010202

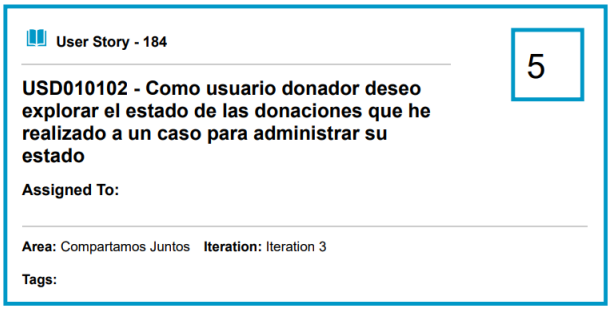

Ilustración 37 - Historia de usuario USD010102

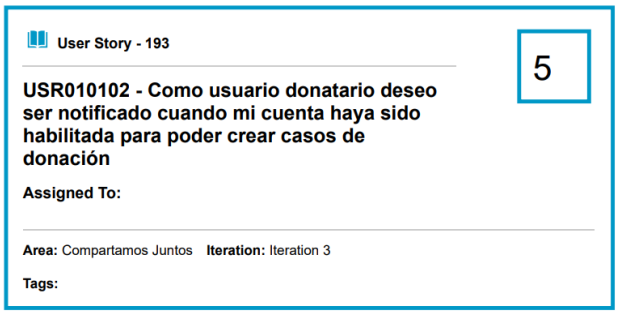

Ilustración 38 - Historia de usuario USR010102

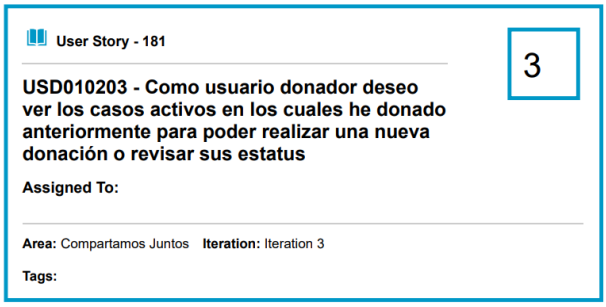

Ilustración 39 - USD010203

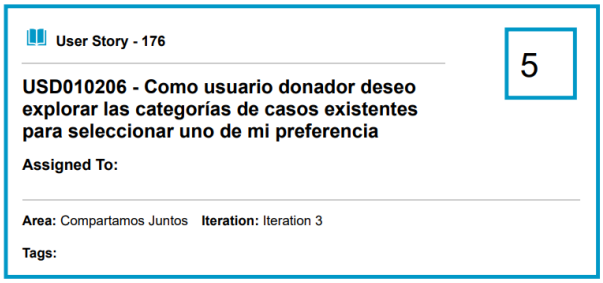

Ilustración 40 - USD010206

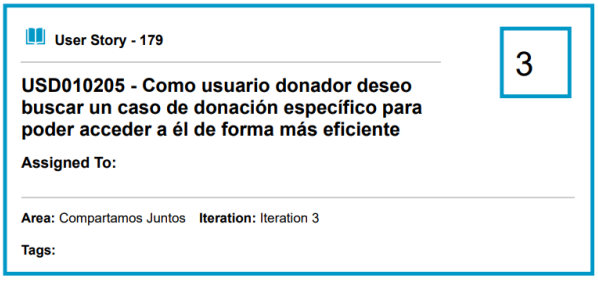

Ilustración 41 - USD010205

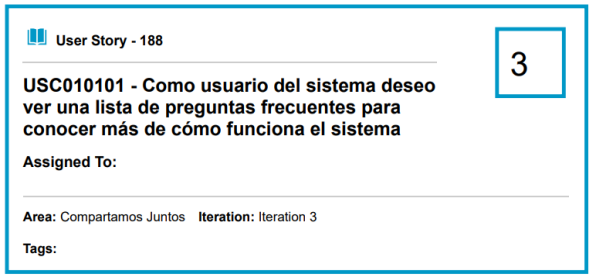

Ilustración 42 - USC010101

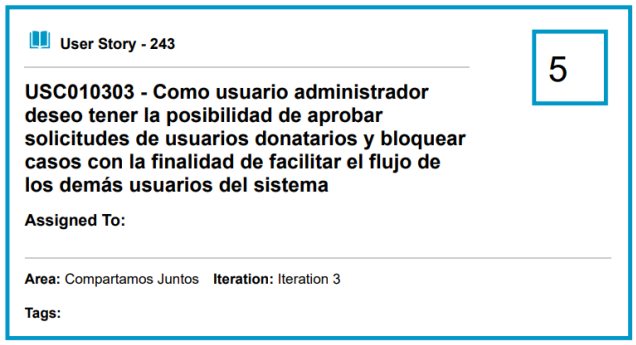

Ilustración 43 - USC010303

Sprint backlog

La tabla 10 muestra las tareas definidas para las historias previamente definidas:

Order	Title
1	USC010201 - Como usuario del sistema deseo poder registrarme para acceder a las fun...
	Al registrar usuario con datos de una cuenta ya existente, se muestra un mensaje no ...
2	USD010103 - Como usuario donador deseo ingresar los detalles de mi donación p... •••
	Diseño de la pantalla de donaciones
	Crear logica de creacion de donaciones
3	USR010203 - Como usuario donatario deseo acceder a todos los casos que he creado p...
	Insertar lista de donaciones recientes en pantalla principal junto con la opción de ing...
	Crear pantalla de donaciones realizadas junto con sus respectivos filtros según el est...
4	USR010204 - Como usuario donatario deseo administrar las donaciones de los casos q...
	Diseño de pantalla genérica de lista
	Adaptación de pantalla genérica a componentes específicos
	Programación de lógica de pantalla genérica
	Programación de lógica de pantalla específica
5	USD010202 - Como usuario donador deseo explorar todos los casos a los cuales he co...
	Modificar pantalla de lista genérica para mostrar las donaciones
6	USD010102 - Como usuario donador deseo explorar el estado de las donaciones que h...
	Configuración de pantalla genérica
	Extracción de datos de donaciones
7	USR010102 - Como usuario donatario deseo ser notificado cuando mi cuenta haya sido...
	Creación de tarjetas de notificación
	Configuración de despliegue de tarjetas
8	USD010203 - Como usuario donador deseo ver los casos activos en los cuales he dona...
	Mostrar lista de donaciones en pantalla principal
9	USD010206 - Como usuario donador deseo explorar las categorías de casos existentes ...
	Modificación de panel derecho con categorías de casos
	Pantalla de casos con filtros (Se selecciona automaticamente el filtro seleccionado pr...
10	USD010205 - Como usuario donador deseo buscar un caso de donación específico par...
	Implementación de pantalla de búsqueda
11	USC010101 - Como usuario del sistema deseo ver una lista de preguntas frecuentes pa...
	Creación de componente FAQ
	Creación de FAQs en firebase
12	USC010303 - Como usuario administrador deseo tener la posibilidad de aprobar s... •••
	Creación de pantalla de administración
	Conexión firebase

Tabla 10 – Backlog – Sprint 3

2.4.2 Ejecución del Sprint

Durante la ejecución del sprint las siguientes actividades fueron relevantes:

Conclusión del prototipo.

El prototipo contiene las funcionalidades planteadas en las historias de usuario y cuenta con la arquitectura planeada inicialmente. Los módulos de donación y administración al finalizar el sprint no fueron sometidos a pruebas funcionales por un equipo de QA externo. Las pruebas sobre los flujos de trabajo fueron realizadas por el equipo de desarrollo y no se reportaron incidencias que puedan afectar al correcto desenvolvimiento de los usuarios en la aplicación.

Módulo de administración del sistema

El módulo de administración del sistema fue incorporado al backlog del sistema al finalizar el sprint número 2, su funcionalidad cumple con los requerimientos planteados en la historia de usuario, la ilustración 44 muestra la pantalla que podrá visualizar un usuario administrador al ingresar al sistema.

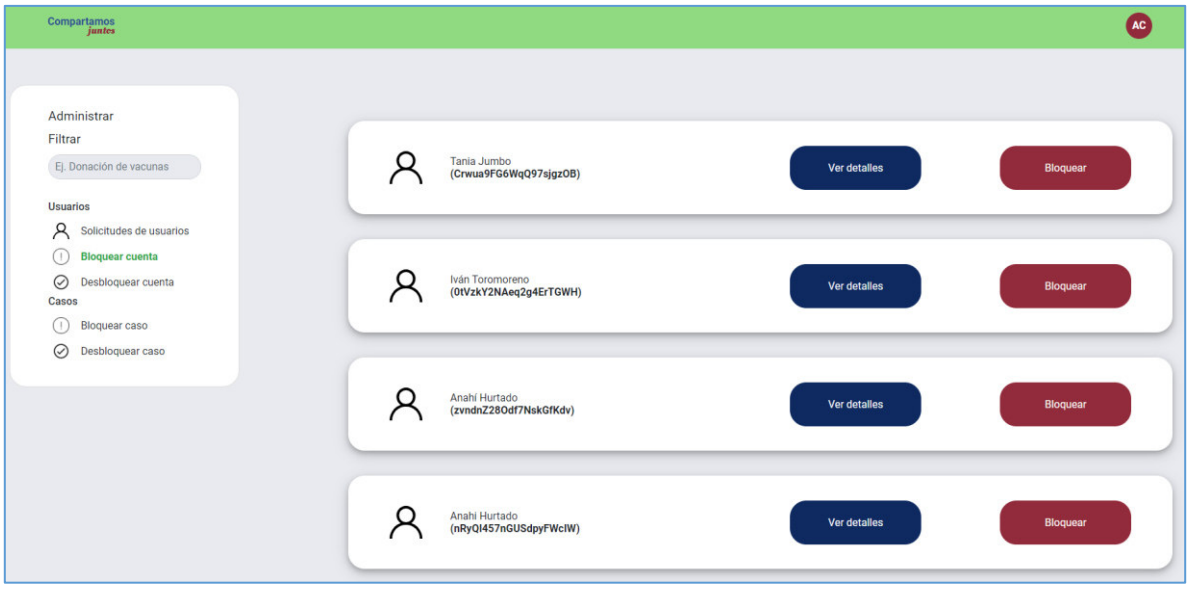

Ilustración 44 - Pantalla de administración del sistema

Módulo de donaciones

El módulo de donaciones fue otra de las funcionalidades clave para la finalización del prototipo ya que el núcleo de la aplicación está contenido dentro de esta funcionalidad. Los métodos de donación permitidos en la aplicación son los siguientes:

- Donación de bienes por medio de contacto con el donatario.
- Donación de recursos económicos por medio de contacto con el donatario.
- Donación de recursos económicos por medio de un botón de pagos.

La modalidad de donación de recursos económicos por medio de un botón de pagos es un prototipo de una pantalla de pagos, sin embargo, es necesario recalcar que esta funcionalidad al no estar contenida dentro del alcance inicial del proyecto no se encuentra conectada a un proveedor real de este servicio como puede ser “kushky”. La ilustración 45 muestra el prototipo de la pantalla de donaciones por medio del botón de pagos en donde se resalta que la información ingresada no debe ser información real.

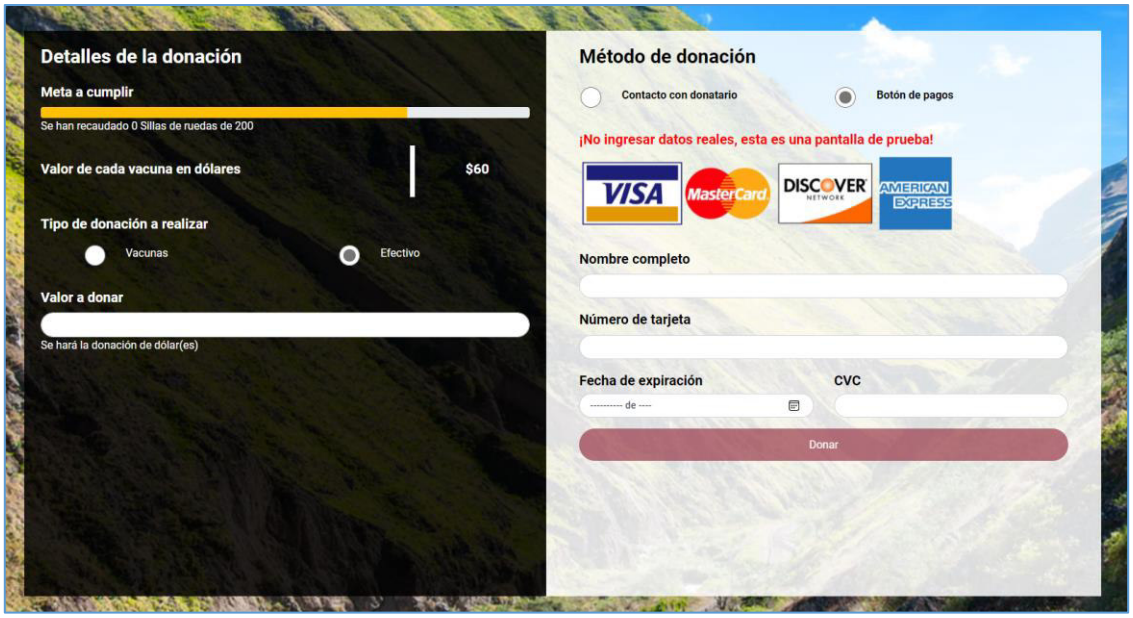

Ilustración 45 - Pantalla de donaciones por medio de botón de pagos

Administración de donaciones

Los usuarios donadores y donatarios del sistema podrán administrar los casos de donación en los que hayan estado relacionados, cada uno desde una perspectiva distinta. Los

usuarios donadores podrán verificar el estado de las donaciones realizadas y los usuarios donatarios podrán cambiar el estado de las donaciones según se considere necesario. La ilustración 46 muestra la pantalla de administración de donaciones desde la perspectiva de un usuario donatario.

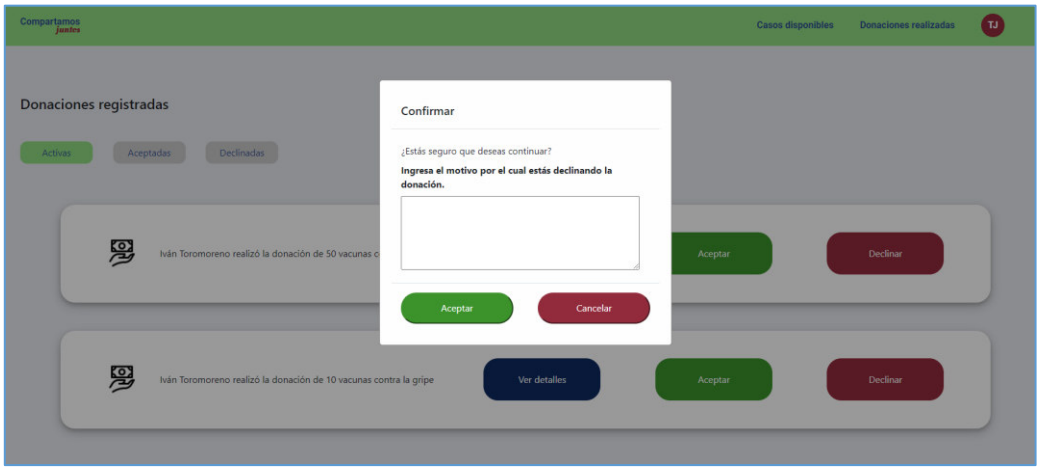

Ilustración 46 - Pantalla de administración de donaciones - Usuario donatario

2.4.3 Sprint Review

El objetivo planteado en el tercer sprint fue conseguido, pese a que el equipo de desarrollo tuvo que extender la duración de la iteración, las metas fueron conseguidas y las historias de usuario pasaron todos los criterios de aceptación establecido dejando al sistema en un estado listo para pruebas funcionales. La tabla 11 muestra la evaluación realizada a las hitorias de usuarios implementadas en el sprint.

Historias de usuario	Criterios de aceptación	Cumple	Comentarios
USC010201	Se deberá mostrar el formulario de registro al dar click en la opción de registrarse en la pantalla inicial del sistema.	Sí	
	Se deberá validar que la información ingresada corresponda a lo detallado en la descripción	Sí	
	Se deberá indicar al usuario que su cuenta ha sido creada	Sí	
	Se deberá notificar al usuario por correo electrónico que su cuenta debe ser validada para poder acceder	Sí	

Historias de usuario	Criterios de aceptación	Cumple	Comentarios
USD010103	Se mostrará un formulario en donde se detallará que la información del usuario será compartida con el donatario y se permitirá ingresar la información detallada en la descripción.	Sí	
USR010203	Se mostrará un botón de acceso a la exploración de casos de donación en el menú de perfil de usuario	Sí	
	Se mostrará una pantalla con los casos creados por el donatario, dando la posibilidad de filtrar los casos según su estado y de buscar un caso específico	Sí	
USR010204	El usuario accederá a la pantalla de administración del caso al dar click en una tarjeta de un caso creado por él.	Sí	
	Se podrá realizar las acciones detalladas en la descripción	Sí	
USD010202	El usuario podrá acceder a la lista de casos a los cuales ha contribuido mediante el menú perfil que se despliega al hacer click en el avatar ubicado en la barra de navegación de la aplicación	Sí	
	El usuario podrá filtrar los casos en los cuales ha contribuido	Sí	
	El usuario podrá buscar un caso específico entre los cuales ha contribuido	Sí	
USR010102	El usuario podrá administrar los casos como está detallado en la descripción al dar click en un caso en el cual ha contribuido previamente.	Sí	
USD010203	En la pantalla principal se mostrará un mensaje en la parte superior indicando el estado de postulación del usuario, siendo las siguientes opciones:	Sí	
	Postularse como donatario (Cuando el usuario no haya presentado una solicitud para ser donatario)	Sí	
	Tu cuenta está siendo validada (Cuando se haya presentado una solicitud de ser donatario, pero esta aún no haya sido aprobada)	Sí	
	Tu cuenta ha sido habilitada (Cuando luego del proceso de validación se haya aprobado la cuenta del usuario para crear casos)	Sí	

	Crear un nuevo caso (Cuando el usuario haya creado un caso previamente)	Sí	
Historias de usuario	**Criterios de aceptación**	**Cumple**	**Comentarios**
USD010206	Se mostrará una lista de casos en la parte izquierda de la pantalla principal con los casos a los cuales el usuario ha contribuido previamente.	Sí	
	El usuario deberá visualizar en la pantalla principal y en el explorador de casos las categorías disponibles en el sistema y seleccionar las de su preferencia.	Sí	
	Al dar click en una categoría se abrirá el explorador de casos con la categoría seleccionada por el usuario.	Sí	
USD010205	Se mostrará un buscador de casos tanto en la pantalla de inicio del sistema como en la pantalla de exploración de casos.	Sí	
	Al ingresar un nombre o identificador de caso y presionar el botón de búsqueda, el usuario podrá observar una lista de casos que puedan satisfacer su búsqueda.	Sí	
USC010101	Se mostrará una lista de preguntas frecuentes en la parte izquierda de la pantalla principal del sistema, al seleccionar una, se desplegará la respuesta a la pregunta en la parte inferior de la misma.	Sí	
USC010303	Dado que el usuario administrador se encuentre en la pantalla de administración, cuando se realice el bloqueo de una cuenta, el usuario no podrá crear nuevos casos de donación.	Sí	
	Dado que el usuario administrador se encuentre en la pantalla de administración, cuando se realice el bloqueo de un caso, los demás usuarios del sistema no podrán volver a visualizar el caso en las pantallas de exploración.	Sí	
	Dado que el usuario administrador se encuentre en la pantalla de administración, cuando se habilite una cuenta de donatario, el usuario tendrá la capacidad de crear nuevos casos de donación.	Sí	

	Dado que el usuario administrador se encuentre en la pantalla de administración, cuando se realice el desbloqueo de una cuenta, el usuario podrá crear nuevos casos de donación.	Sí	
	Dado que el usuario administrador se encuentre en la pantalla de administración, cuando se realice el desbloqueo de un caso, los demás usuarios podrán volver a visualizar el caso en las pantallas de exploración del sistema.	Sí	

Tabla 11 - Evaluación de criterios de aceptación - Sprint 3

Incremento obtenido

El incremento del sistema obtenido a lo largo del sprint consiste en la finalización de la implementación de las funcionalidades planteadas como alcance para el prototipo inicial de la aplicación. Las funcionalidades implementadas a lo largo de la iteración fueron:

- Módulo de donaciones:
    - Pantalla de donaciones por medio de los métodos definidos en la historia de usuario
    - Pantalla de administración de donaciones realizadas
    - Pantalla de exploración de casos disponibles con funcionalidades de:
        - Filtrado por nombre de caso
        - Filtrado por categoría de caso
- Módulo de donatarios:
    - Pantalla de administración de donaciones recibidas desde donde el usuario podrá:
        - Aceptar las donaciones
        - Declinar las donaciones
        - Consultar información a las donaciones recibidas.
- Módulo de administración:
    - Pantalla de administración general del sistema desde donde el usuario administrador podrá:
        - Bloquear / desbloquear casos y donaciones
        - Aprobar solicitudes de usuarios donatarios

La ilustración 47 muestra la pantalla de exploración de casos que permitirá a los usuarios donadores visualizar los casos según sus intereses particulares según el nombre del caso o la categoría a la cual pertenece.

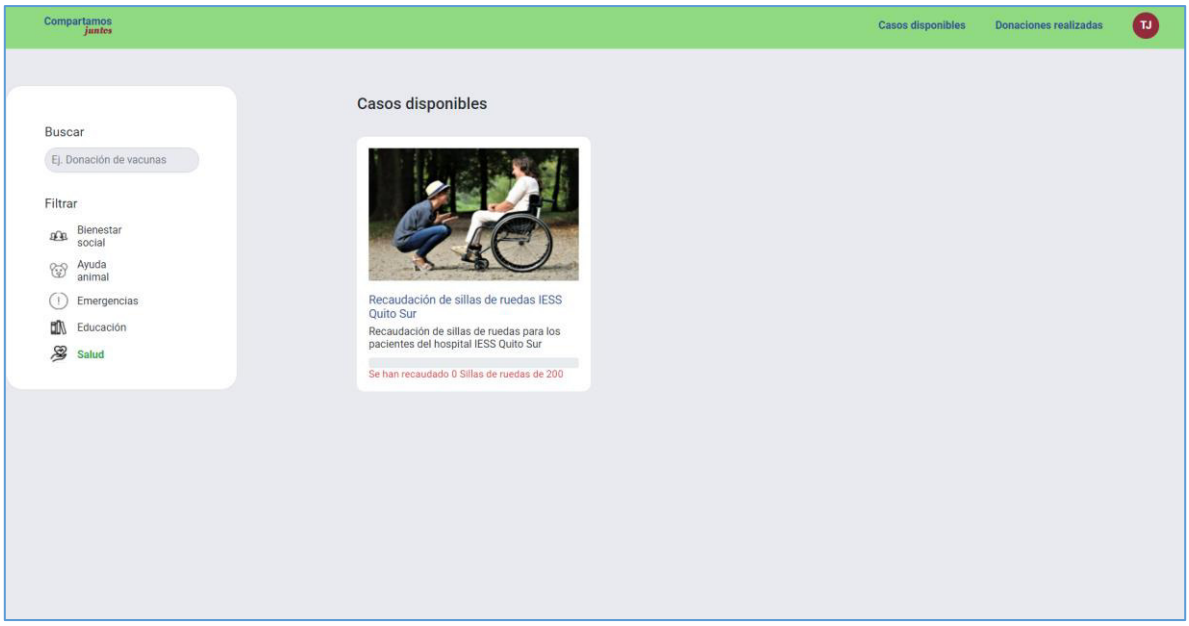

Ilustración 47 - Pantalla de exploración de casos de donación

2.4.4 Sprint Retrospective

Puntos positivos: se identificaron los siguientes puntos positivos en el transcurso del sprint:

- Compromiso del equipo de desarrollo con el cumplimiento del objetivo definido.
- Agilidad en la implementación de las historias de usuario.
- Modularización de componentes para hacer el sistema más robusto.

Puntos de mejora: se identificaron los siguientes puntos de mejora en el transcurso del sprint:

- Definición del esquema de pruebas unitarias para evitar daños en el pipeline de CI/CD

Compromisos de mejora: se ha establecido los siguientes compromisos de mejora de cara a mejorar el desempeño:

- Correr las pruebas unitarias en un entorno local antes de incorporar los cambios al ambiente de pruebas del equipo.

Burndown Chart: La ilustración 48 muestra el burndown chart en base al desempeño a lo largo del sprint:

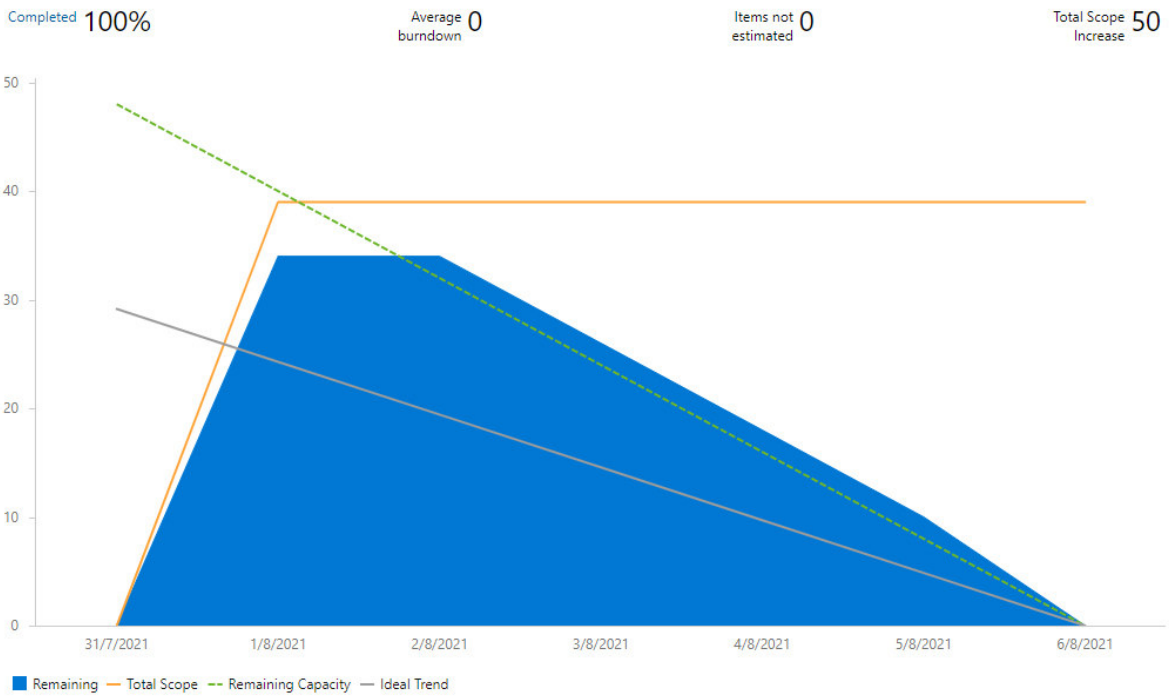

Ilustración 48 - Burndown Chart - Sprint 3

Análisis de tableros del equipo: los tableros de DevOps mostrados en las figuras 49 y 50 corresponden a la etapa final de desarrollo del proyecto y reflejan la siguiente información:

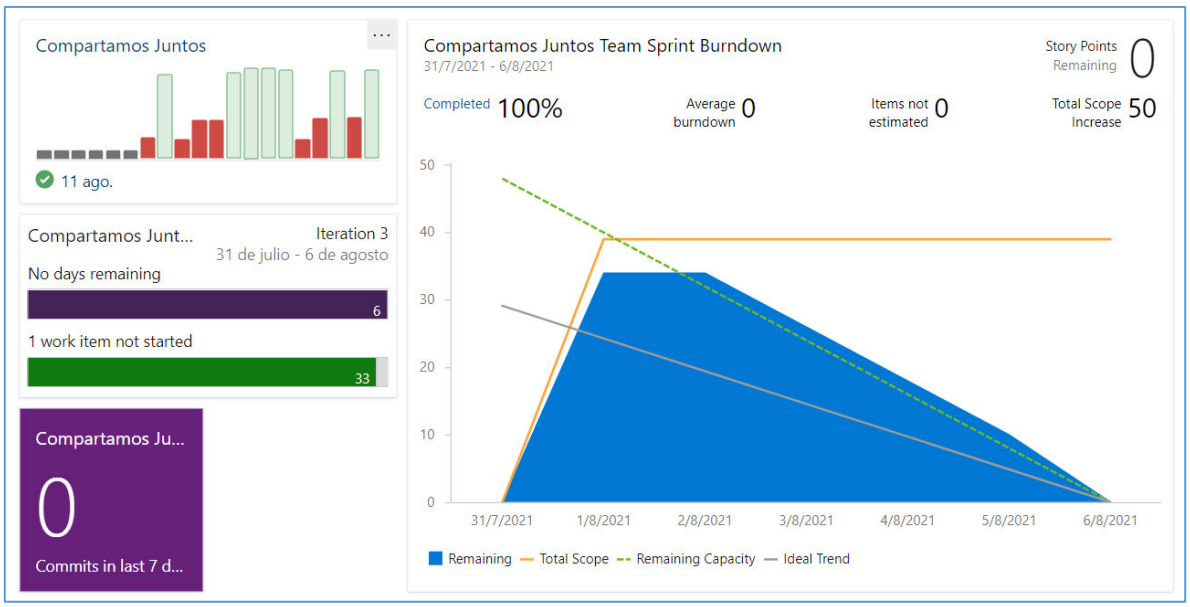

Ilustración 49 - Tableros de desempeño del equipo - Sprint 3 - 1

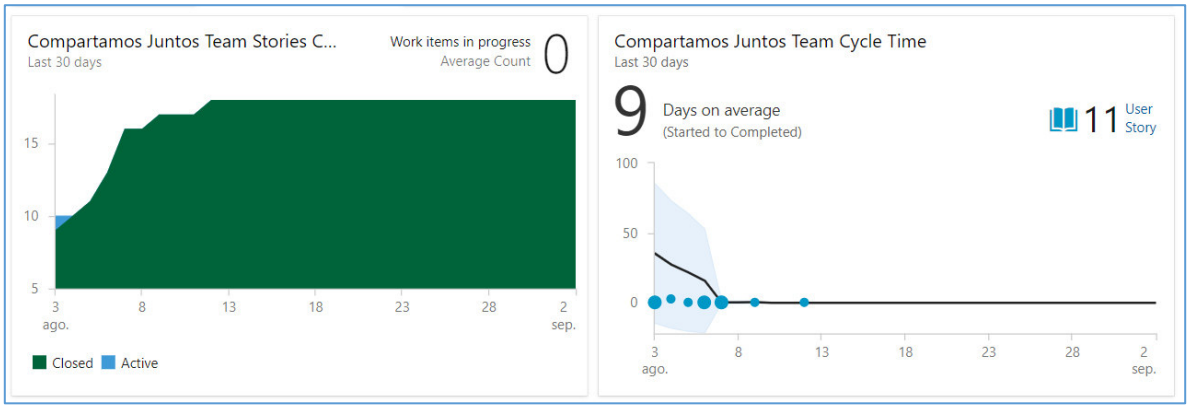

Ilustración 50 - Tableros de desempeño del equipo - Sprint 3 – 2

Los tableros reflejan el aprendizaje por parte del equipo en términos de ejecución de pruebas y de construcción de pipelines ya que el porcentaje de construcciones exitosas es claramente superior al de construcciones fallidas. Los tableros también indican el compromiso del equipo en el cumplimiento de las metas logrando aumentar la productividad de una forma considerable en relación con iteraciones pasadas.

2.5 Sprint 4

2.5.1 Sprint Planning

Objetivo del Sprint

El presente sprint fue definido para corrección de fallos en el sistema con ayuda de pruebas funcionales realizadas a un sujeto de pruebas externo al equipo de desarrollo por lo cual no constó con todas las ceremonias definidas por Scrum ni tampoco se generaron todos los entregables recomendados. La decisión de ejecutar el sprint bajo este esquema se debe a que el sujeto de pruebas no contaba con acceso a la plataforma Azure DevOps y los casos de prueba junto con las incidencias reportadas fueron manejados por medio de la herramienta externa llamada Trello.

Definición de pruebas funcionales del sistema

La ilustración 51 muestra de forma general las pruebas funcionales definidas sobre los flujos del sistema junto con el flujo que se definió para dar cada prueba como terminada y aceptada. Dentro de cada tarjeta creada en la herramienta se reportaron las incidencias que aparecieron en el período de evaluación, mismas que fueron reportadas al equipo de desarrollo y corregidas a lo largo de la iteración e incorporadas al sistema final por medio de los pipelines configurados en el repositorio del proyecto.

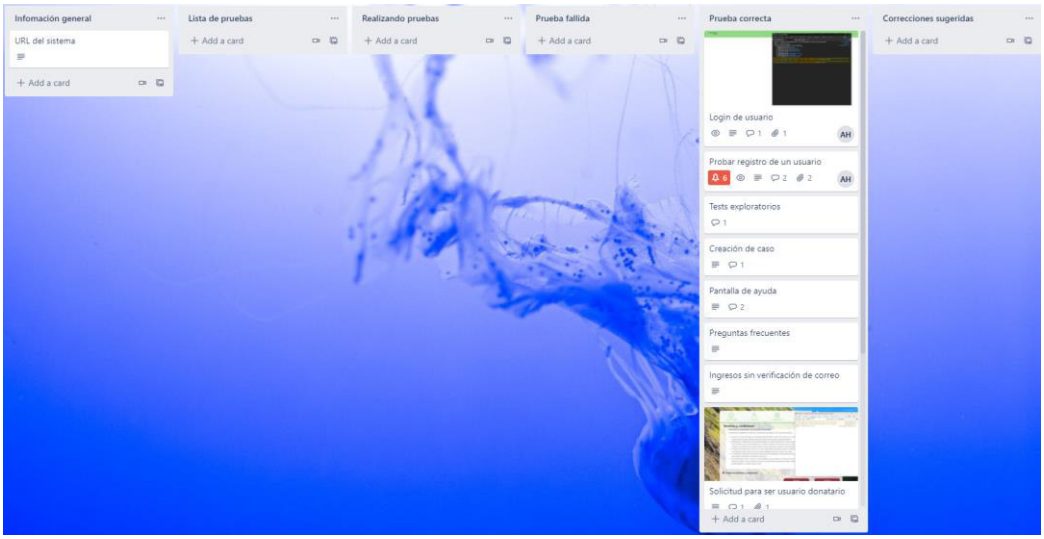

Ilustración 51 - Pruebas funcionales del sistema.

2.5.2 Ejecución y cierre del sprint

En la ejecución del sprint se corrigieron los errores reportados por el sujeto de pruebas. Los cambios realizados al aplicativo fueron incorporados de forma automática gracias al pipeline de CI/CD configurado para el proyecto. El sujeto de pruebas iteró varias veces sobre los casos de pruebas con la finalidad de comprobar que las nuevas implementaciones no afectaran a los flujos previamente aceptados.

El resultado del sprint fue un sistema funcionalmente aceptable por un sujeto de pruebas externos al equipo el cual no cuenta con experiencia en el desarrollo de sistemas informáticos y cuya experiencia de uso con plataformas web es el de un usuario final del sistema promedio.

Las acciones previamente comprometidas por el equipo para la mejora del desempeño fueron adoptadas de tal manera que no se produjeron fallos en el CI/CD del proyecto como resultado de mala configuración en las pruebas unitarias.

3 RESULTADOS Y DISCUSIÓN

3.1 Producto final

La aplicación final del proyecto planteado contiene los módulos y funcionalidades definidos inicialmente los cuales son:

- Módulo de donaciones / donadores:
    - Creación de donaciones
    - Administración de donaciones
- Módulo de donatarios:
    - Postulación como usuario donatario
    - Creación de casos de donación
    - Administración de casos de donación
- Módulo general del sistema:
    - Autenticación y creación de usuarios
    - Administración general del sistema

La ilustración 52 muestra de forma general el diseño del sistema en donde se pueden visualizar las definiciones antes descritas.

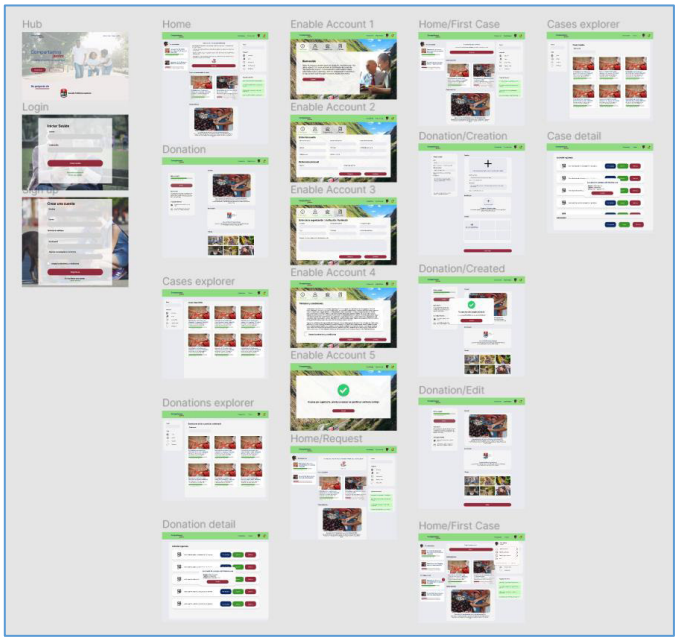

Ilustración 52 - Diseño general UI/UX de la aplicación

3.1.1 Modulo general de la aplicación

Este módulo contiene funcionalidades generales del aplicativo que servirán de soporte para los demás módulos definidos.

Las funcionalidades principales constan de los siguientes flujos:

3.1.1.1 Pantalla de inicio del sistema

La pantalla de inicio del sistema muestra las opciones de autenticación para los usuarios, además ofrece acceso al módulo de ayuda del sistema. La ilustración 53 muestra a detalle la pantalla con las funcionalidades previamente descritas.

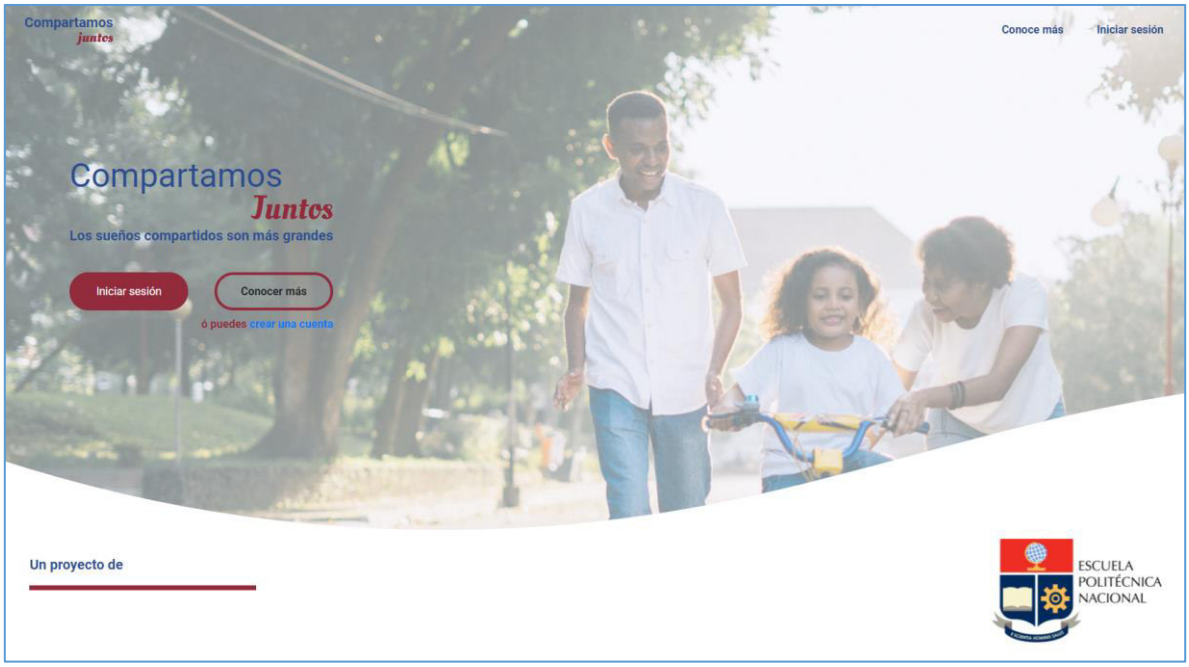

Ilustración 53 - Versión final de la pantalla de inicio

3.1.1.2 Pantalla de ayuda del sistema

La pantalla de ayuda del sistema contiene información relevante para los usuarios registrados, los términos y condiciones de la plataforma, las tecnologías utilizadas y finalmente el equipo de desarrollo involucrado en la construcción de la aplicación junto con las herramientas principales utilizadas para la construcción del sitio. La ilustración 54 muestra una parte de la pantalla de ayuda del sistema.

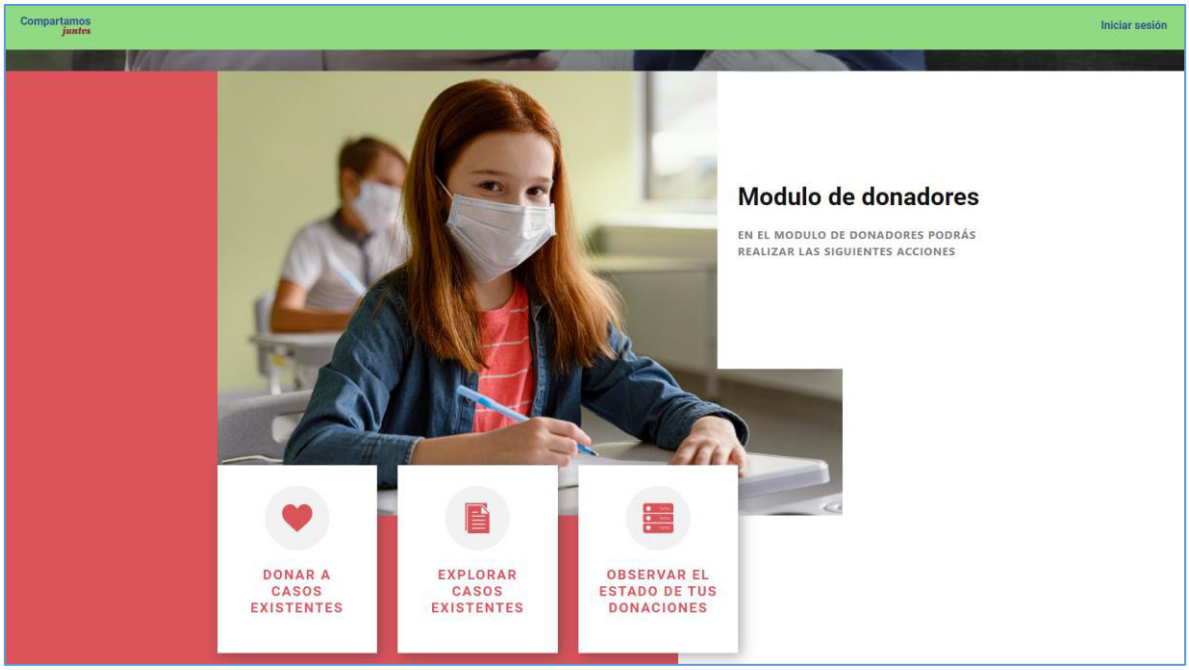

Ilustración 54 - Versión final de la pantalla de ayuda

3.1.1.3 Pantallas de registro y autenticación

Las pantallas de registro y autenticación del sistema son el inicio del flujo de trabajo de los usuarios dentro de la plataforma, la información requerida inicialmente no consta de información sensible de los usuarios y permite tener un contexto de su identidad dentro de la plataforma. Las ilustraciones 55 y 56 muestran las pantallas implementadas en el sistema final.

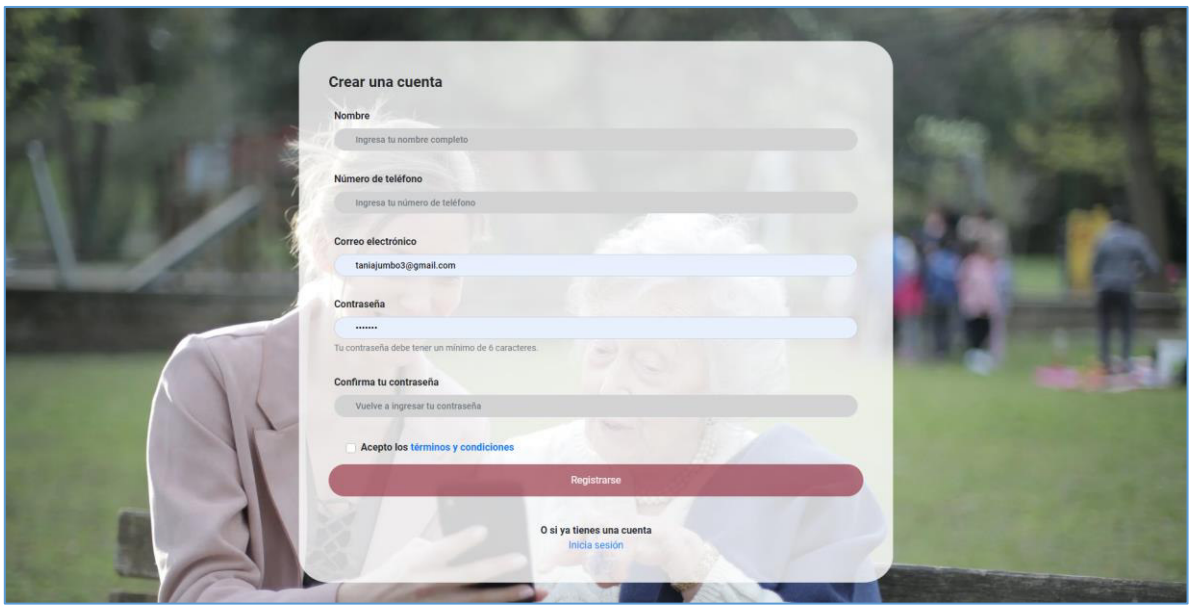

Ilustración 55 - Pantalla final de registro

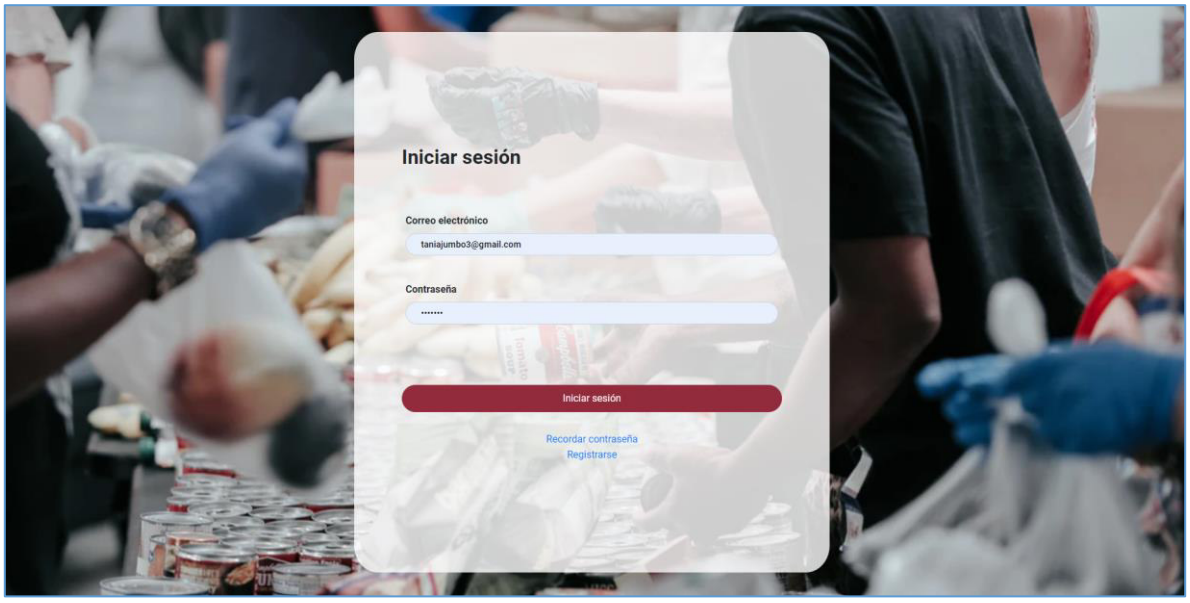

Ilustración 56 - Pantalla final de inicio de sesión

3.1.1.4 Pantalla de administración del sistema

El sistema cuenta con un módulo de administración el cual permite a un usuario definido con dicho rol realizar las acciones previamente definidas. La ilustración 57 muestra la pantalla de administración en donde se puede observar las acciones que se pueden realizar sobre los casos activos en el sistema.

Ilustración 57 - Pantalla de administración - administración de casos

Dentro de la pantalla de bloqueo de casos se puede bloquear el acceso a un caso por parte de otros usuarios en caso de que se detecte actividad irregular dentro del mismo. Al momento de intentar realizar un bloqueo el sistema solicitará información adicional para poder continuar.

La ilustración 58 muestra la pantalla de bloqueo de usuarios, que al igual que la pantalla de bloqueo de casos permite restringir la creación de casos a usuarios donatarios que muestren acciones sospechosas dentro del sistema. Al momento de intentar realizar un bloqueo el sistema solicitará información adicional para poder continuar.

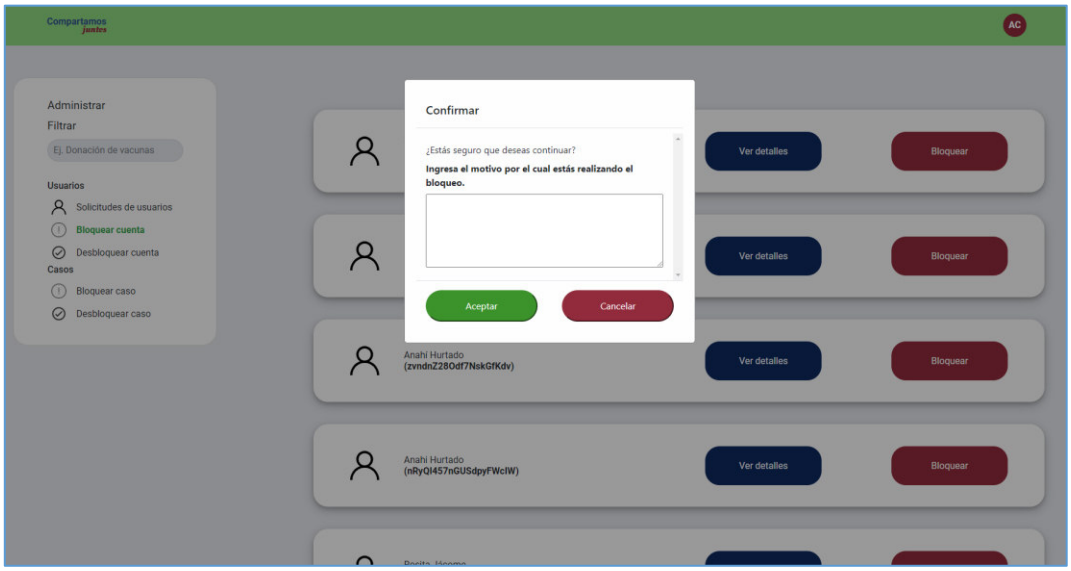

Ilustración 58 - Pantalla de administración - administración de usuarios

3.1.1.5 Pantalla principal del sistema

La pantalla principal del sistema contiene información sobre los casos creados en el sistema. Los usuarios donadores y donatarios podrán explorar casos en los cuales pueden realizar donaciones, así como administrar los casos que han sido creados por ellos mismos. Adicionalmente dependiendo del rol del usuario se mostrará la opción de solicitar una cuenta de usuario donatario o los distintos estados en los cuales se encuentra su solicitud. Finalmente se brinda una sección de preguntas frecuentes en donde se solventarán las dudas que se consideraron adecuadas tener de fácil acceso por parte de los usuarios. La figura 59 muestra la pantalla principal del sistema desde el punto de vista de un usuario donatario.

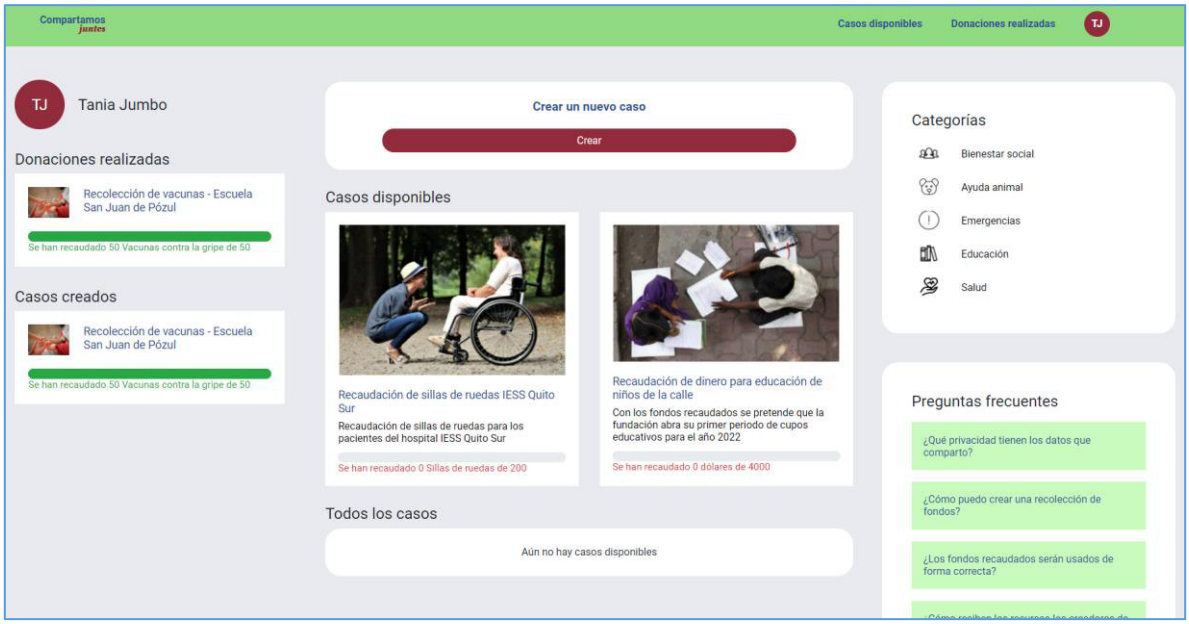

Ilustración 59 - Pantalla principal del sistema - Usuario donatario

3.1.2 Módulo de donadores

Los usuarios donadores tendrán tres funcionalidades principales dentro del sistema, las cuales son:

- Explorar los casos disponibles
- Crear donaciones a los casos existentes
- Observar el estado de las donaciones realizadas

3.1.2.1 Exploración de casos disponibles

Los casos disponibles pueden ser accedidos mediante la pantalla principal del sistema (Ilustración 59) o desde la pantalla de exploración de casos mostrada en la ilustración 60. La pantalla de exploración de casos permite a los usuarios filtrar los casos por categorías o por nombres.

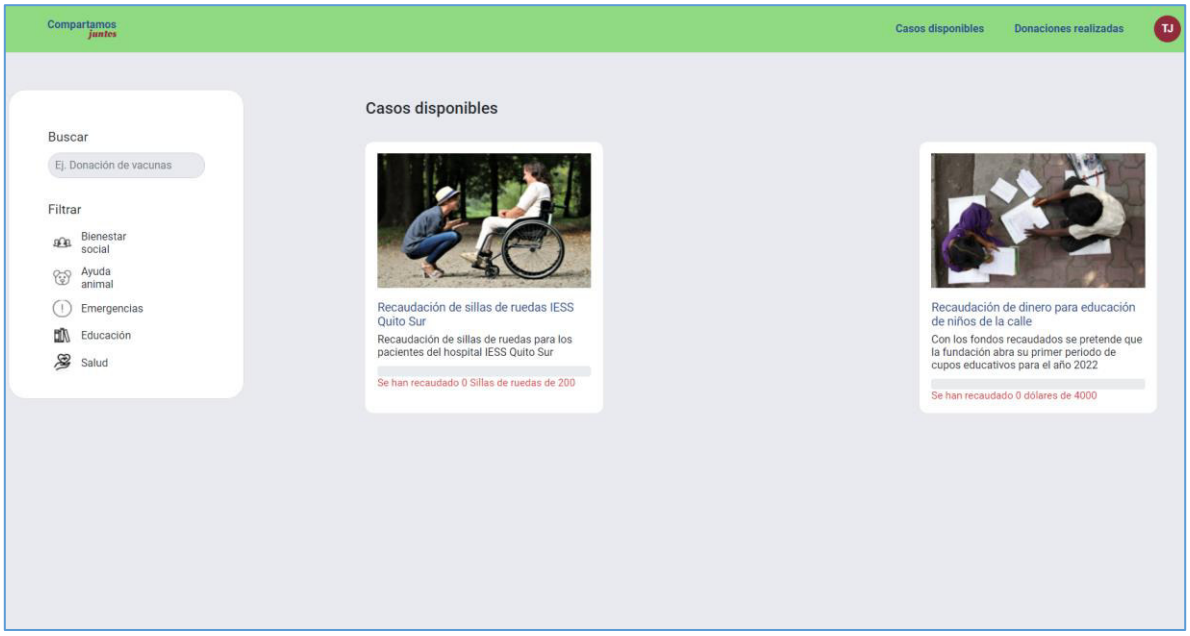

Ilustración 60 - Pantalla de exploración de casos

Los usuarios podrán seleccionar los casos en los que se encuentren interesados para acceder a la información detallada y comenzar un proceso de donación. La ilustración 61 muestra un ejemplo del detalle de un caso de donación dentro del sistema.

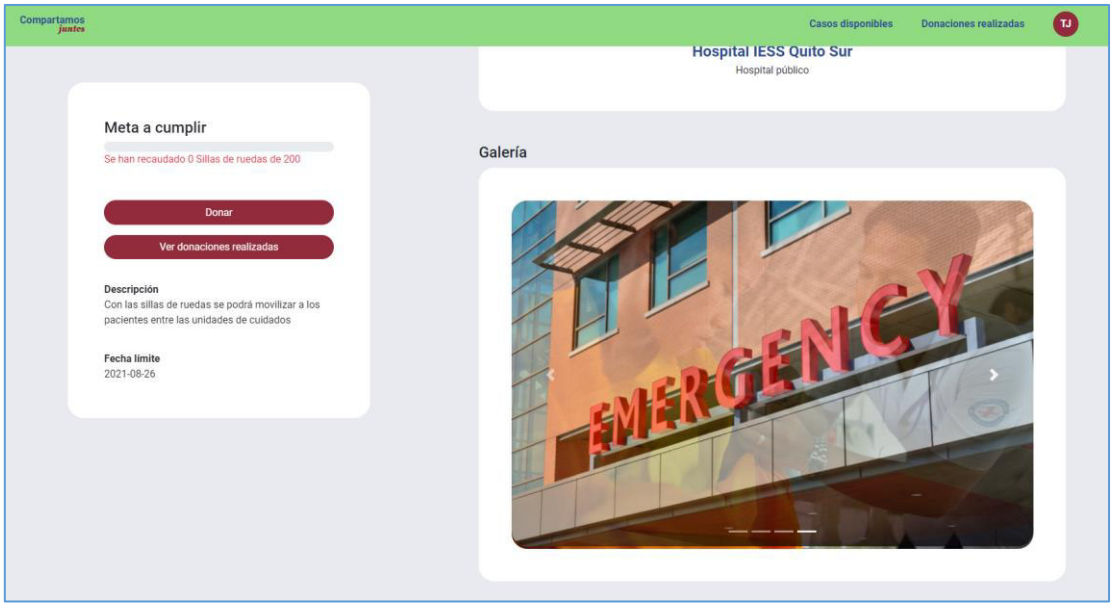

Ilustración 61 - Pantalla de detalle de caso de donación - Módulo donadores

3.1.2.2 Donaciones a casos

Los casos pueden recibir donaciones por medio de distintos métodos. La figura 62 muestra el ejemplo del registro de una donación en donde el usuario donador elije la opción de ponerse en contacto con el usuario donatario para entregar 40 sillas de ruedas.

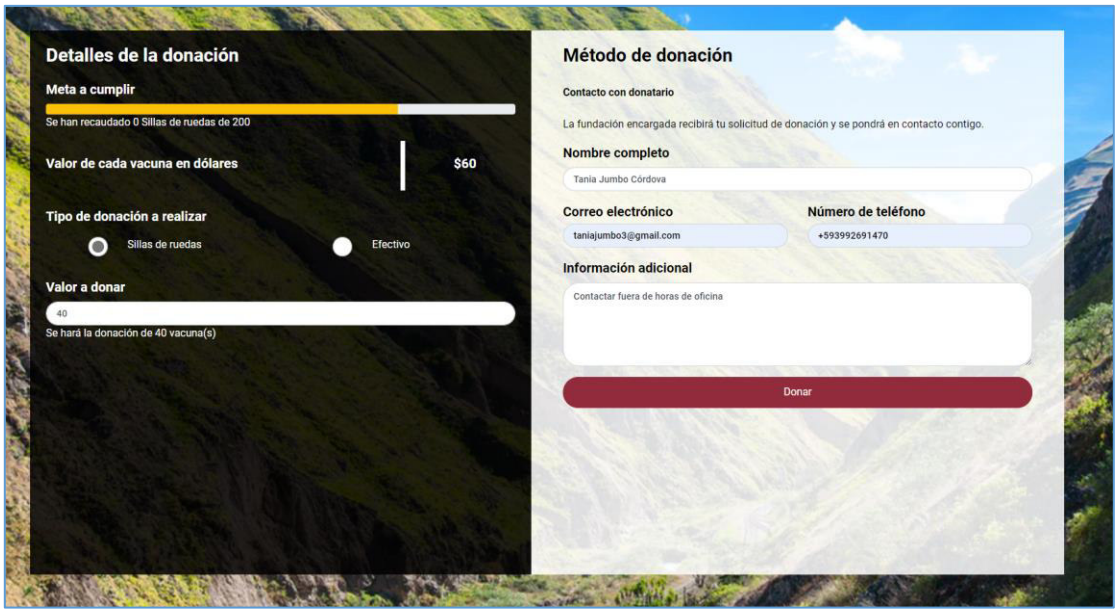

Ilustración 62 - Pantalla de donaciones - Contacto con donatario

3.1.2.3 Estado de donaciones realizadas

Las donaciones que se han registrado en un caso en específico pueden ser monitoreadas para observar el cambio de estado a lo largo del tiempo. El usuario podrá acceder a dicha funcionalidad por medio de la pantalla de detalle del caso. La figura 63 muestra un ejemplo del estado de la donación realizada en la figura 62 cuando aún no ha sido aprobada o rechazada.

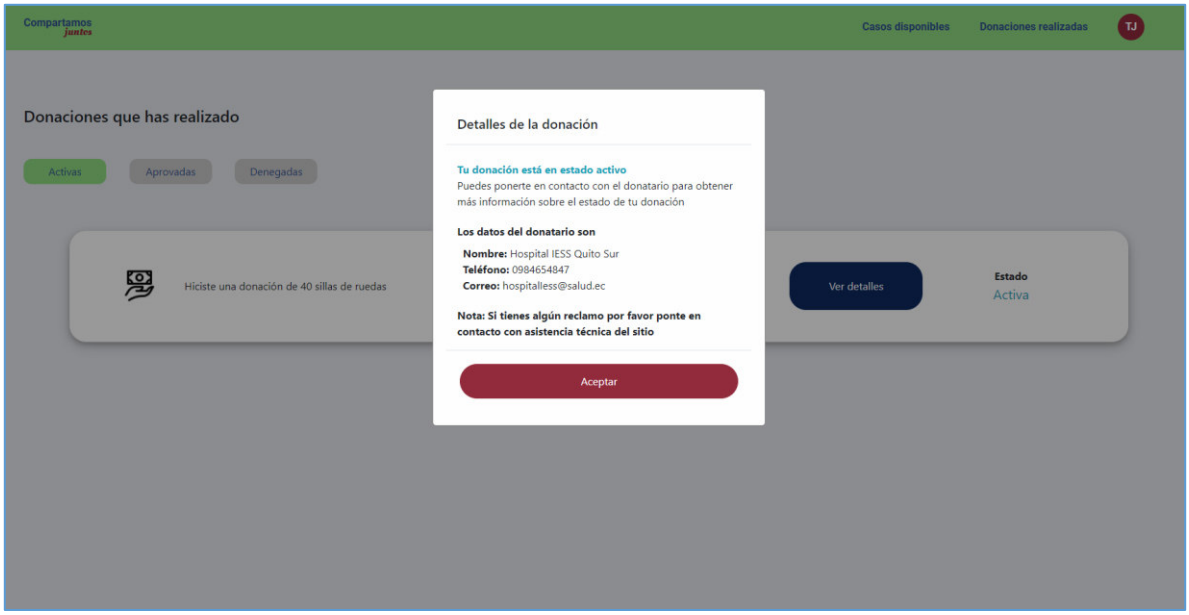

Ilustración 63 - Pantalla de monitoreo de donaciones

3.1.3 Módulo de donatarios

Los usuarios donatarios pueden realizar las siguientes acciones dentro del sistema:

- Acciones definidas para usuarios donadores
- Postular como usuarios donatarios
- Crear y editar casos de donación
- Administrar donaciones recibidas
- Explorar casos creados

3.1.3.1 Postulación de usuarios donatarios

Los usuarios donadores pueden postular como usuarios donatarios por medio de la pantalla principal del sistema. Las ilustraciones de la 64 a la 67 muestran el flujo de postulación que deberán los usuarios que quieran optar como donatarios dentro del sistema.

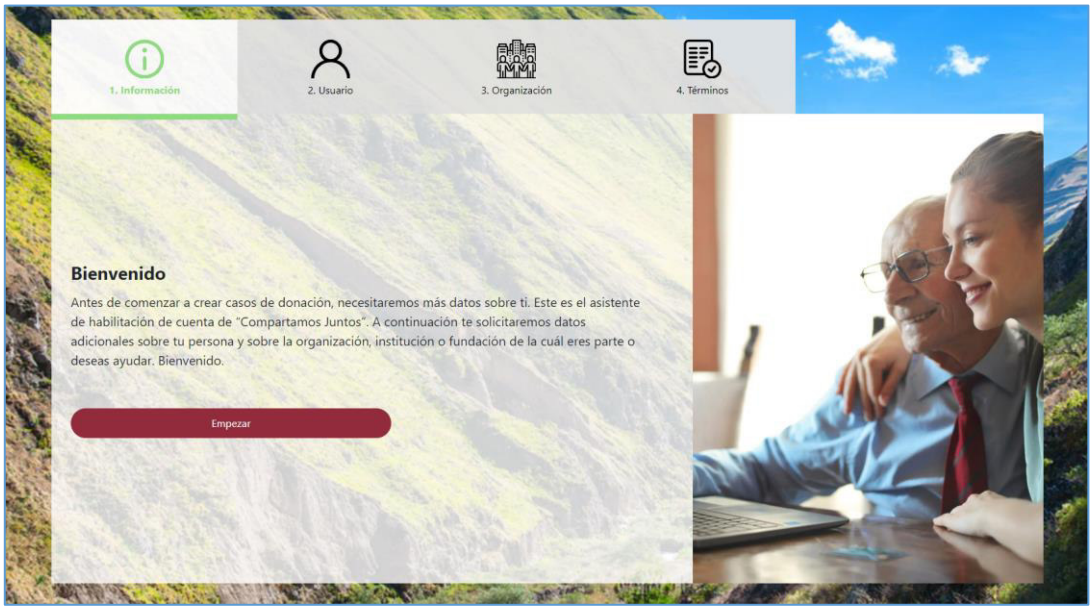

Ilustración 64 - Postulación como usuario donatario - Paso 1

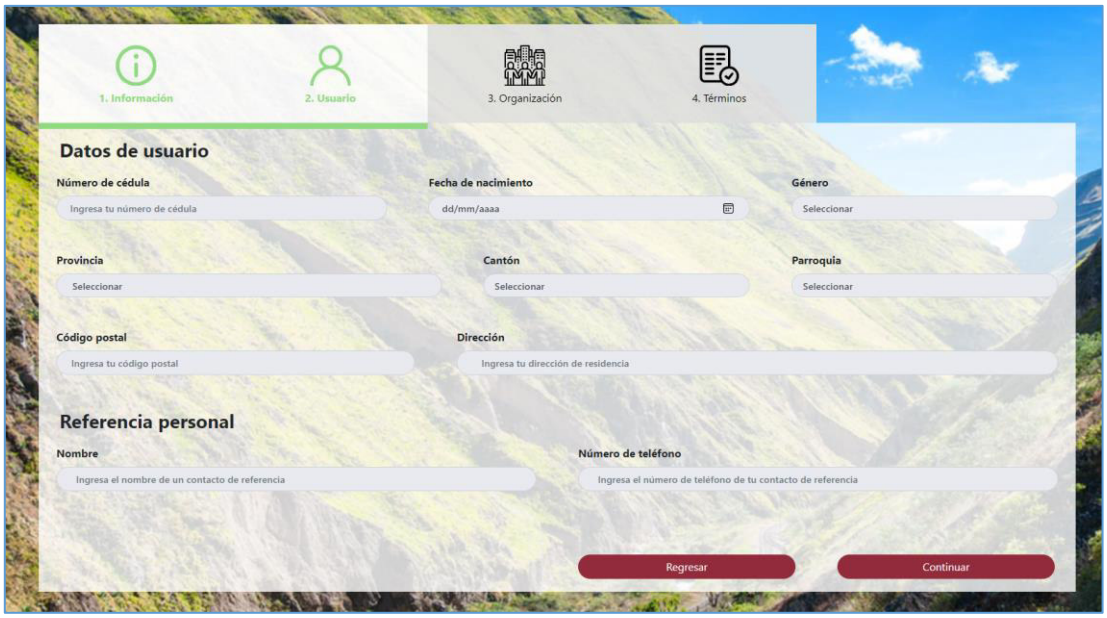

Ilustración 65 - Postulación como usuario donatario - Paso 2

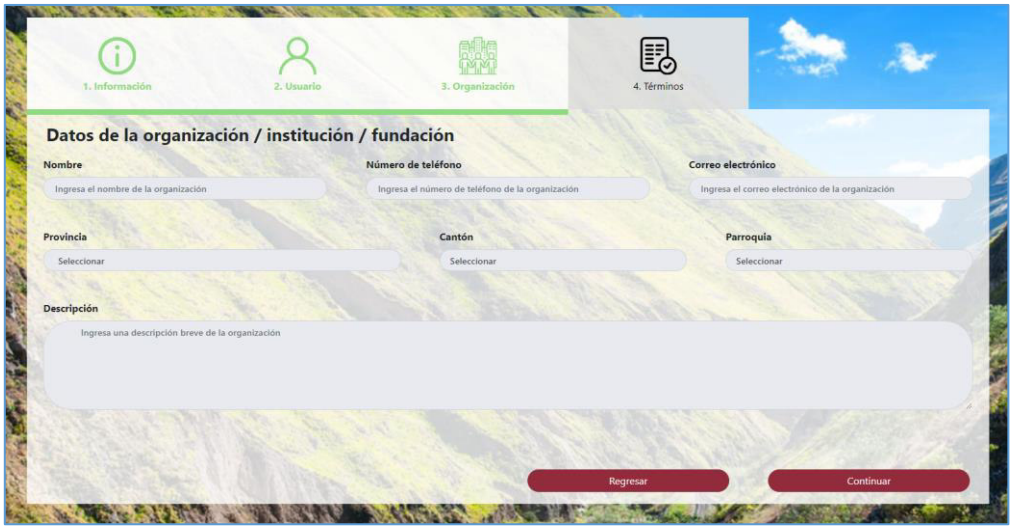

Ilustración 66 - Postulación como usuario donatario - Paso 3

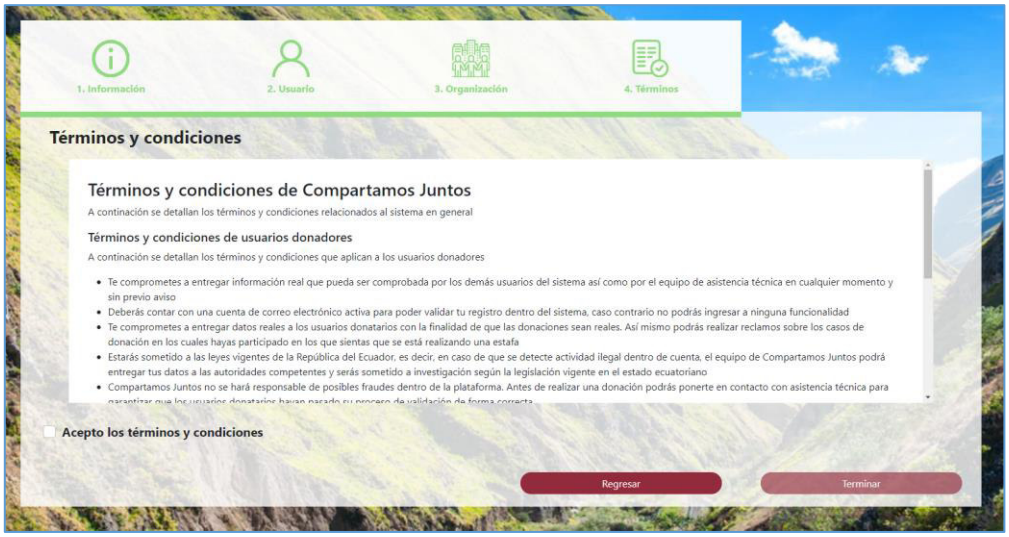

Ilustración 67 - Postulación como usuario donatario - Paso 4

3.1.3.2 Creación y edición de casos de donación

Los usuarios donatarios cuya cuenta haya sido validada podrán acceder a la creación y edición de casos de donación. La figura 68 muestra el formulario de edición de un caso de donación previamente registrado en el sistema.

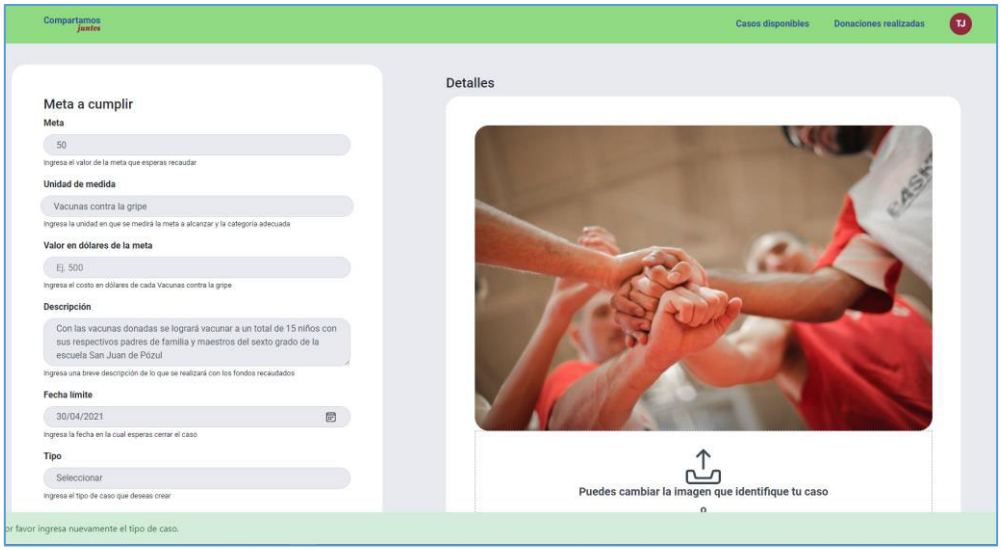

Ilustración 68 - Pantalla de edición de casos

3.1.3.3 Administración de donaciones recibidas

Los usuarios donatarios recibirán donaciones mientras sus casos de donación se encuentren activos dentro del sistema. Los usuarios podrán acceder a la pantalla de administración de casos desde el detalle de los casos creados y podrán aceptar o rechazar las donaciones recibidas. La ilustración 69 muestra las opciones de las que dispone el usuario donatario dentro de la pantalla de administración de donaciones.

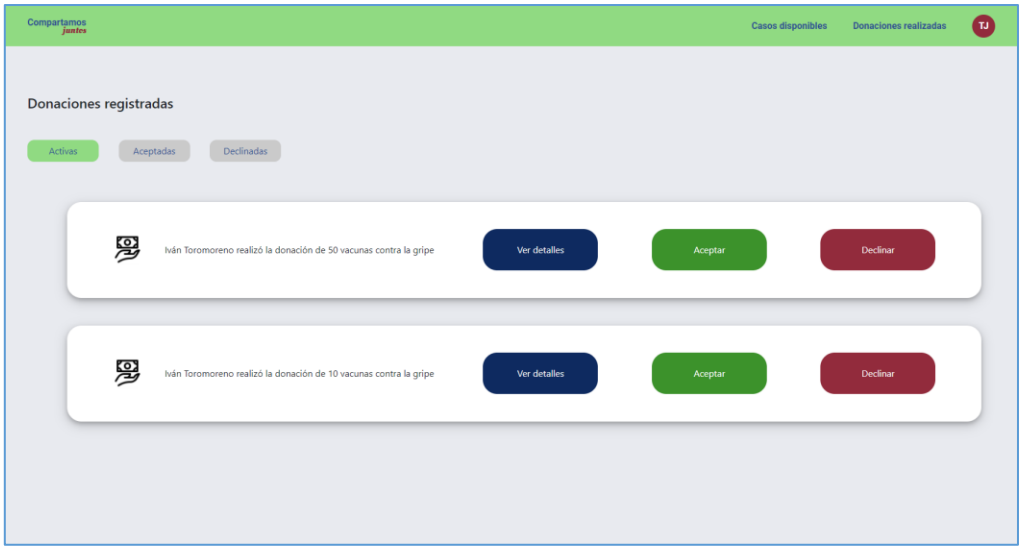

Ilustración 69 - Pantalla de administración de donaciones

3.1.3.4 Exploración de casos creados

Los usuarios donatarios podrán explorar los casos que han creado a lo largo del tiempo. La figura 70 muestra la pantalla de exploración de casos creados en donde se observan las funcionalidades de filtrar casos según su estado y nombre.

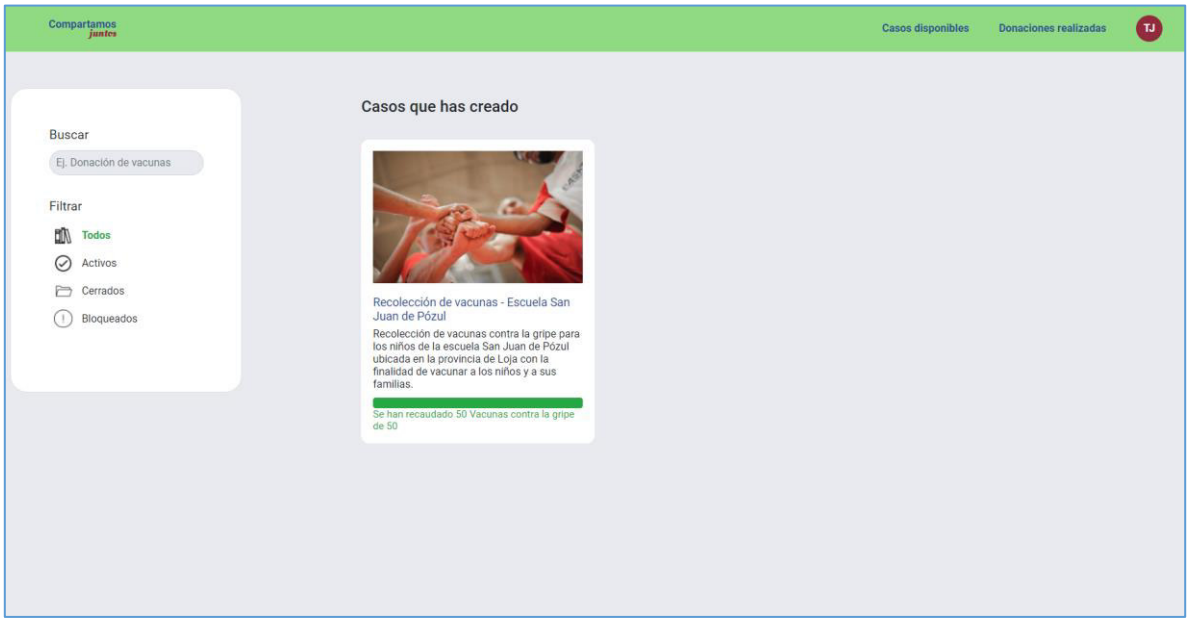

Ilustración 70 - Pantalla de exploración de casos creados

3.2 Documentación en Azure DevOps

El eje central del presente proyecto es mostrar el flujo de trabajo que se sigue dentro de una cultura DevOps, por lo cual la herramienta Azure DevOps fue el pilar fundamental para tener una documentación centralizada acerca de la evolución del proyecto a lo largo de su construcción. La presente sección mostrará detalladamente la documentación generada de forma automática y manual dentro de la herramienta y servirá como reflexión de los beneficios obtenidos al utilizar este esquema de trabajo por encima de un esquema de trabajo convencional.

3.2.1 Centro de documentos del proyecto (Wiki)

El centro de documentos permite mantener al equipo al tanto de la información del proyecto. El sistema desarrollado en el presente documento cuenta con las siguientes páginas dentro de la wiki del proyecto:

- Overview: Contiene información general del proyecto
- Roles de usuario: Contiene los roles de los usuarios del sistema
- Ambiente: Cuenta con enlaces a los ambientes que guardan relación con el proyecto
- Diseño del sitio: Cuenta con un enlace al proyecto en figma donde se detalla el diseño del aplicativo
- Diccionario de datos: Contiene la definición de los diccionarios de datos que usará el sistema
- Diagrama entidad relación: Contiene el diagrama de base de datos que sirvió de base para la implementación del sistema
- Diagrama de componentes: Contiene el diagrama de la arquitectura del sistema mostrando a detalle los componentes relacionados
- Estructura de IDs de work ítems: Contiene la estructura que deberá seguir el equipo para el nombramiento de los ítems de trabajo del proyecto
- Tablero de pruebas del sistema: Contiene un enlace al proyecto en trello donde se encuentra el detalle de las pruebas funcionales del sistema

La ilustración 71 muestra la página de Overview del proyecto.

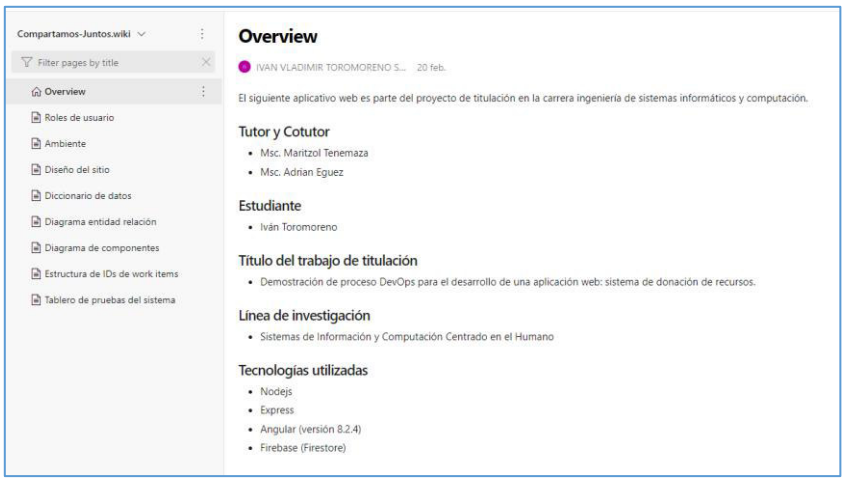

Ilustración 71 - Wiki - Overview del proyecto

3.2.2 Dashboards

Los tableros de desempeño del equipo muestran información relevante que involucra a los equipos de desarrollo y operaciones. La información mostrada está relacionada con métricas que permiten evaluar el correcto desempeño de los diferentes actores dentro del proyecto. Los tableros mostrados en las ilustraciones 49 y 50 son ejemplo de los Dashboards configurados para el presente proyecto. A continuación, se explicará a detalle la información contenida dentro de cada uno de los tableros previamente mostrados.

3.2.2.1 Tablero de historial de ejecución de pipelines

El tablero de historial de ejecución de pipelines muestra las últimas ejecuciones de un pipeline en específico del equipo. Las barras que se muestran en verde corresponden a las construcciones que se realizaron de forma correcta y las barras que se muestran en rojo corresponden a construcciones que tuvieron un fallo. La altura de las barras refleja la etapa en la cual ocurrió el fallo, entre más alta sea la barra desplegada, más avanzada estuvo la construcción de un commit específico. La figura 72 muestra el historial de construcciones realizadas en el último sprint de desarrollo en donde se puede apreciar que las construcciones con éxito superan en número a las construcciones fallidas. Este tablero es sumamente útil ya que el equipo de operaciones puede estar al tanto de los pipelines fallidos en tiempo real y averiguar la causa del problema lo antes posible junto al equipo de desarrollo.

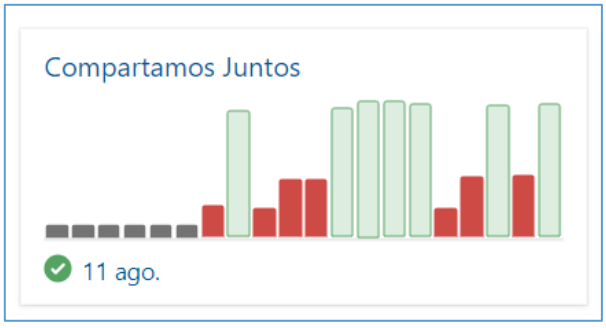

Ilustración 72 - Historial de construcciones

3.2.2.2 Tablero de desempeño del equipo

El tablero de desempeño del equipo muestra una comparación entre los ítems de trabajo realizados y los días de trabajo restantes de tal manera que todo el equipo tenga un entendimiento claro del desempeño cada día del sprint. A diferencia del burndown chart este gráfico contiene la información abreviada de los ítems que se deben implementar sin tomar en cuenta la capacidad del equipo. La figura 73 muestra el gráfico de desempeño correspondiente al sprint 3 en donde se aprecia como las tareas han sido terminadas dentro de los días de trabajo.

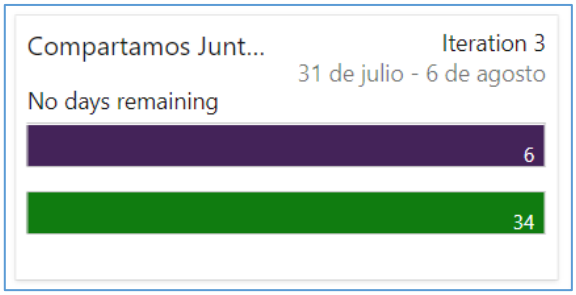

Ilustración 73 - Desempeño del equipo - Sprint 3

3.2.2.3 Burndown Chart

El burndwon chart muestra la evolución del avance del equipo con respecto a los ítems del sprint backlog, en la gráfica se detalla el alcance del sprint, así como, la capacidad del equipo junto con el avance realizado a lo largo de la iteración.

3.2.2.4 Commits realizados

La gráfica de commits realizados muestra la cantidad de push que se han realizado al repositorio de git Azure, esto sirve como contexto al equipo de la cantidad de cambios que se han realizado al sistema y en conjunto con el tablero de construcción puede dar una idea clara de cómo el equipo de desarrollo realiza su trabajo a lo largo del tiempo.

3.2.3 Boards

La sección de boards permite tener un registro de todos los ítems de trabajo registrados en el proyecto organizados según la configuración inicial de la metodología utilizada. El

presente proyecto hizo uso del framework Scrum por lo que los ítems de trabajo fueron agrupados en las siguientes categorías:

- Épicas: Especificación de funcionalidades a nivel de módulos
- Features: Especificación de funcionalidades de nivel específico dentro del módulo
- Historia de usuario: Especificación atómica de funcionalidades que puede empezar a ser desarrollada por el equipo de trabajo
- Bugs: Registros de errores detectados en la ejecución del sistema
- Tareas: Especificación técnica de un trabajo necesario para conseguir resolver una historia de usuario o un bug.

3.2.3.1 Ítems de trabajo

El tablero de ítems de trabajo permite explorar todos los ítems registrados por el equipo además de permitir registrar nuevos ítems que ayuden a gestionar los requerimientos funcionales y no funcionales de la aplicación. El tablero es accesible por todos los miembros del equipo para tener transparencia sobre los detalles de cada ítem creado. La ilustración 74 muestra el tablero de ítems de trabajo del presente proyecto en donde se ve a breves rasgos las épicas, features e historias de usuario implementadas.

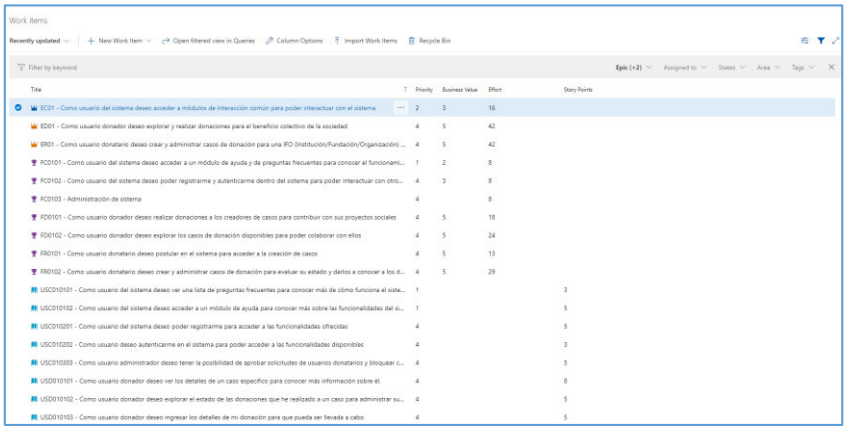

Ilustración 74 - Tablero de ítems de trabajo

3.2.3.2 Boards

El tablero de boards corresponden a un tablero kanban en donde se muestra en tiempo real el estado de las historias de usuario definidas para el proyecto. El tablero es sumamente

útil para el equipo ya que permite mantener a todos en conocimiento de las actualizaciones realizadas al backlog y tener constancia del avance del proyecto a lo largo del tiempo. La ilustración 75 muestra el tablero de boards del presente proyecto al momento de haber finalizado las implementaciones de todas las historias de usuario planteadas a lo largo del desarrollo.

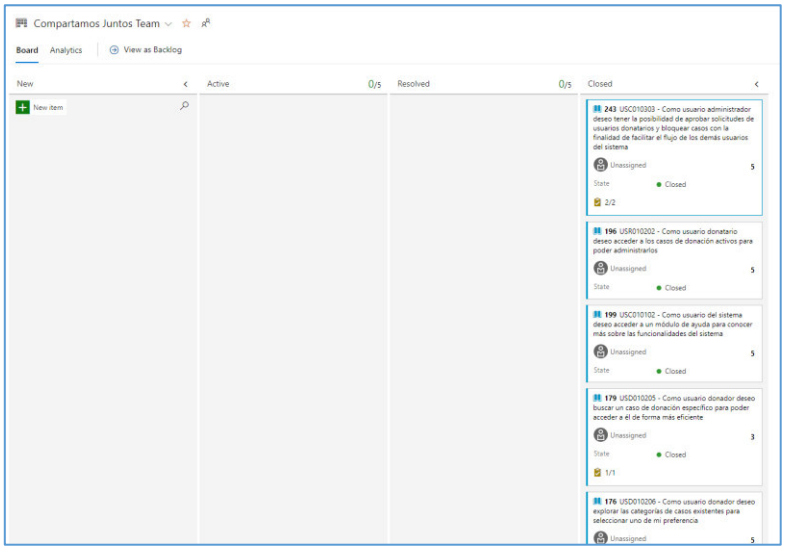

Ilustración 75 - Tablero de historias de usuario al finalizar el proyecto

3.2.3.3 Backlogs

El tablero de backlogs permite distribuir las historias de usuario en las iteraciones definidas en la configuración del proyecto. El tablero permite visualizar en tiempo real los cambios en el sprint backlog. La ilustración 76 muestra el tablero de backlog una vez asignadas todas las historias de usuario del proyecto a las iteraciones registradas.

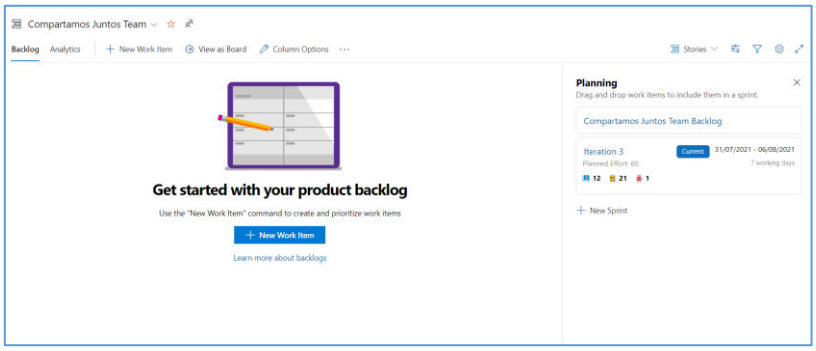

Ilustración 76 - Backlogs de proyecto al finalizar la fase de desarrollo

3.2.3.4 Sprints

El tablero de sprints permite administrar el sprint backlog, la capacidad de trabajo de los miembros del equipo y evaluar el desempeño a lo largo de la iteración. Los tableros de burndown chart mostrados a lo largo del presente proyecto junto con el sprint backlog fueron generados automáticamente por esta funcionalidad de la plataforma. La figura 77 muestra el tablero kanban de tareas correspondiente al sprint 3 (en la esquina se puede observar una miniatura del burndown chart del desempeño del equipo).

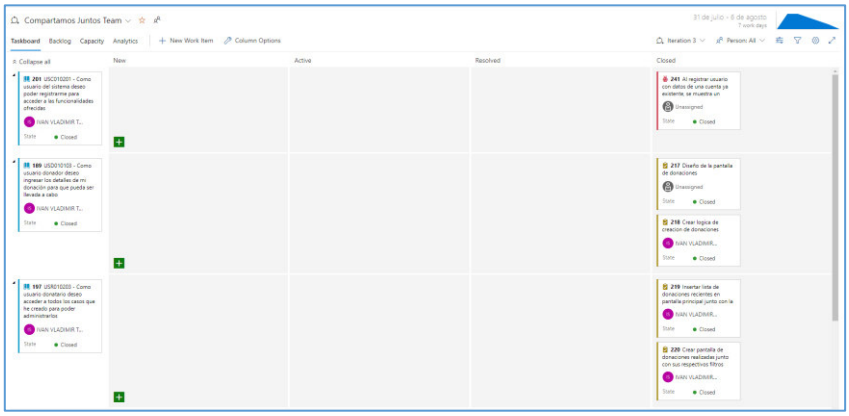

Ilustración 77 - Tablero kanban de tareas - Sprint 3

3.2.4 Repos

El repositorio de azure configurado para el presente proyecto fue de tipo Git. Dentro de la sección repos del proyecto se puede realizar manejo de varios repositorios administrar el código fuente de las aplicaciones que formen parte del sistema sin embargo al contar con una única aplicación para el desarrollo del presente aplicativo solo se contó con un repositorio central organizado con el esquema de trabajo Gitflow. La ilustración 78 muestra el historial de commits realizados sobre el presente proyecto a lo largo de la etapa de desarrollo sobre la rama master o principal, adicionalmente en la parte derecha de los commits se puede visualizar el resultado de la ejecución del pipeline asociado al commit.

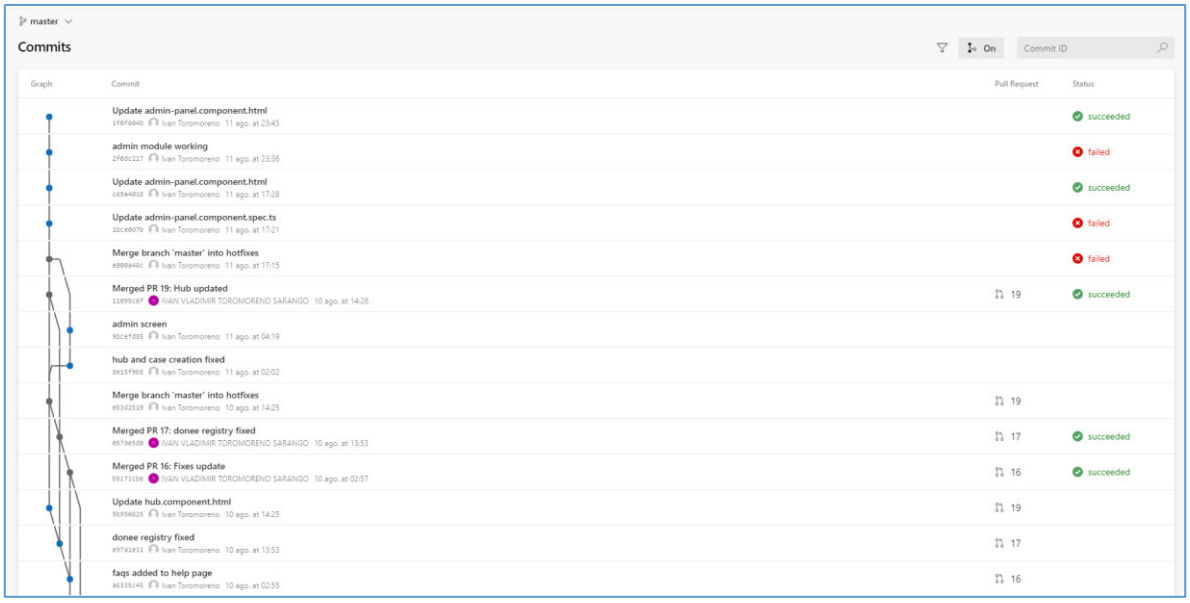

Ilustración 78 - Historial de commits sobre rama master

3.2.5 Pipelines

La funcionalidad de pipelines es una de las principales herramientas para los equipos de DevOps ya que permite automatizar las fases de integración y despliegue de cambios de la aplicación (CI/CD). El presente proyecto contó con un pipeline que realizaba el ciclo de construcción, pruebas y despliegue de cambios implementados al ambiente de pruebas sobre la plataforma Heroku. La ilustración 79 muestra el historial de las últimas construcciones realizadas por medio del pipeline principal del proyecto.

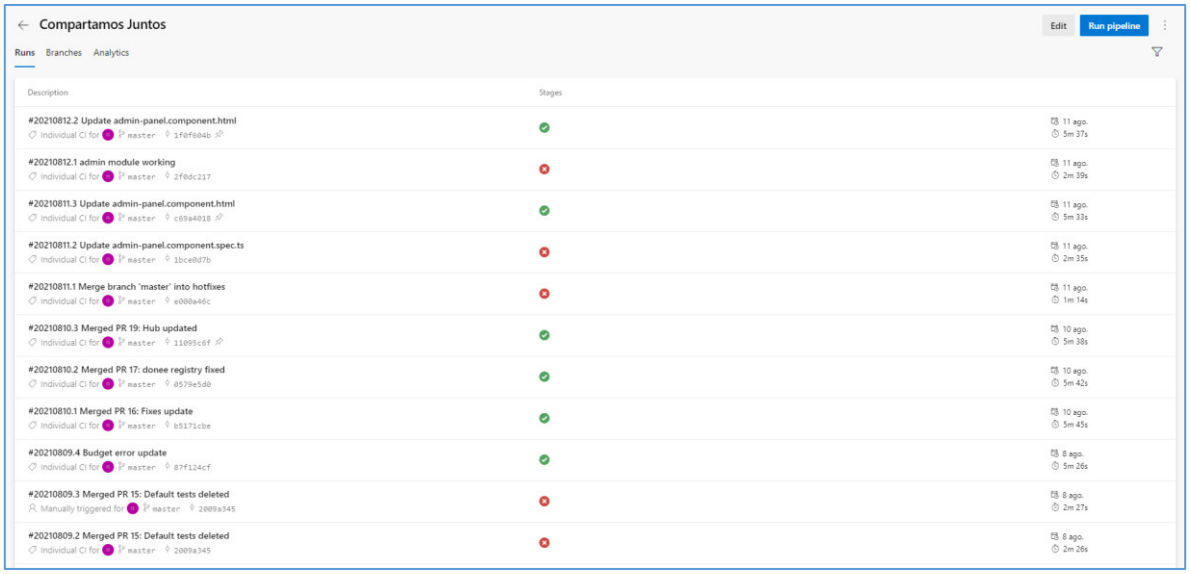

Ilustración 79 - Historial de construcción de la app por medio del pipeline configurado

3.2.6 Discusión de esquema de trabajo

El presente proyecto tomó como objetivo principal demostrar el proceso de construcción de un producto de software haciendo uso del esquema de trabajo DevOps. DevOps presenta un conjunto de prácticas y recomendaciones para equipos grandes de trabajo en donde se cuenta con varios departamentos y en donde usualmente los encargados del desarrollo de un aplicativo no son necesariamente los responsables de llevar a operaciones el código fuente con los últimos cambios realizados. Sin embargo, la metodología las mismas soluciones planteadas por DevOps para grandes equipos pueden ser usadas para proyectos de menor escala en donde el equipo de desarrollo conste de pocos miembros de trabajo en donde las responsabilidades son compartidas entre todos los miembros o incluso proyectos realizados por una sola persona con varias responsabilidades como se presenta en el presente documento.

Las prácticas que se plantean en un equipo de trabajo DevOps permiten agilizar las entregas funcionales a los clientes y reducir la incertidumbre que se produce al introducir nuevos cambios a un ambiente productivo. La base para conseguir estos objetivos es tener transparencia entre todos los miembros del equipo y herramientas que permitan monitorizar el trabajo realizado a lo largo del ciclo de vida de un proyecto. La herramienta elegida para el presente proyecto es Azure DevOps la cual cuenta con todas las funcionalidades

necesarias para que un equipo de trabajo que implemente esta cultura dentro de su organización pueda gestionarse y tener sinergia para conseguir los objetivos planteados.

Las principales ventajas que se identificaron de implementar el esquema de trabajo DevOps para la realización del presente proyecto fueron:

- Transparencia del trabajo que se está realizando a lo largo del tiempo.
- Organización clara de los ítems de trabajo.
- Evaluación diaria de los cambios que sufre el proyecto para monitorear el rendimiento del equipo y de los procesos.
- Un único centro de la verdad definido por el repositorio con el código fuente.
- Automatización de procesos repetitivos

De igual manera el equipo pudo identificar los siguientes puntos a mejorar al seguir este esquema de trabajo:

- Tener mayor contexto de las herramientas necesarias.
- Contar con mayor experiencia en automatización de tareas.
- Mejor uso de la metodología TDD para comprobar el estado del sistema.

Si bien el equipo no tuvo que pasar por un proceso de adaptación de una metodología convencional hacia el esquema de trabajo DevOps, el esquema de trabajo utilizado resultó flexible y entendible para actores externos que participaron como soporte en el proceso de desarrollo.

3.3 Pruebas funcionales con actores externos

El presente proyecto contó con 2 etapas de pruebas funcionales. La primera etapa fue realizada por Anahí Hurtado, estudiante de la carrera de derecho de la Universidad de las Américas y se realizó sobre la versión alfa del aplicativo con la finalidad de encontrar errores en los flujos de trabajo de los usuarios. La segunda fase de pruebas fue llevada a cabo por los actores mostrados en la tabla 12 y se centró en la percepción de los usuarios hacia el sistema.

Sujetos de pruebas – Fase beta del aplicativo
Ocupación
Estudiante de derecho – Universidad de las Américas
Estudiante de Ciencias Sociales – Universidad Central del Ecuador
Estudiante de Administración de empresas – Universidad Central del Ecuador
Estudiante de Psicología – Universidad Politécnica Salesiana
Estudiante de Química – Universidad Central del Ecuador

Tabla 12 - Sujetos de prueba de fase beta del aplicativo

3.3.1 Fase de pruebas sobre la versión alfa

La fase de pruebas inicial realizada sobre el aplicativo fue documentada por medio de un tablero en la herramienta *Trello* debido a problemas de accesos a la plataforma Azure DevOps por parte del actor de pruebas. La fase de pruebas inició a mitad del sprint 3 y los errores reportados fueron documentados y corregidos a lo largo del sprint de cierre del proyecto. El esquema de pruebas constó de distintas etapas por las cuales las pruebas podían pasar dependiendo del contexto en el que se encontraban y fueron:

- Lista de pruebas: Se definen las pruebas a realizar sobre el aplicativo, en esta lista se encuentran las pruebas que aún no han sido llevadas a cabo.
- Realizando pruebas: en esta etapa se encuentran las pruebas que se encuentran activas por el sujeto de pruebas y permite conocer al equipo de desarrollo que funcionalidades no deben ser afectadas para no sesgar el ciclo de pruebas.
- Prueba fallida: es esta etapa se encuentra una lista de pruebas que no cumplen con los requerimientos establecidos dentro de la especificación de la prueba y se agrega un detalle breve del fallo encontrado para posteriormente ser corregido por el equipo de desarrollo.
- Prueba correcta: en esta etapa se encuentran las pruebas que han cumplido todas las especificaciones definidas dentro de la prueba.
- Correcciones sugeridas: en esta etapa se encuentran las pruebas que han cumplido las especificaciones definidas por la prueba pero que sin embargo muestran fallos que no representan un bloqueo dentro de la aplicación y que se sugiere sean corregidos.

Las pruebas reportadas como fallidas o con acciones sugeridas fueron solventadas a lo largo del sprint 4, al finalizar la iteración el tablero de pruebas reflejaba que todas las pruebas pasaron a fase de prueba correcta.

3.3.2 Fase de pruebas sobre prototipo

La última fase de pruebas fue llevada a cabo por actores externos al equipo de desarrollo del presente proyecto que no poseían conocimientos técnicos sobre programación y que buscan simular un grupo de usuarios reales que pueden interactuar con el sistema final.

Las pruebas llevadas a cabo fueron de carácter exploratorio, es decir, no se contó con un listado de pruebas específico, sino que más bien se solicitó una libre interacción con el sistema con la finalidad de reportar su percepción con las funcionalidades a las cuales tuvieron acceso.

El resultado de las pruebas realizadas en esta fase fue documentado por medio de una encuesta realizada en Google Forms cuyo resultado a nivel general se ve reflejado en la ilustración 80.

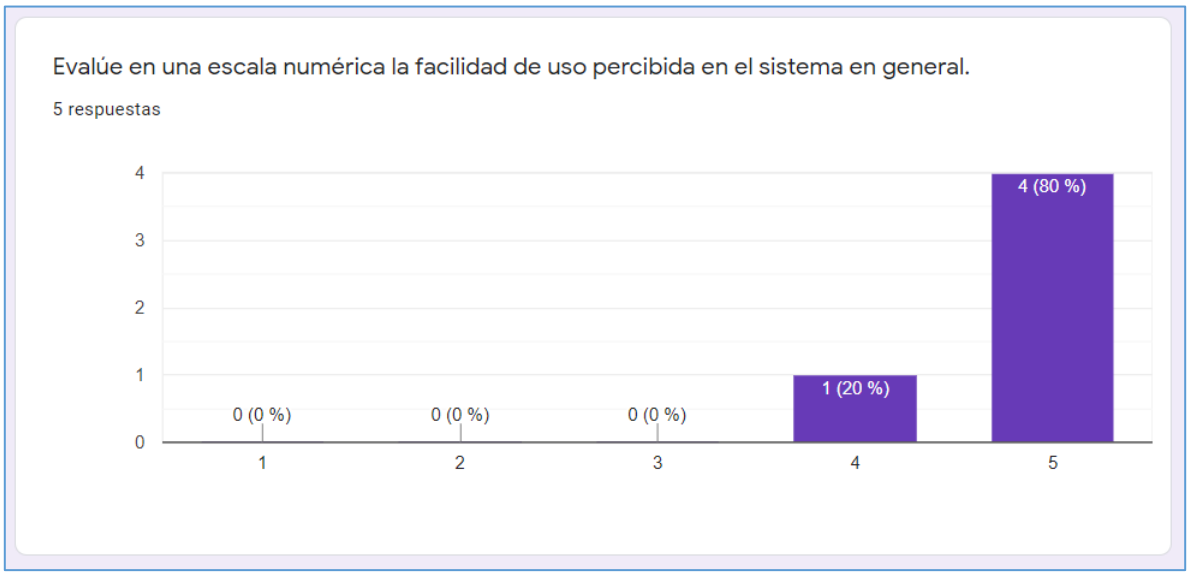

Ilustración 80 - Resultados de evaluación de percepción de uso del sistema.

Los resultados reflejan que la aplicación cuenta con un alto nivel de aceptación con respecto a la facilidad de uso percibida por los sujetos de pruebas que formaron parte de esta etapa

del proceso. La encuesta puede ser visualizada a detalle en la sección de anexos del presente documento.

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- DevOps es un concepto que puede tomar varias definiciones dependiendo el contexto en el que sea cuestionado, sin embargo, la definición más acertada es considerar a DevOps como un marco de trabajo que brinda un conjunto de recomendaciones y herramientas para los equipos de desarrollo de aplicaciones de software.
- El núcleo del esquema de trabajo DevOps no se encuentra en las herramientas de trabajo utilizadas en la etapa de desarrollo de un proyecto sino en la cultura de sinergia que debe tener el equipo que busca adaptarse a esta forma de trabajo.
- Azure DevOps es una plataforma que contiene un set de herramientas que permiten a un equipo llevar a cabo el esquema de trabajo DevOps de forma efectiva y transparente. Las funcionalidades que fueron mostradas a lo largo del desarrollo del proyecto resultaron sumamente útiles para poder mantener consistencia entre los miembros que formaron parte del equipo y permitieron volver a integrarse al trabajo con facilidad posterior a un largo período de inactividad.
- Las métricas obtenidas como parte del proceso de DevOps ayudaron a monitorear el estado del equipo a lo largo del período de desarrollo, permitiendo transparentar el desempeño que se estaba llevando a cabo a lo largo del proyecto y obteniendo punto de mejora que podían ser tomados en cuenta al finalizar cada sprint definido.
- La utilización de firebase como herramienta de desarrollo permitió agilizar el proceso de construcción del prototipo ya que las funcionalidades que provee esta plataforma ayudan a evitar la fase de construcción de una aplicación de backend que maneje la lógica de autenticación, almacenamiento de datos y manejo de archivos que son necesarios para los requerimientos establecidos.
- El framework de desarrollo Angular permite brindar una experiencia de uso más fluida a los usuarios de la aplicación por medio del esquema SPA, además permite que el equipo de desarrollo mantenga un esquema estándar de implementación de la lógica necesaria haciendo que el sistema sea más mantenible a largo plazo y contenga mejores prácticas de desarrollo web.

- La utilización de pipelines permite al equipo automatizar tareas repetitivas relacionadas a la construcción, pruebas y despliegue de las aplicaciones realizadas, lo cual permite minimizar los tiempos de despliegue, mantener transparencia entre los equipos involucrados y aprender de los errores que puedan suceder en las fases de construcción.
- La utilización del esquema de trabajo gitflow permite mantener una organización clara del código fuente del sistema, separando en distintas ramas las distintas etapas por las que puede pasar el código. La utilización de este esquema de trabajo permite monitorear de forma más sencilla los cambios que se han implementado en el sistema y controlar de mejor manera lo errores que puedan surgir en el entorno principal de trabajo.
- La aplicación resultante del proyecto contó con todos los requerimientos establecidos en la fase inicial de desarrollo y contó con la participación de actores externos al equipo que validaron la facilidad de uso del sistema junto con la utilidad alineada al objetivo inicial del sistema.

4.2 Recomendaciones

- Se recomienda definir varios pipelines de construcción, despliegue y pruebas con la finalidad de poder monitorear las distintas etapas por las que pueden pasar los entregables del proyecto de forma individual.
- Se recomienda estudiar a profundidad los tableros que ofrece la herramienta Azure DevOps como parte de su set de dashboards con la finalidad de mantener al equipo informado del estado del proyecto desde distintos frentes.
- Se recomienda contar con licencias para el equipo de pruebas, de tal manera que puedan acceder al proyecto principal de la aplicación y colaborar de forma directa con los demás equipos involucrados.
- Se recomienda explorar las reglas de configuración de la base de datos de firestore para evitar exponer datos sensibles a usuarios que no tengan el rol de administración.
- Se recomienda implementar diseño responsivo al sitio para que la experiencia de usuario no se vea afectada en dispositivos móviles.
- Se recomienda estudiar la integración con plataformas de pagos para proveer el servicio de donaciones en efectivo de forma segura.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] M. Walls, «Building a DevOps Culture,» de *Building a DevOps Culture*, Sebastopol, O'Reilly Media, Inc., 2013, pp. 3-7.

[2] L. Yu Beng, L. Wooi Khong, T. Wai Yip y T. Soo Fun, «Software Development Life Cycle AGILE vs Traditional Approaches,» de *International Conference on Information and Network Technology*, Singapore, 2012.

[3] J. Jeremiah, «TechBeacon,» Micro Focus, [En línea]. Available: https://techbeacon.com/app-dev-testing/survey-agile-new-norm. [Último acceso: 19 Agosto 2021].

[4] M. Cabrera, «mfcabrera,» mfcabrera, 14 Febrero 2011. [En línea]. Available: http://mfcabrera.com/blog/2011/2/14/operations-vs-development.html. [Último acceso: 21 Agosto 2021].

[5] I. Buchanan, «Historia de DevOps,» Atlassian, [En línea]. Available: https://www.atlassian.com/devops/what-is-devops/history-of-devops. [Último acceso: 21 Agosto 2021].

[6] L. Ellen, K. Pasi y O. Markku, *Dimensions of DevOps,* Oulu: Springer, Cham, 2015.

[7] C. Jones y F. Group, Software Methodologies a Quantitative Guide, Boca Raton: CRC Press, 2018.

[8] Scrum.org, «What is SCRUM?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-scrum. [Último acceso: 30 Agosto 2021].

[9] Scrum.org, «Scrum Values Tabloid,» Scrum.org, [En línea]. Available: https://scrumorg-website-prod.s3.amazonaws.com/drupal/2018-05/ScrumValues-Tabloid.pdf. [Último acceso: 30 Agosto 2021].

[10] Scrum.org, «What is a Product Owner?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-product-owner. [Último acceso: 30 Agosto 2021].

[11] Scrum.org, «What is a Scrum Master?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-scrum-master. [Último acceso: 30 Agosto 2021].

[12] Scrum.org, «What is a Developer in Scrum?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-scrum-developer. [Último acceso: 30 Agosto 2021].

[13] Scrum.org, «What is a Sprint?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-sprint-in-scrum. [Último acceso: 30 Agosto 2021].

[14] Scrum.org, «What is Sprint Planning?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-sprint-planning. [Último acceso: 30 Agosto 2021].

[15] Scrum.org, «What is a Daily Scrum?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-daily-scrum. [Último acceso: 30 Agosto 2021].

[16] Scrum.org, «What is a Sprint Review?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-sprint-review. [Último acceso: 30 Agosto 2021].

[17] Scrum.org, «What is a Sprint Retrospective?,» Scrum.org, [En línea]. Available: https://www.scrum.org/resources/what-is-a-sprint-retrospective. [Último acceso: 30 Agosto 2021].

[18] Atlassian, «¿Qué es el control de versiones?,» Atlassian, [En línea]. Available: https://www.atlassian.com/es/git/tutorials/what-is-version-control. [Último acceso: 26 Agosto 2021].

[19] Atlassian, «Flujo de trabajo de Gitflow,» Atlassian, [En línea]. Available: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow. [Último acceso: 26 Agosto 2021].

[20] J. Davis y R. Daniels, Effective DevOps, Sabastopool: O'Reilly, 2016.

[21] T. Erl, Z. Mahmood y R. Puttini, Cloud Computing: Concepts, Technology & Architecture, New Jersey: Prentince Hall, 2013.

[22] S. Cummins, «MVP,» Jexo, 02 Abril 2021. [En línea]. Available: https://jexo.io/blog/ppm-glossary-what-is-mvp/. [Último acceso: 29 Agosto 2021].

[23] D. Coward, «Java Servlet Specification,» Sun Microsystems, Inc., Palo Alto, 2000.

[24] Neoteric, «Single-page application vs. multiple-page application,» Neoteric, 2 Diciembre 2016. [En línea]. Available: https://medium.com/@NeotericEU/single-page-

application-vs-multiple-page-application-2591588efe58. [Último acceso: 29 Agosto 2021].

[25] J. Galloway, B. Wilson, K. S. Allen y D. Matson, Professional ASP.NET MVC 5, Indianapolis: John Wiley & Sons, Inc., 2014.

[26] Microsoft, «What is Azure DevOps?,» Microsoft, 22 Enero 2021. [En línea]. Available: https://docs.microsoft.com/en-us/azure/devops/user-guide/what-is-azure-devops?view=azure-devops. [Último acceso: 22 Septiembre 2021].

[27] Google Inc, «Firebase Documentación,» Google - Firebase, [En línea]. Available: https://firebase.google.com/docs. [Último acceso: 22 Septiembre 2021].

[28] Angular, «What is Angular?,» Angular, [En línea]. Available: https://angular.io/guide/what-is-angular. [Último acceso: 30 Agosto 2021].

[29] J. Jeremiah, «TechBeacon,» Micro Focus. [En línea].

6 ANEXOS

6.1 Anexo I: Épicas del sistema

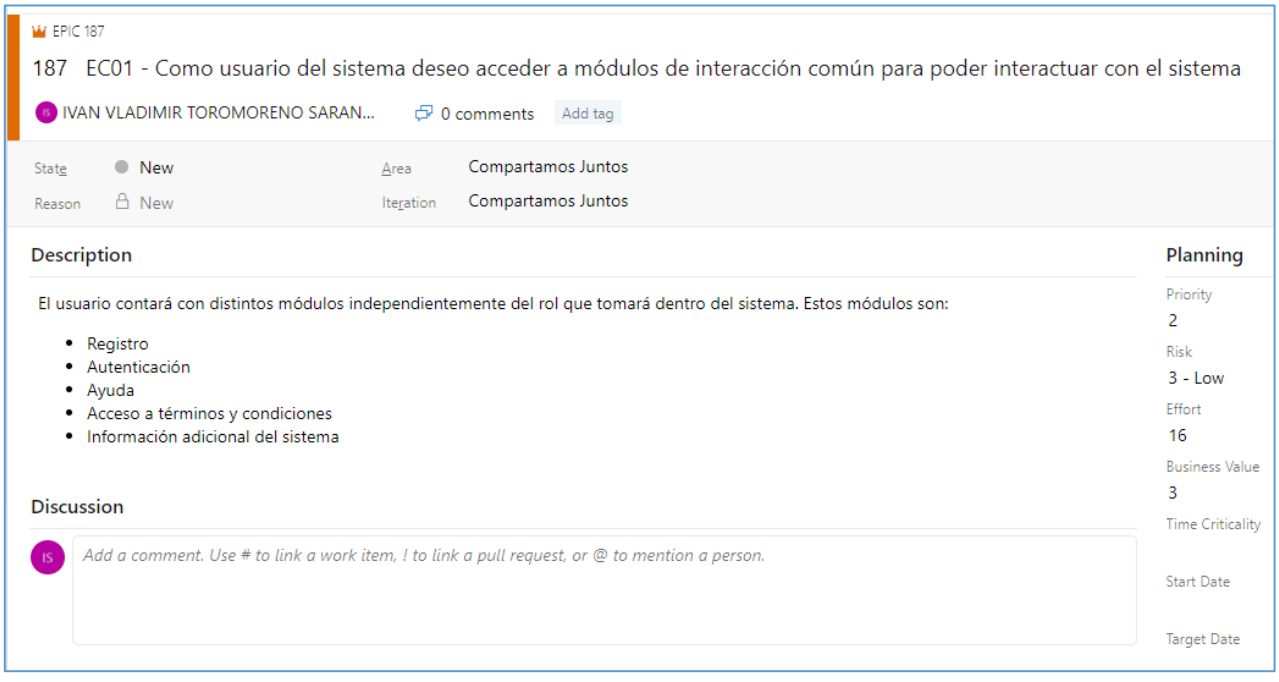

Ilustración 81 - Épica EC01

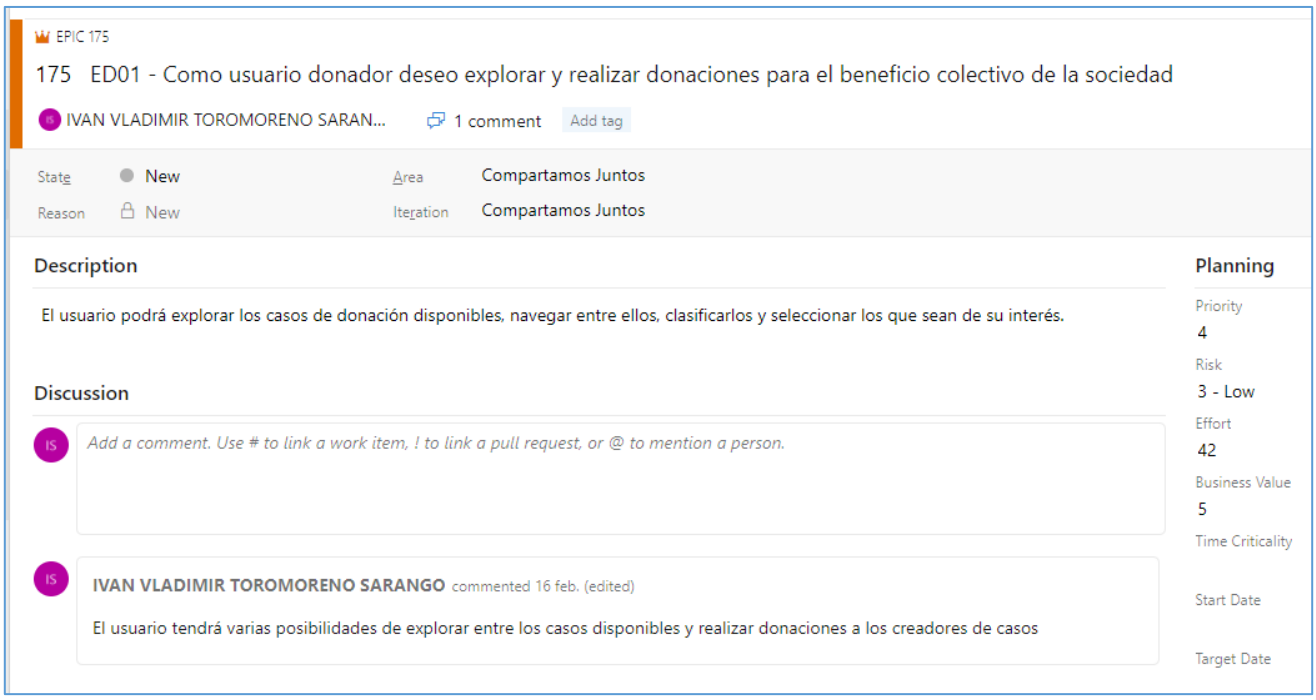

Ilustración 82 - Épica ED01

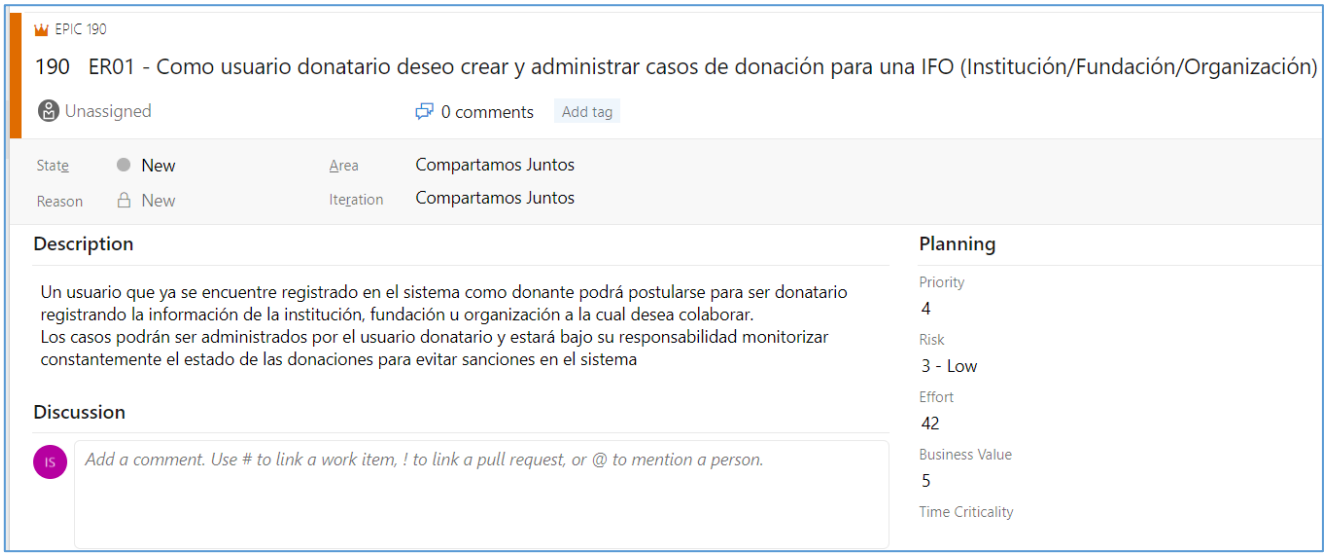

Ilustración 83 - Épica ER01

6.2 Anexo II: Features del sistema

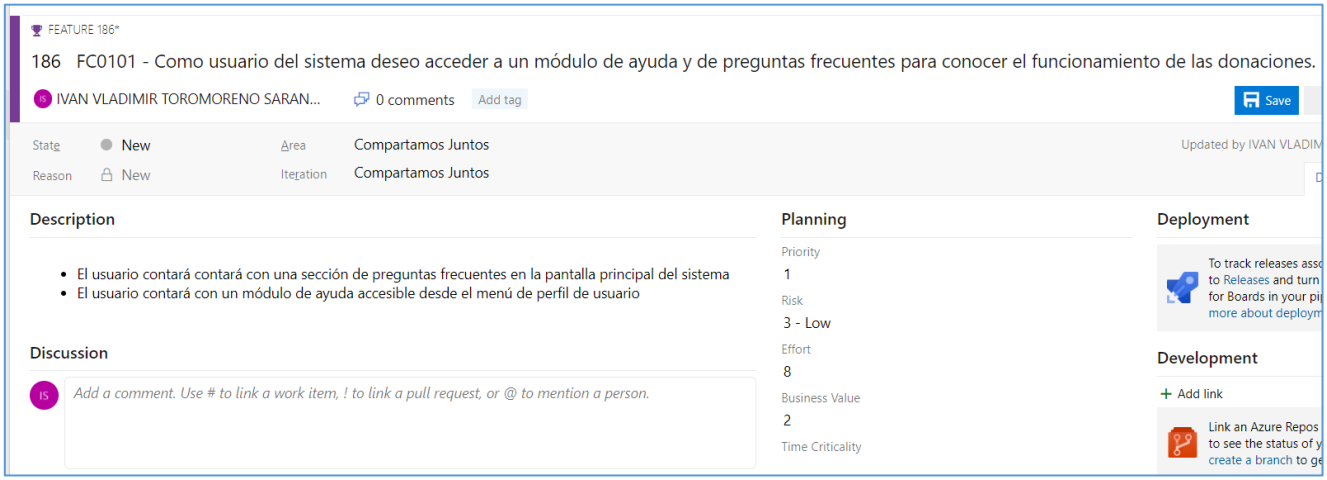

Ilustración 84 - Feature FC0101

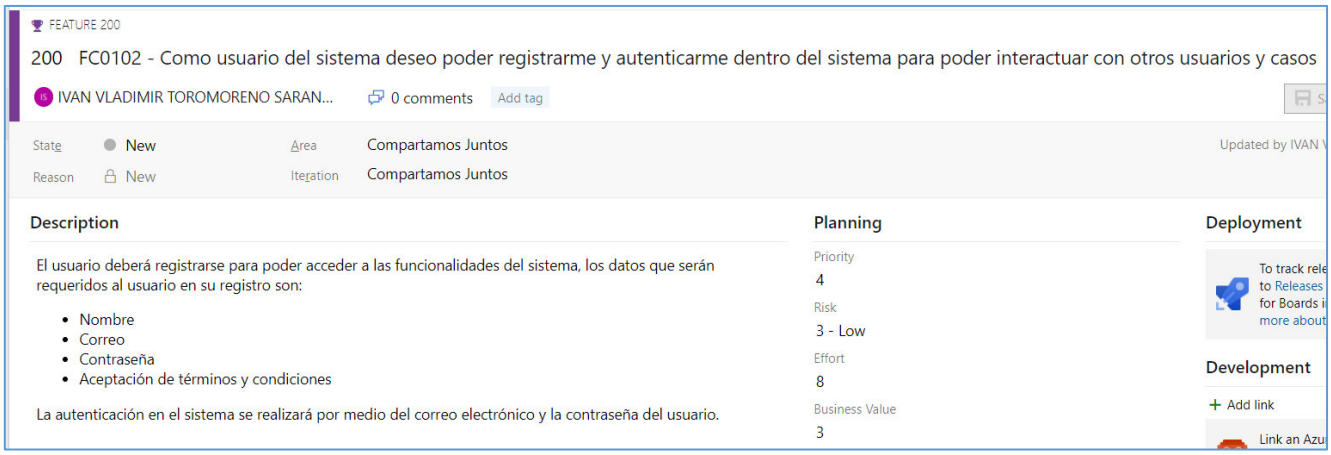

Ilustración 85 - Feature FC0102

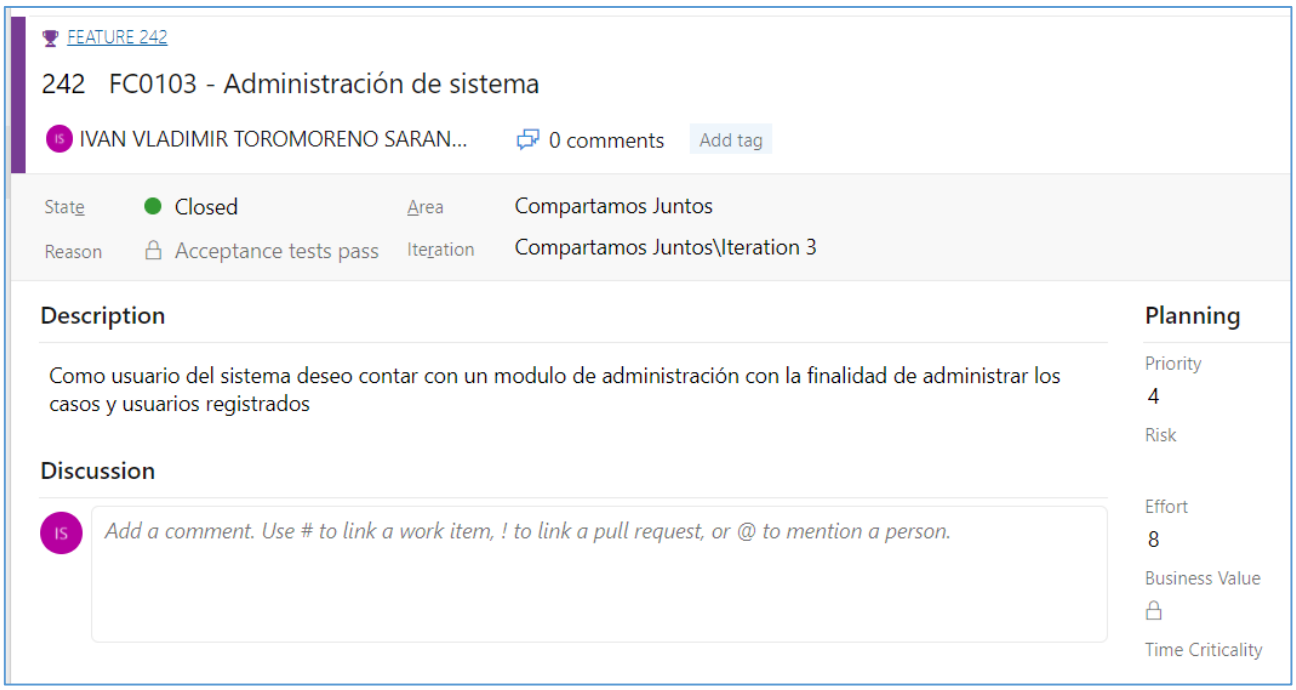

Ilustración 86 - Feature FC0103

Ilustración 87 - Feature FD0101

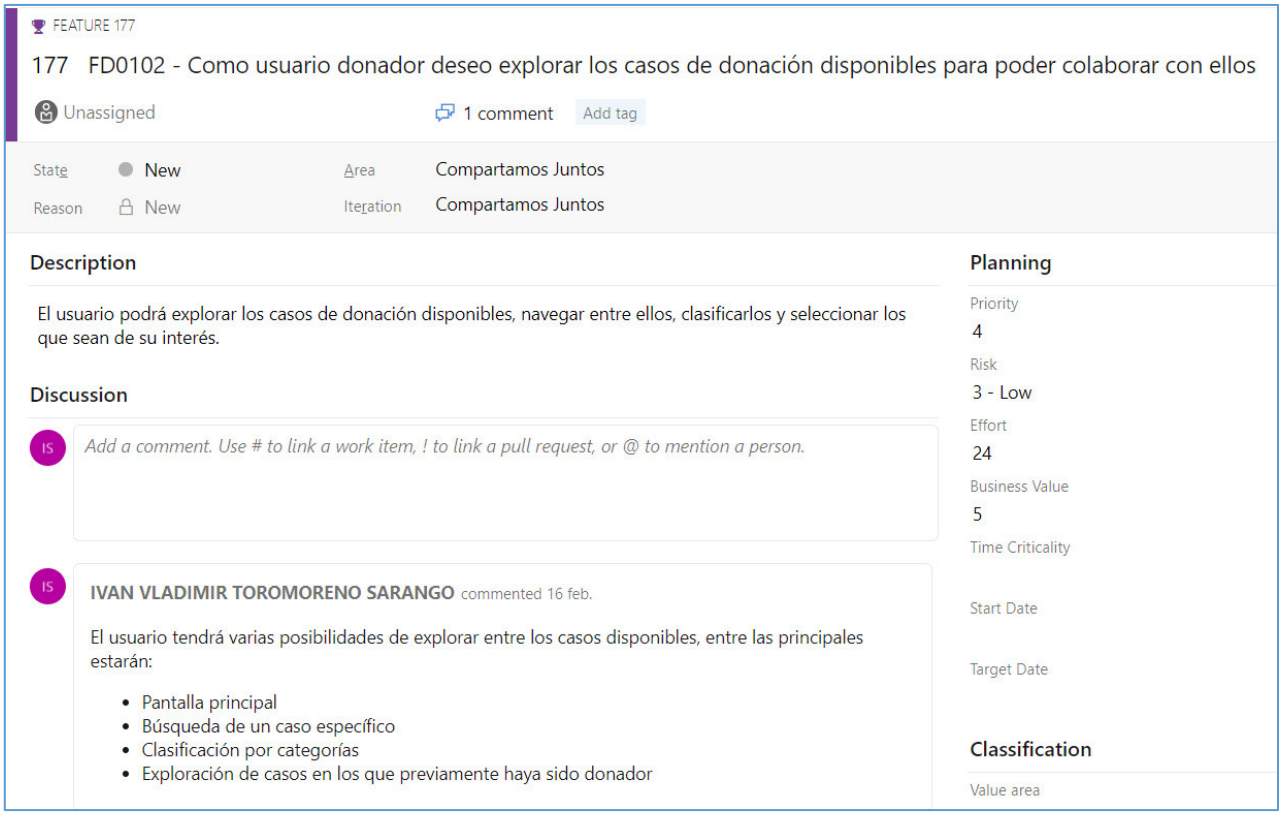

Ilustración 88 - Feature FD0102

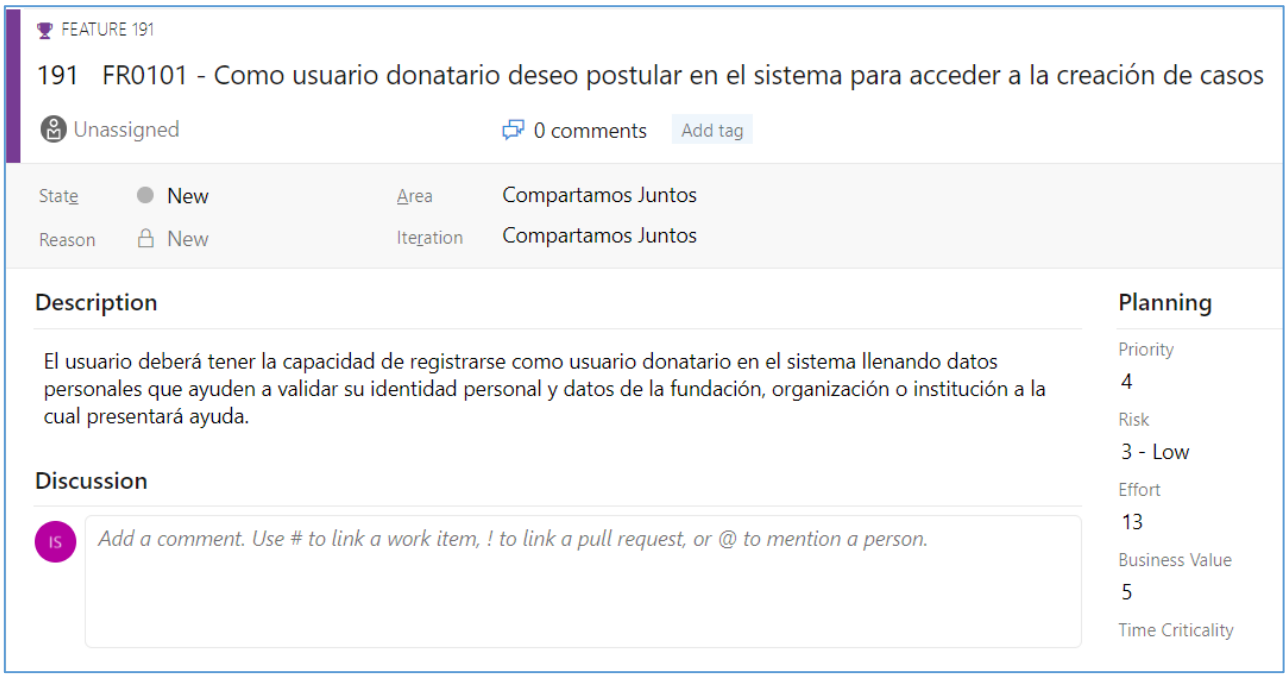

Ilustración 89 - Feature FR0101

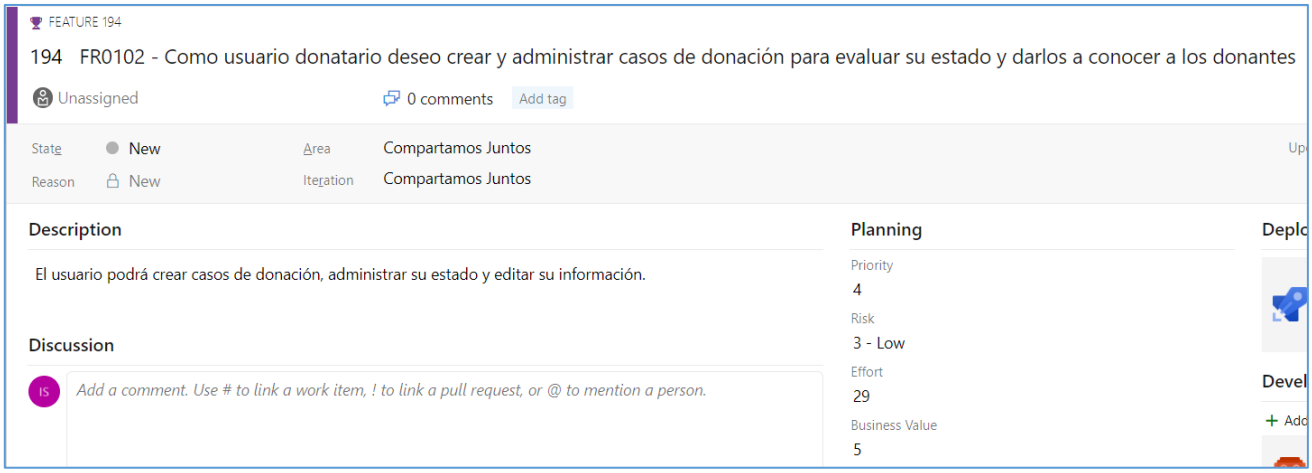

Ilustración 90 - Feature FR0102

6.3 Anexo III: Historias de usuario Sprint 1

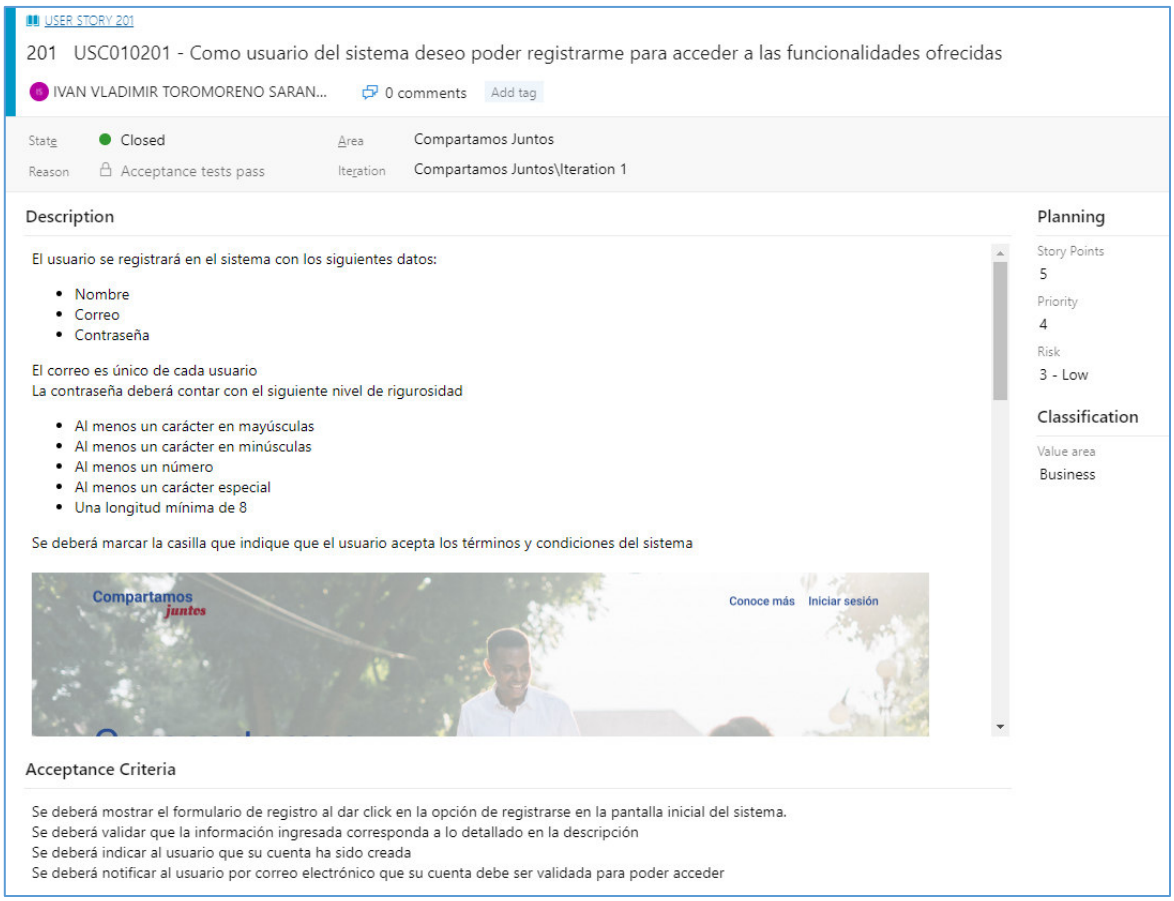

Ilustración 91 - Historia de usuario USC010201 - Extendida

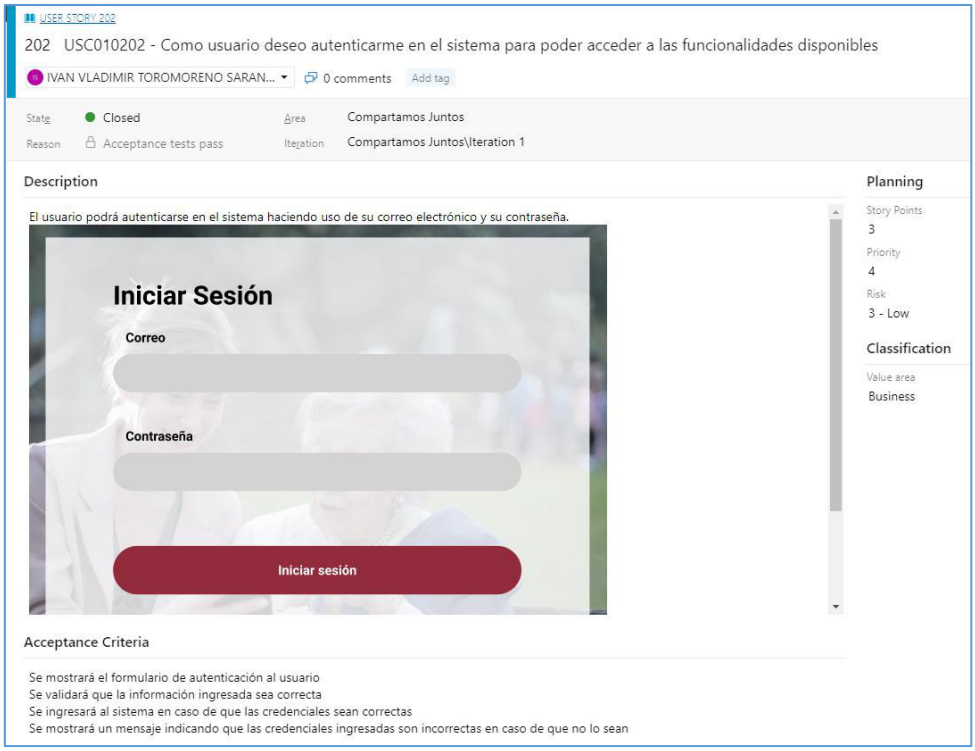

Ilustración 92 - Historia de usuario USC010202 - Extendida

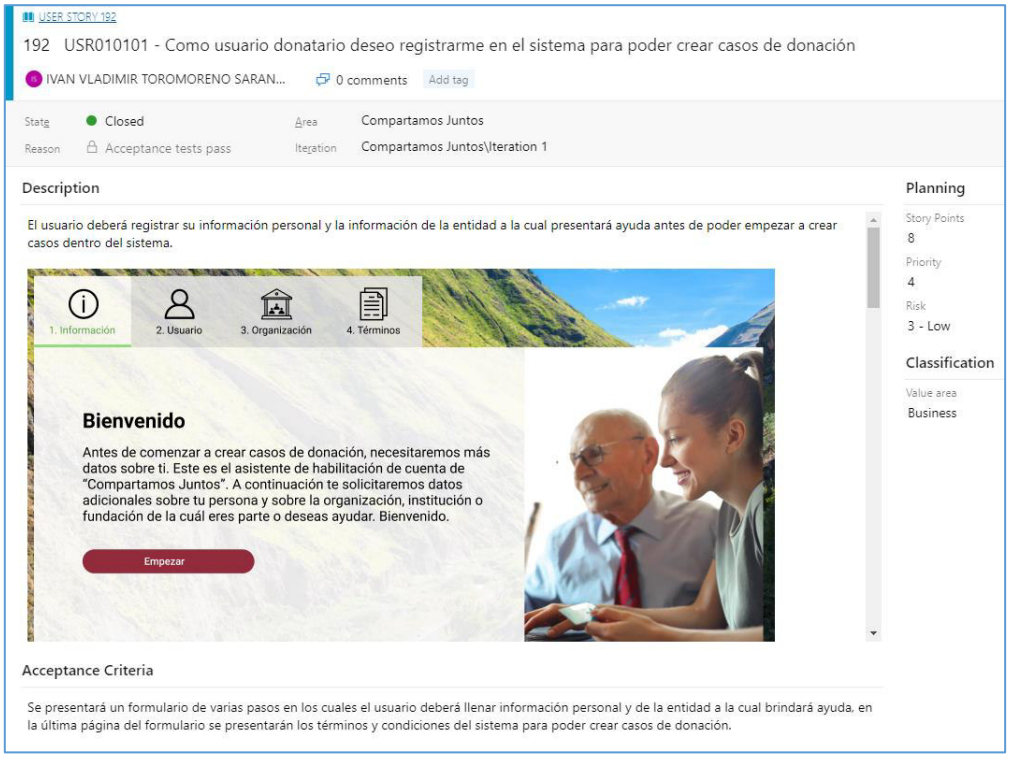

Ilustración 93 - Historia de usuario USR010101 - Extendida

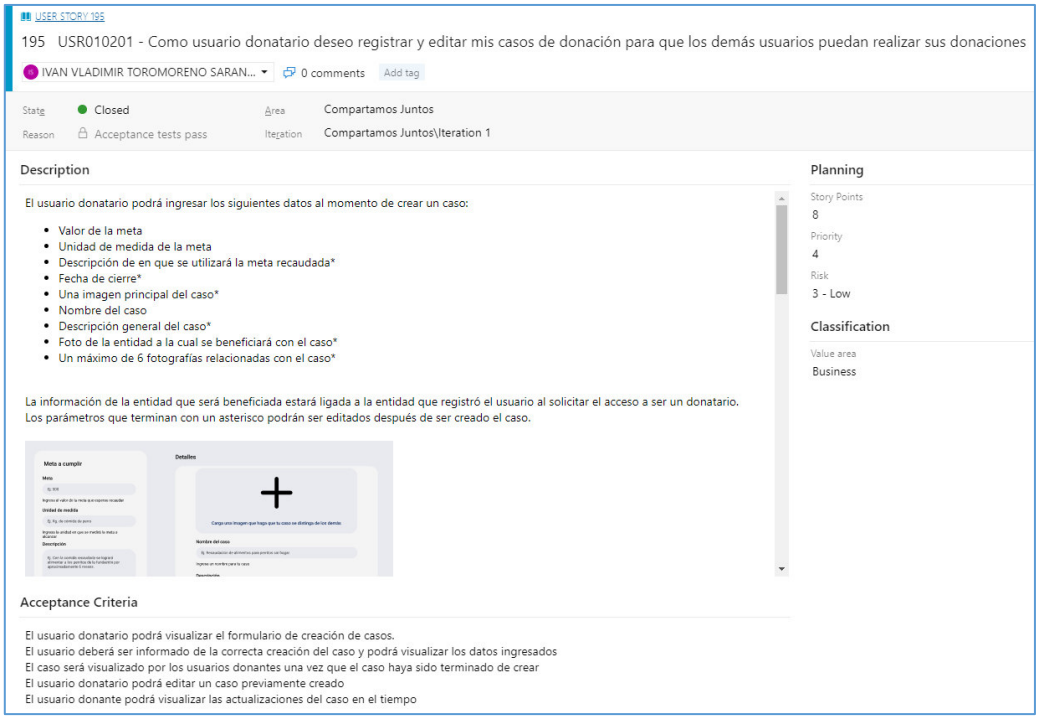

Ilustración 94 - Historia de usuario USR010201 - Extendida

6.4 Anexo IV: Historias de usuario Sprint 2

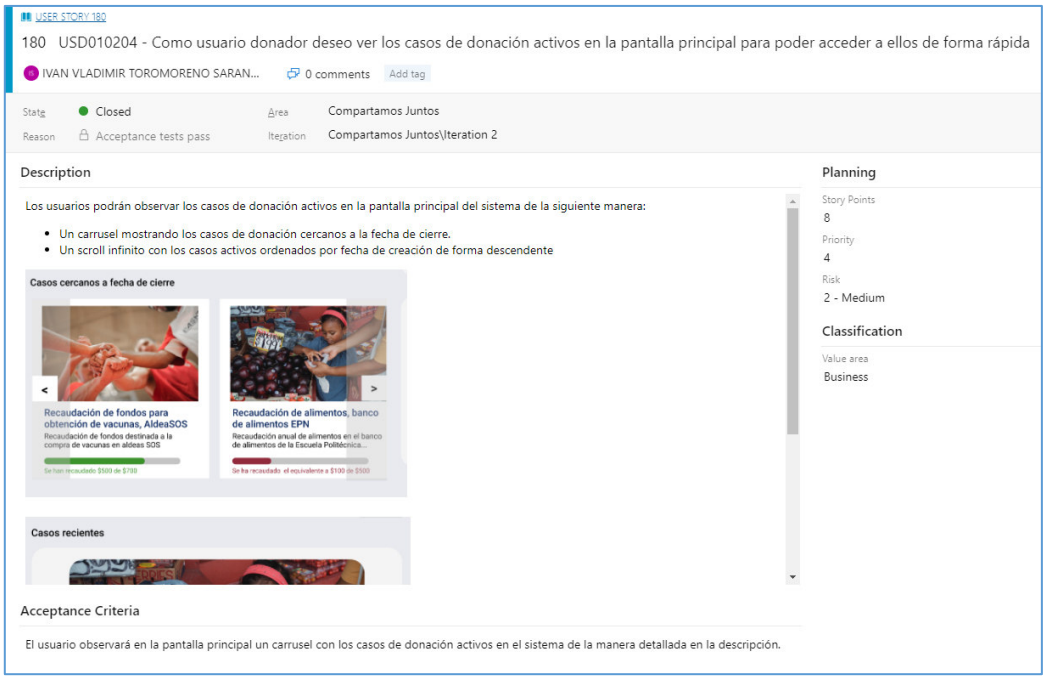

Ilustración 95 - Historia de usuario USD010204 - Extendida

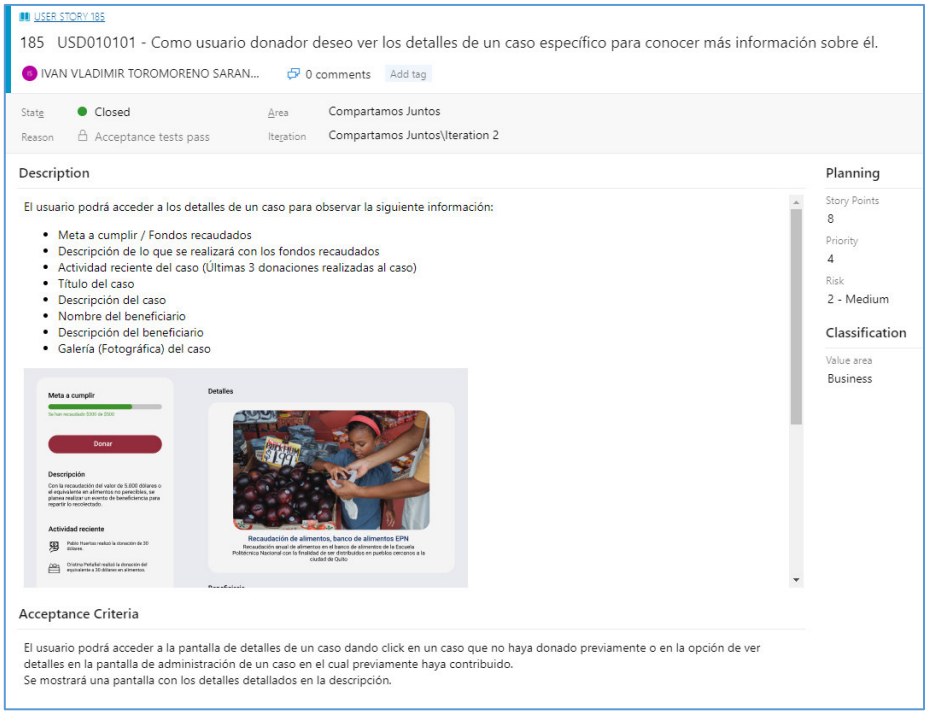

Ilustración 96 - Historia de usuario USD010101 - Extendida

6.5 Anexo V: Historias de usuario Sprint 3

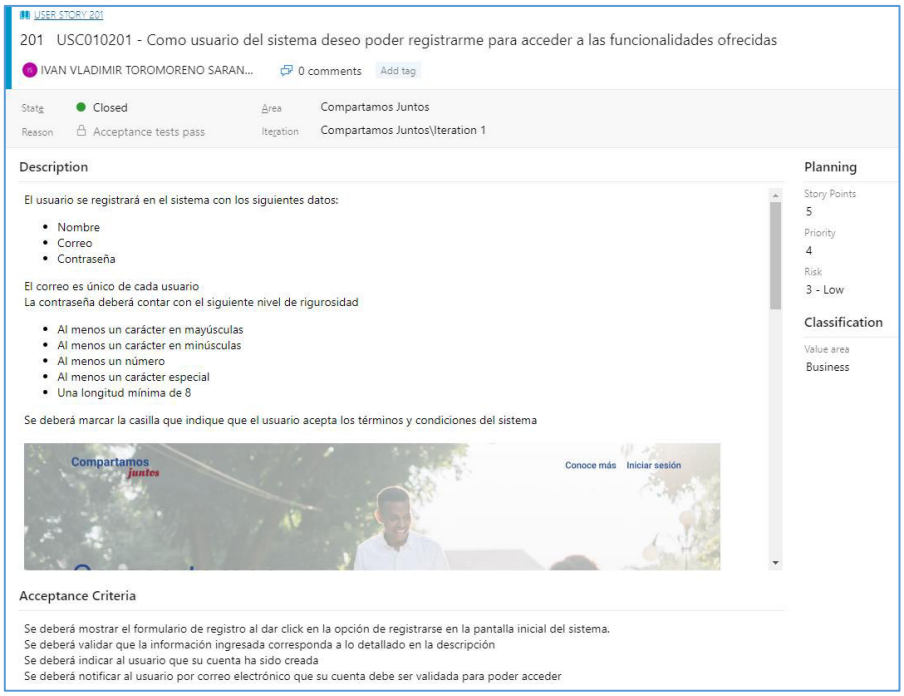

Ilustración 97 - Historia de usuario USC010201 - Extendida

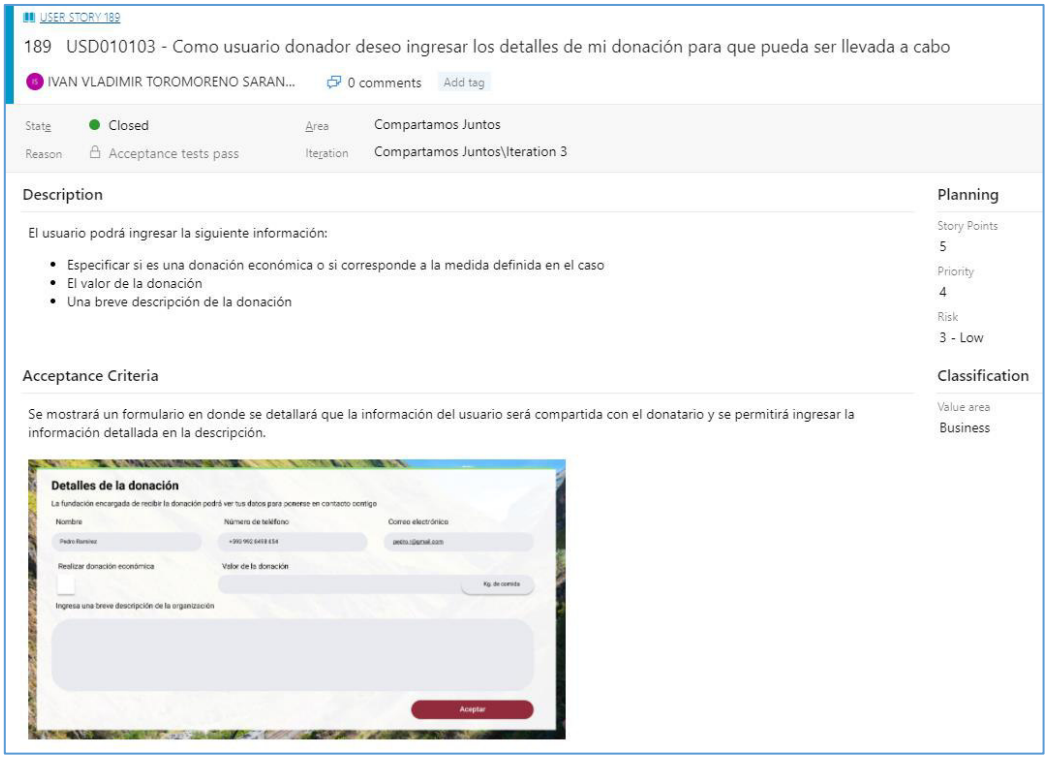

Ilustración 98 - Historia de usuario USD010103 - Extendida

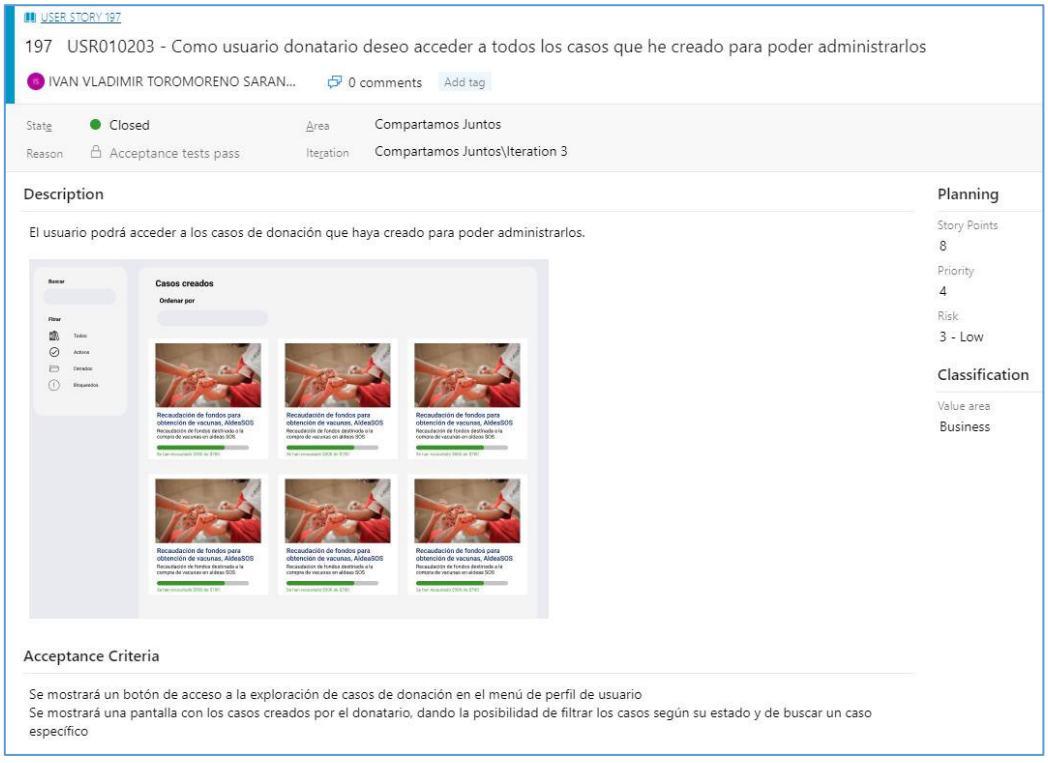

Ilustración 99 - Historia de usuario USR010203 - Extendida

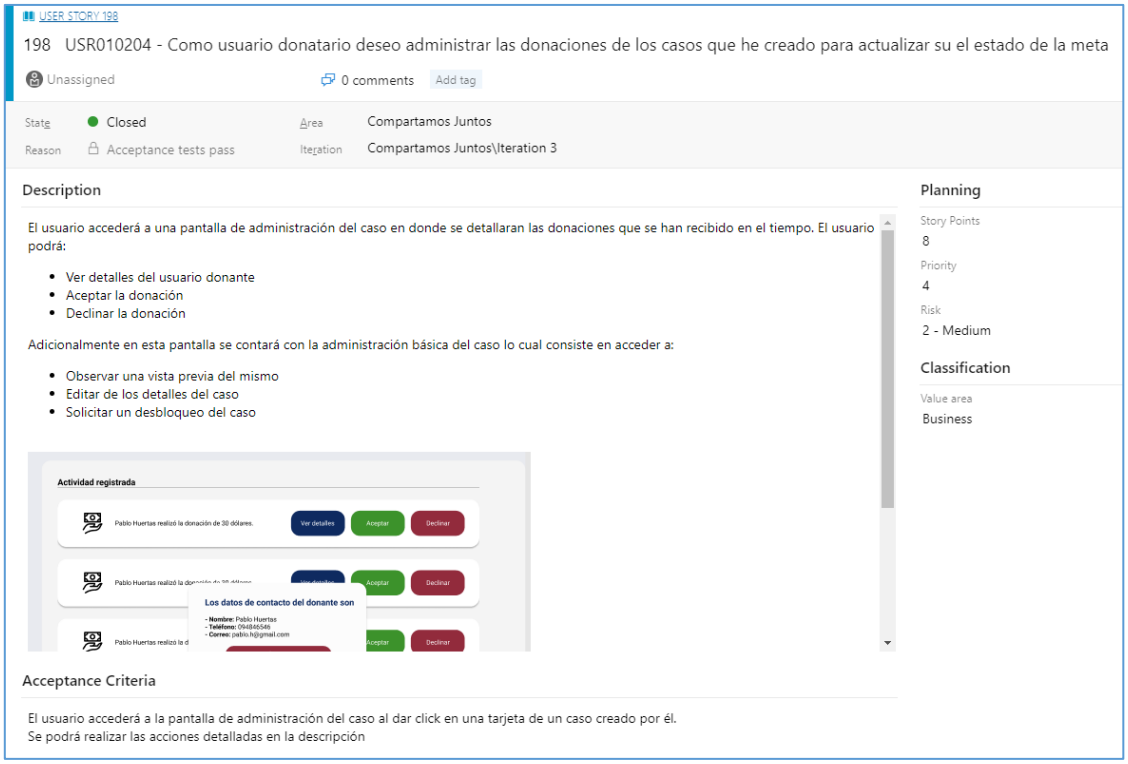

Ilustración 100 - Historia de usuario USR010204 - Extendida

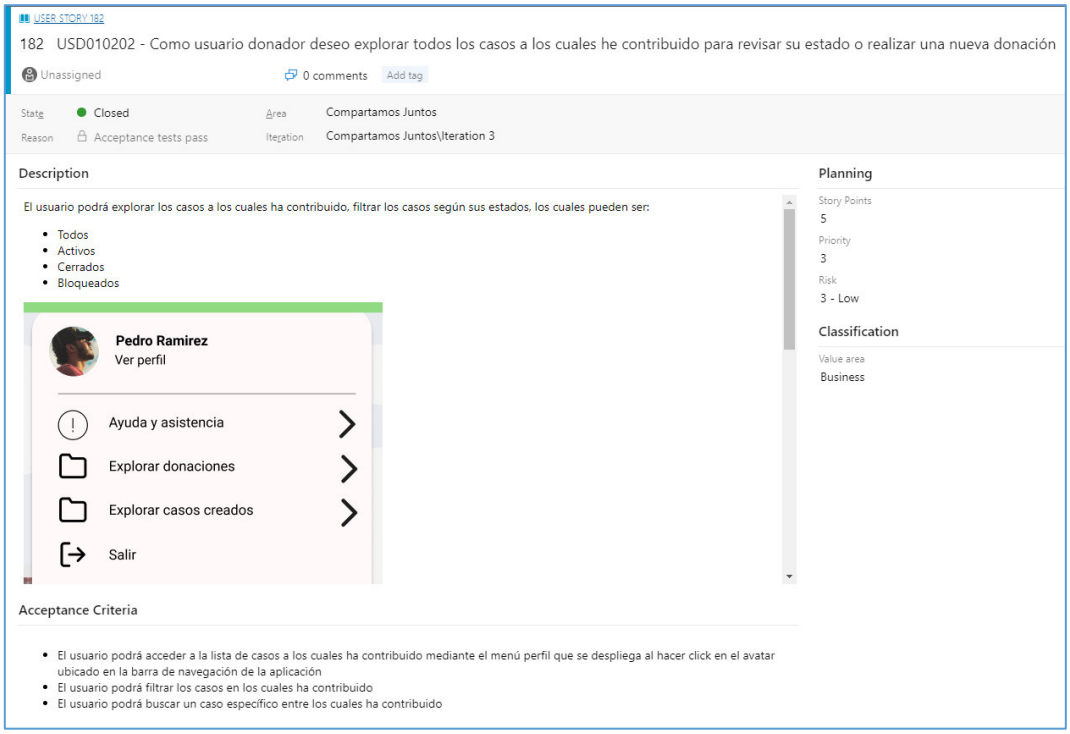

Ilustración 101 - Historia de usuario USD010202 - Extendida

Ilustración 102 - Historia de usuario USD010102 - Extendida

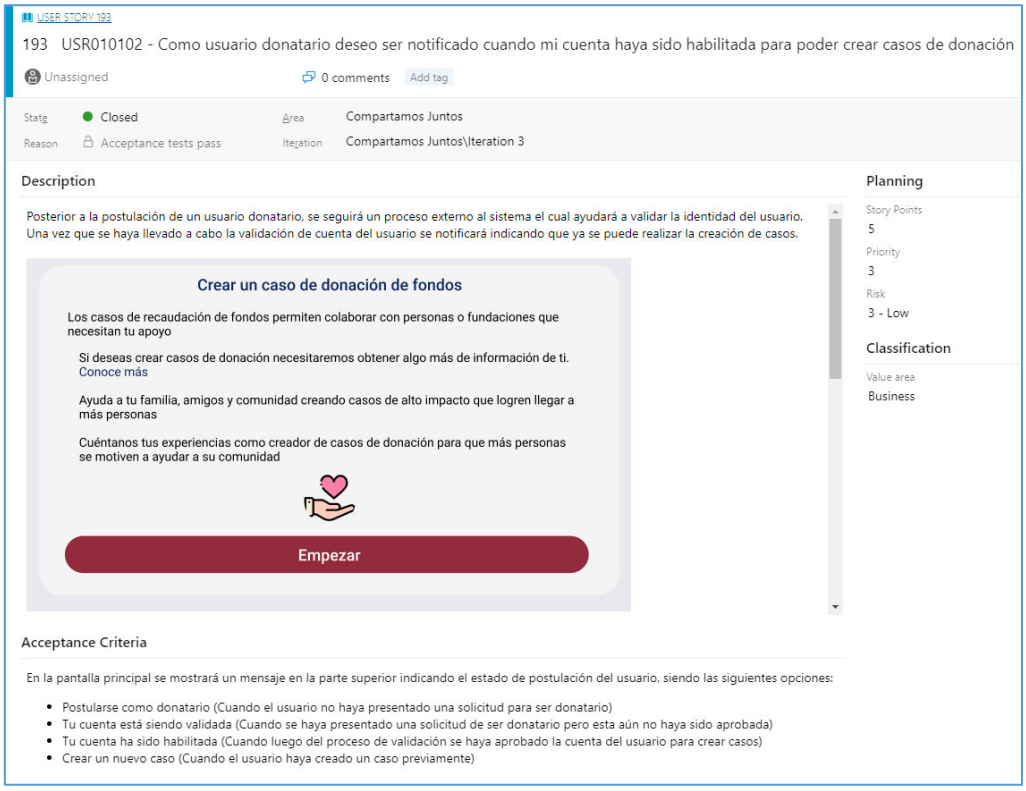

Ilustración 103 - Historia de usuario USR010102 - Extendida

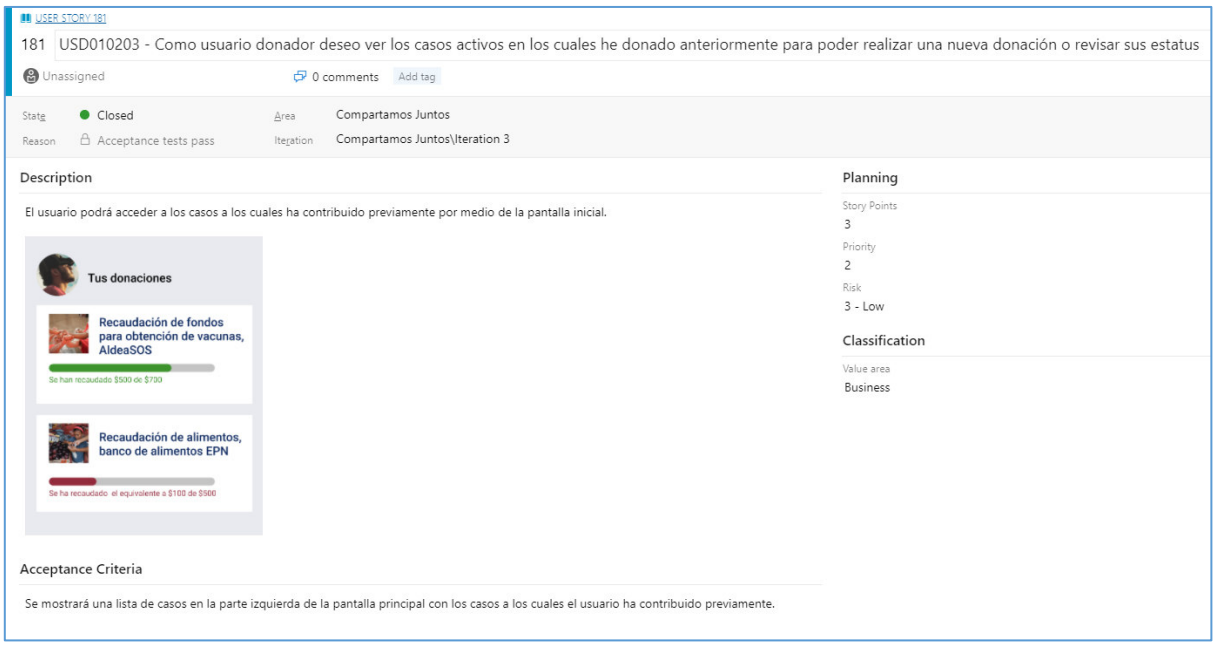

Ilustración 104 - Historia de usuario USD100203 - Extendida

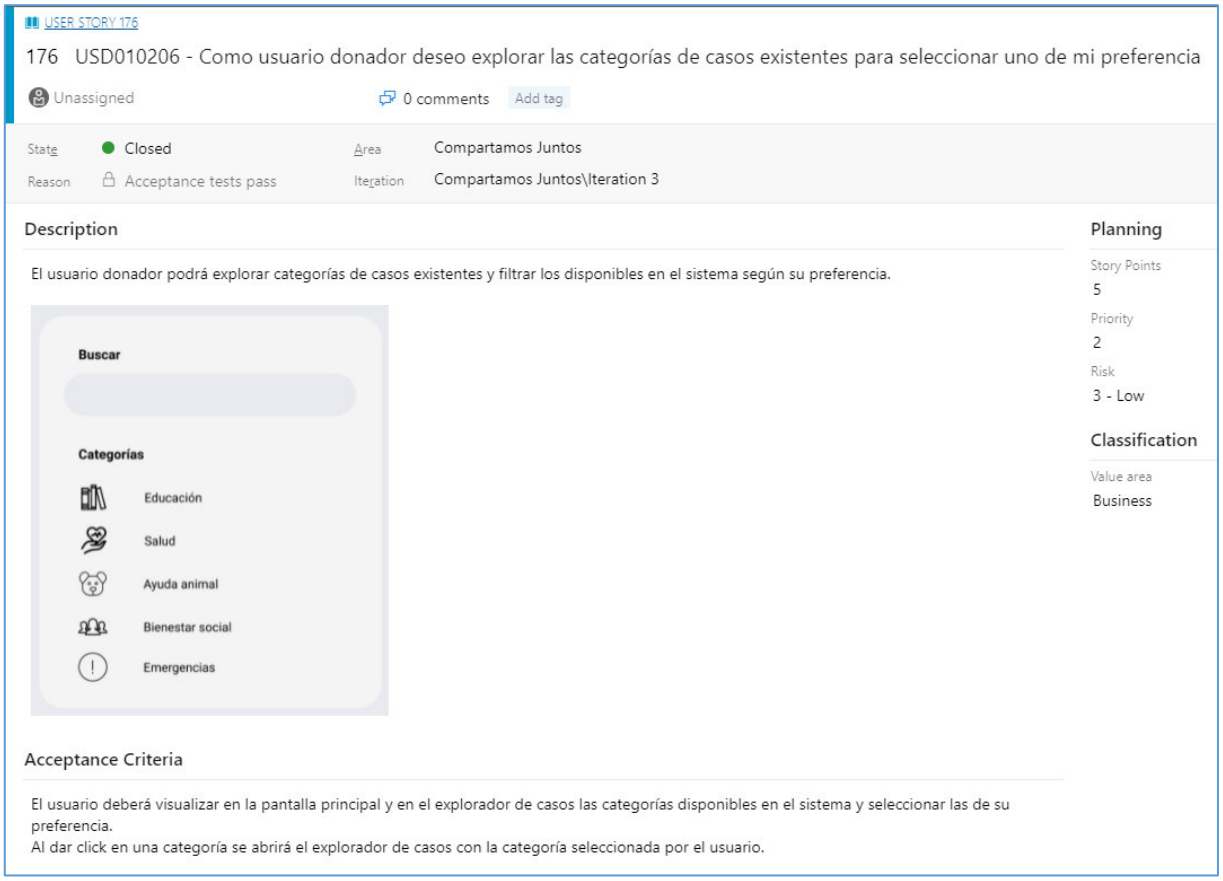

Ilustración 105 - Historia de usuario USD010206 - Extendida

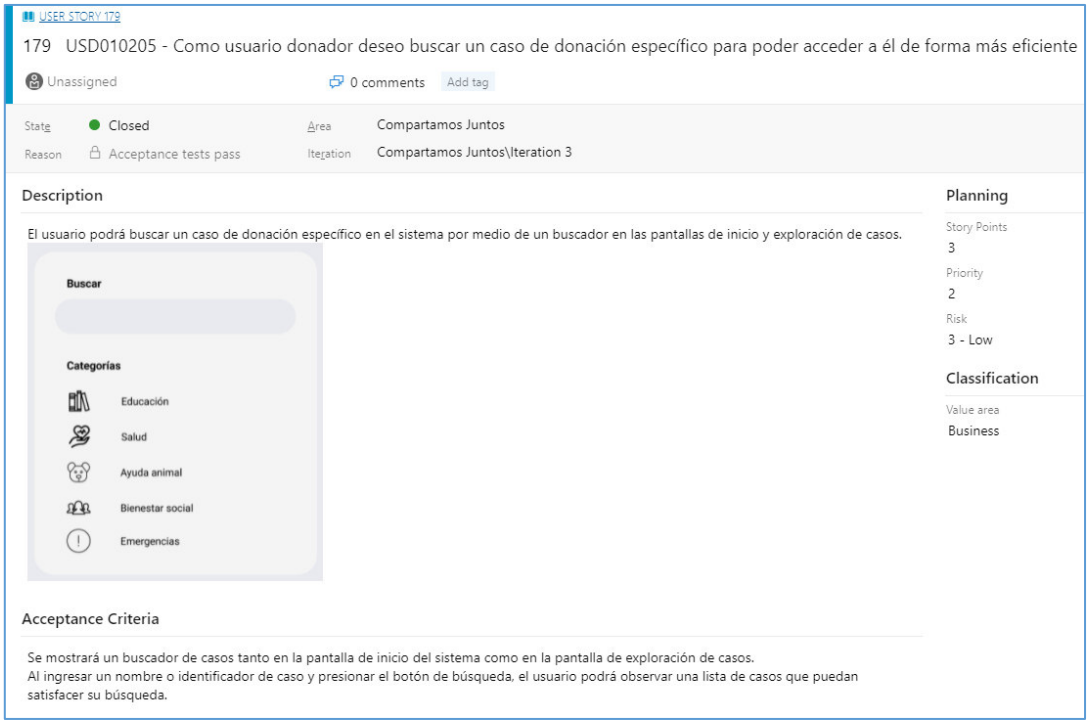

Ilustración 106 - Historia de usuario USD010205 - Extendida

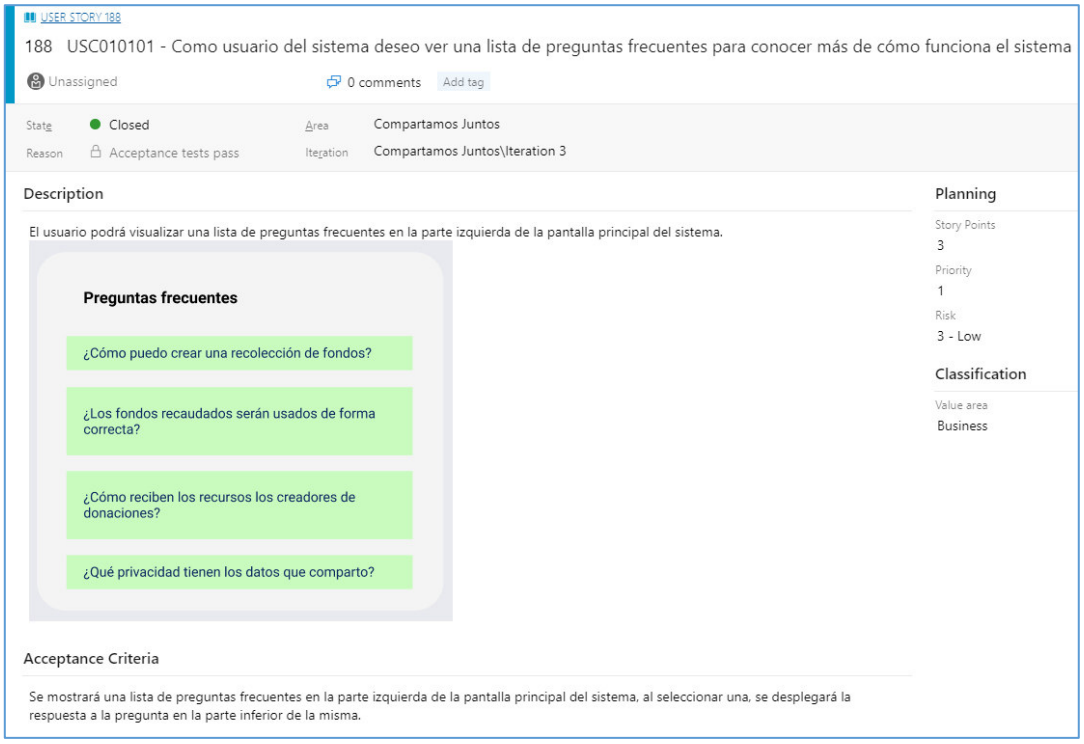

Ilustración 107 - Historia de usuario USC010101 - Extendida

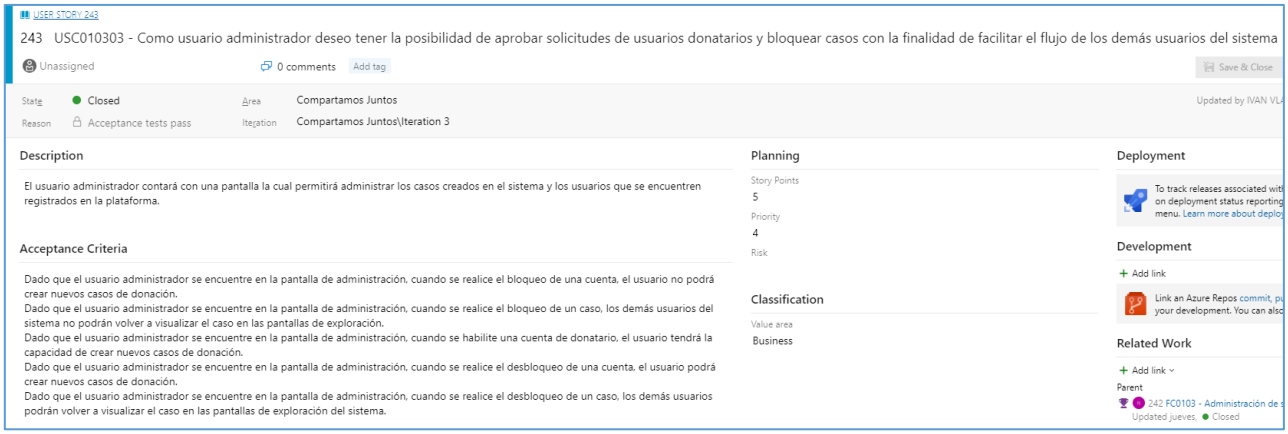

Ilustración 108 - Historia de usuario USC010303 - Extendida

6.6 Anexo VI: Diseño de las interfaces de usuario final

El diseño del sistema fue realizado haciendo uso de la herramienta Figma.

Ilustración 109 - Diseño UI - Pantalla de inicio

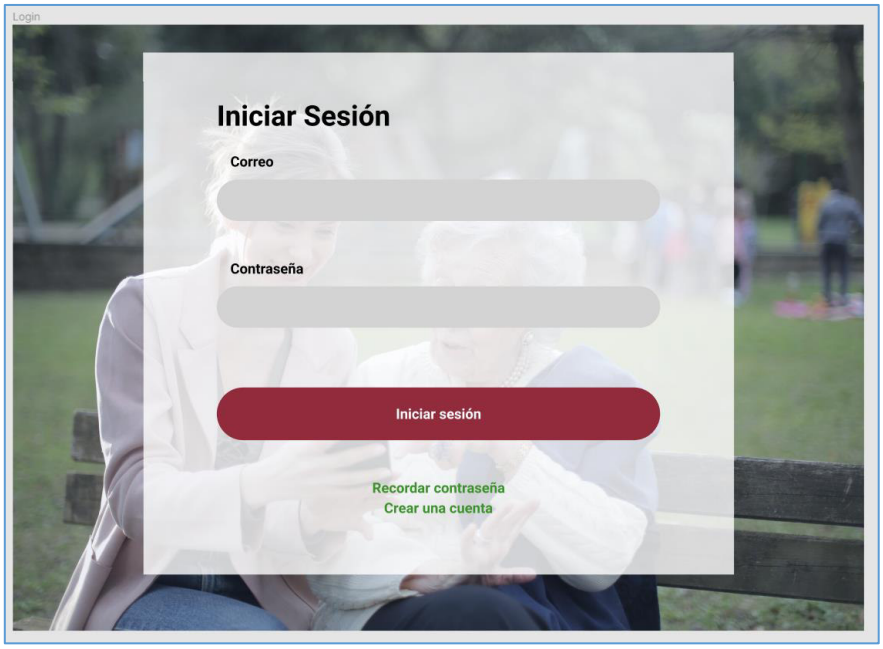

Ilustración 110 - Diseño UI - Pantalla de inicio de sesión

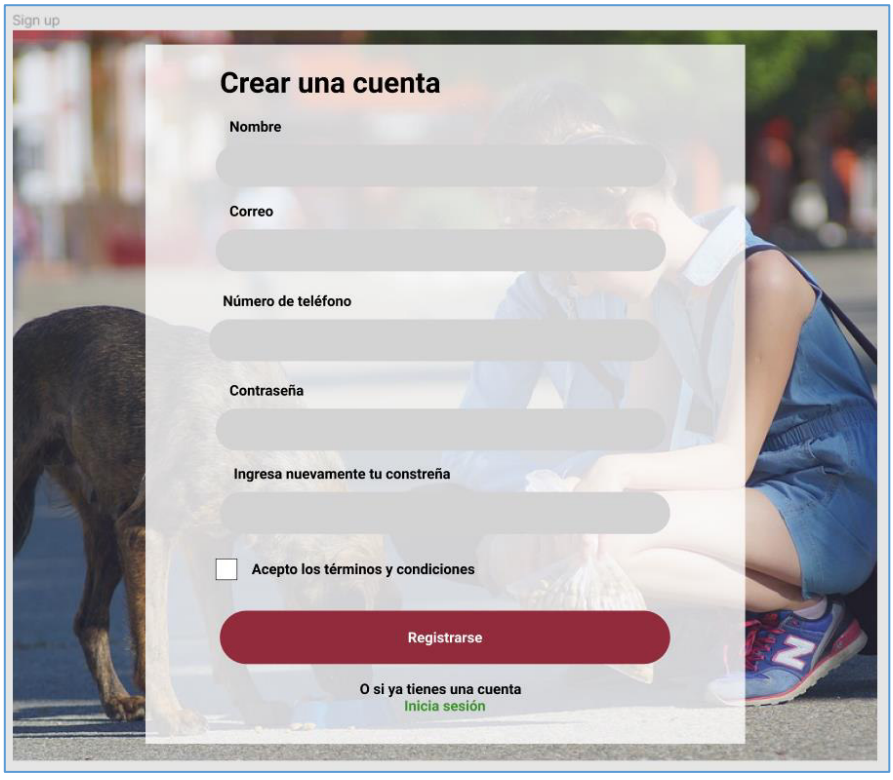

Ilustración 111 - Diseño UI - Pantalla de registro

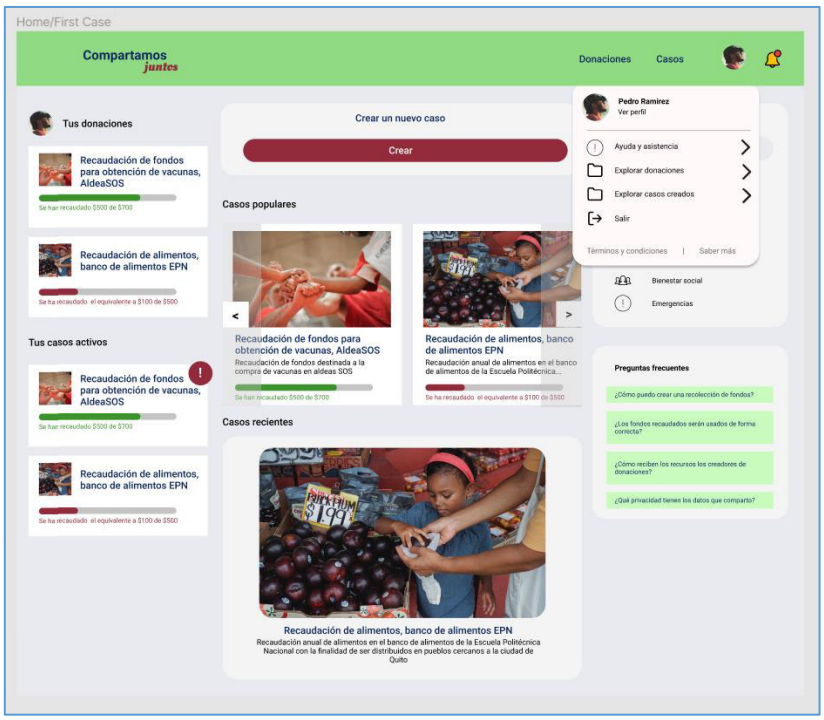

Ilustración 112 - Diseño UI - Pantalla principal

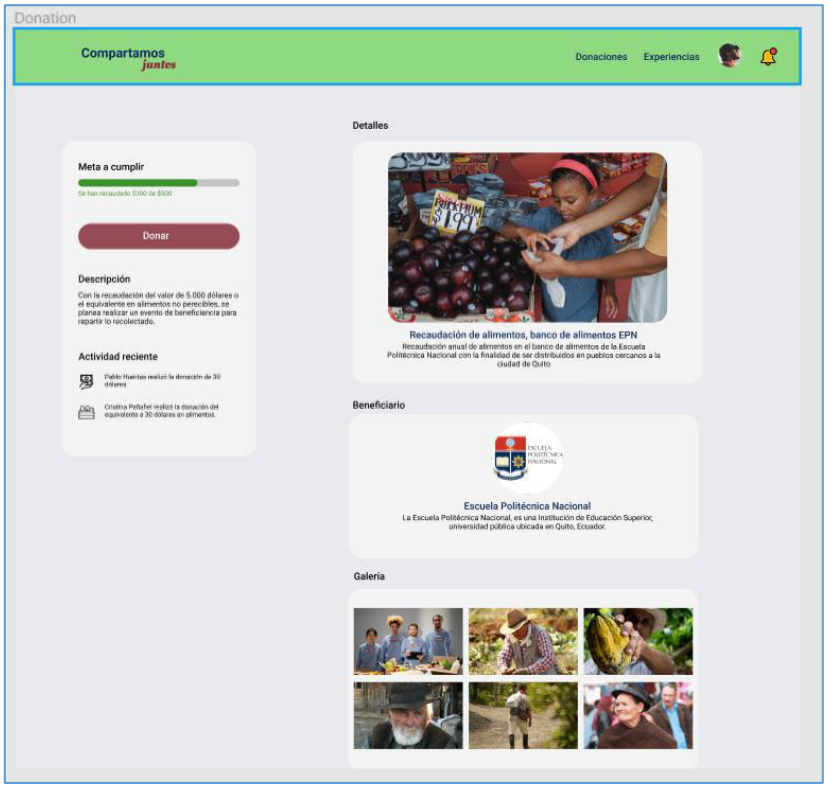

Ilustración 113 - Diseño UI - Pantalla de detalle de casos

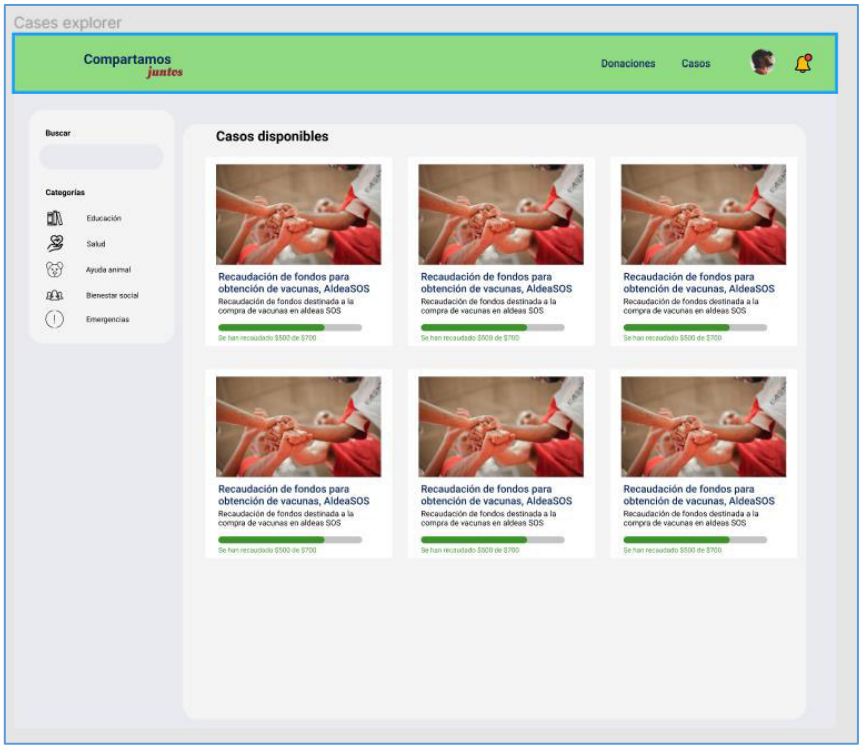

Ilustración 114 - Diseño UI - Pantalla genérica de exploración

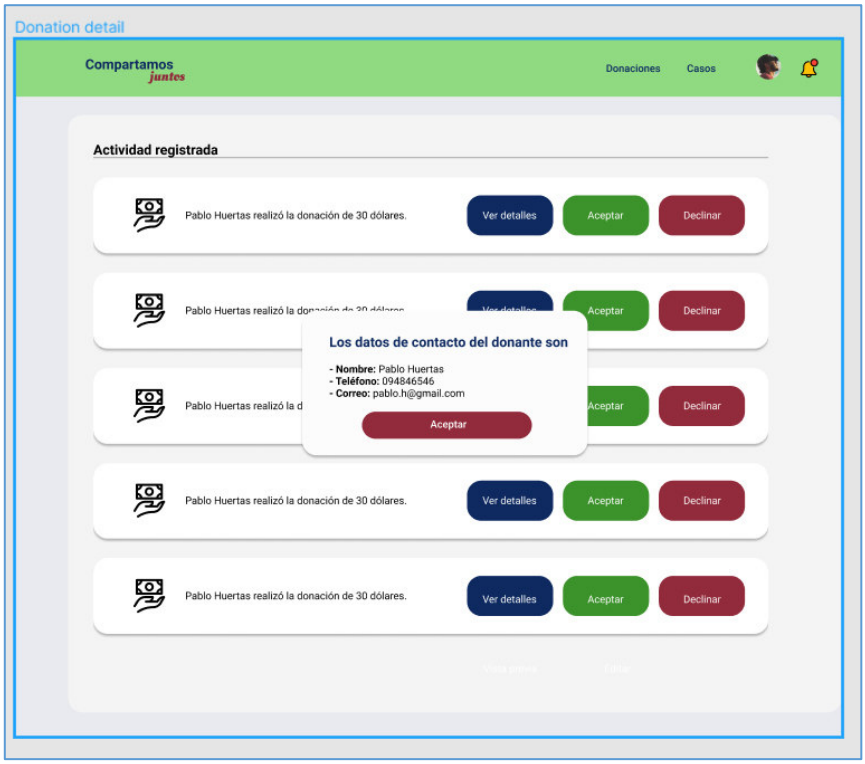

Ilustración 115 - Diseño UI - Pantalla general de administración

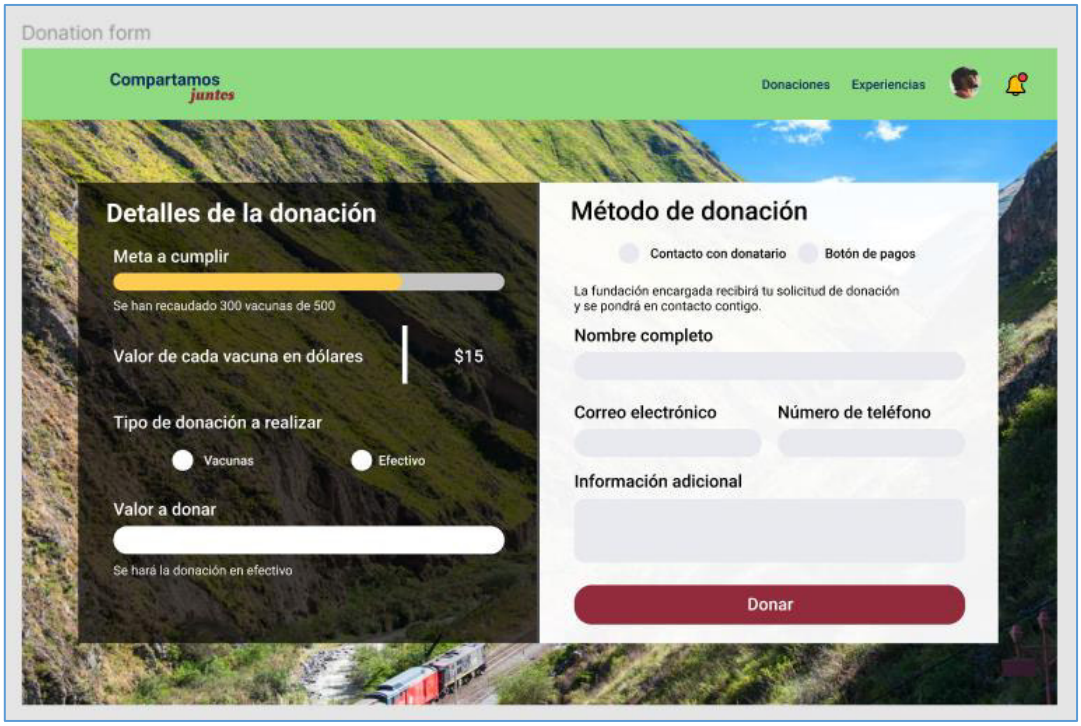

Ilustración 116 - Diseño UI - Pantalla de donaciones

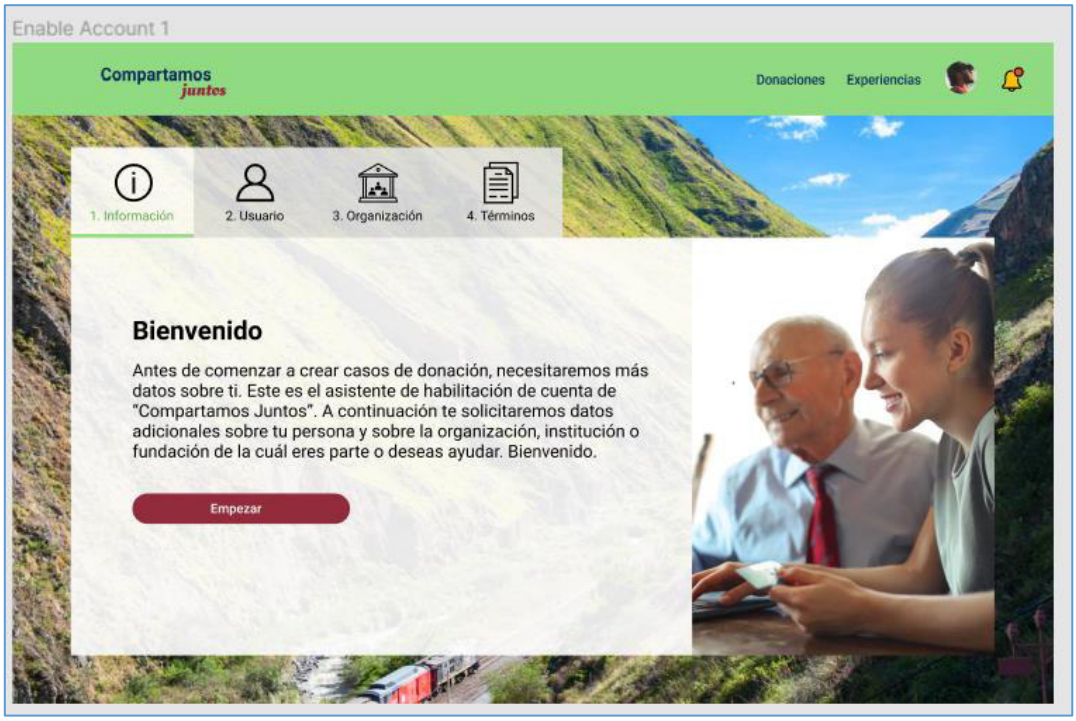

Ilustración 117 - Diseño UI - Pantalla de registro de donatarios 1

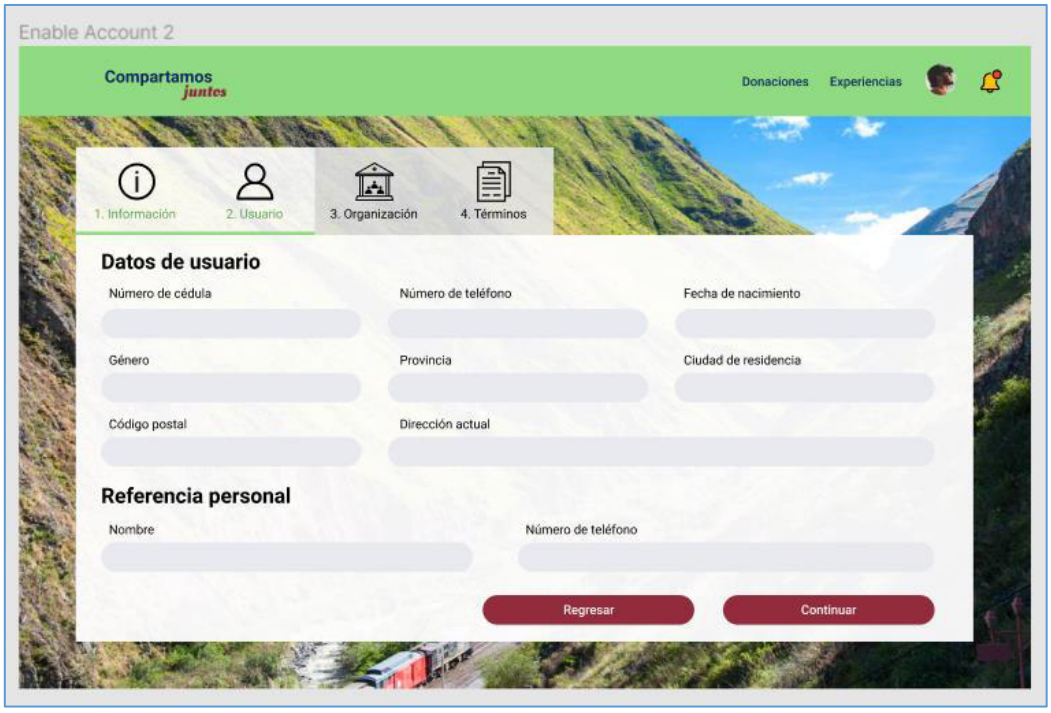

Ilustración 118 - Diseño UI - Pantalla de registro de donatarios 2

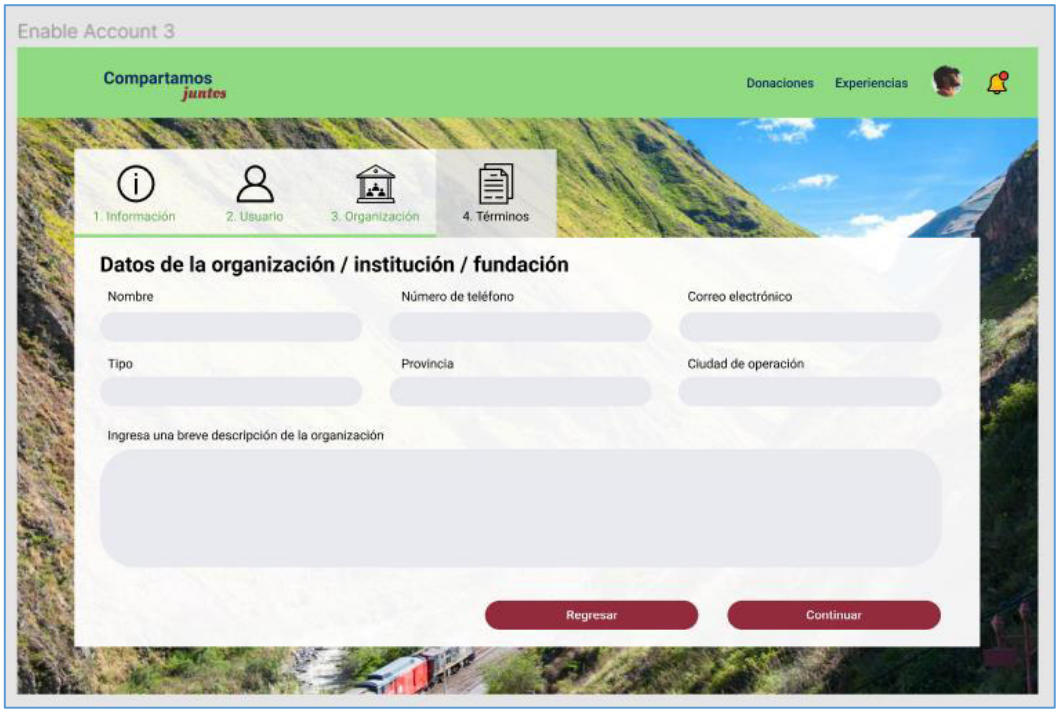

Ilustración 119 - Diseño UI - Pantalla de registro de donatarios 3

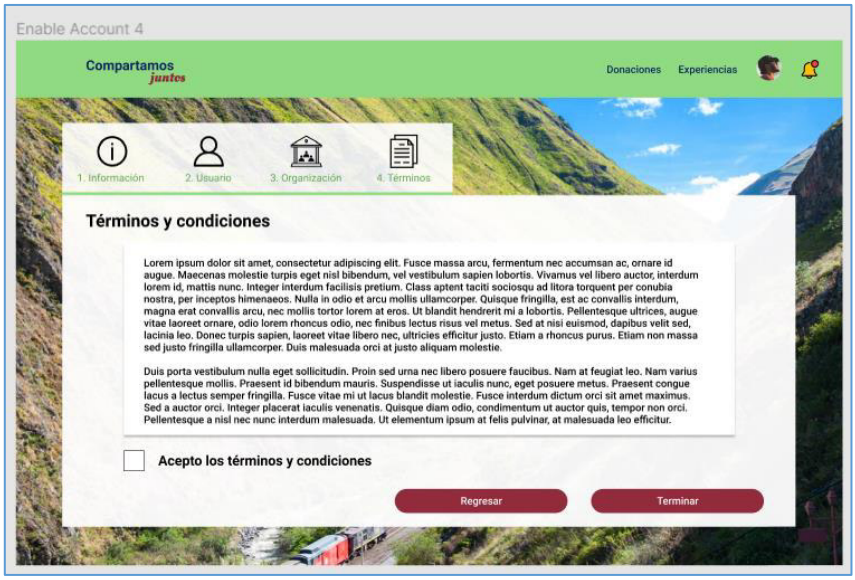

Ilustración 120 - Diseño UI - Pantalla de registro de donatarios 4

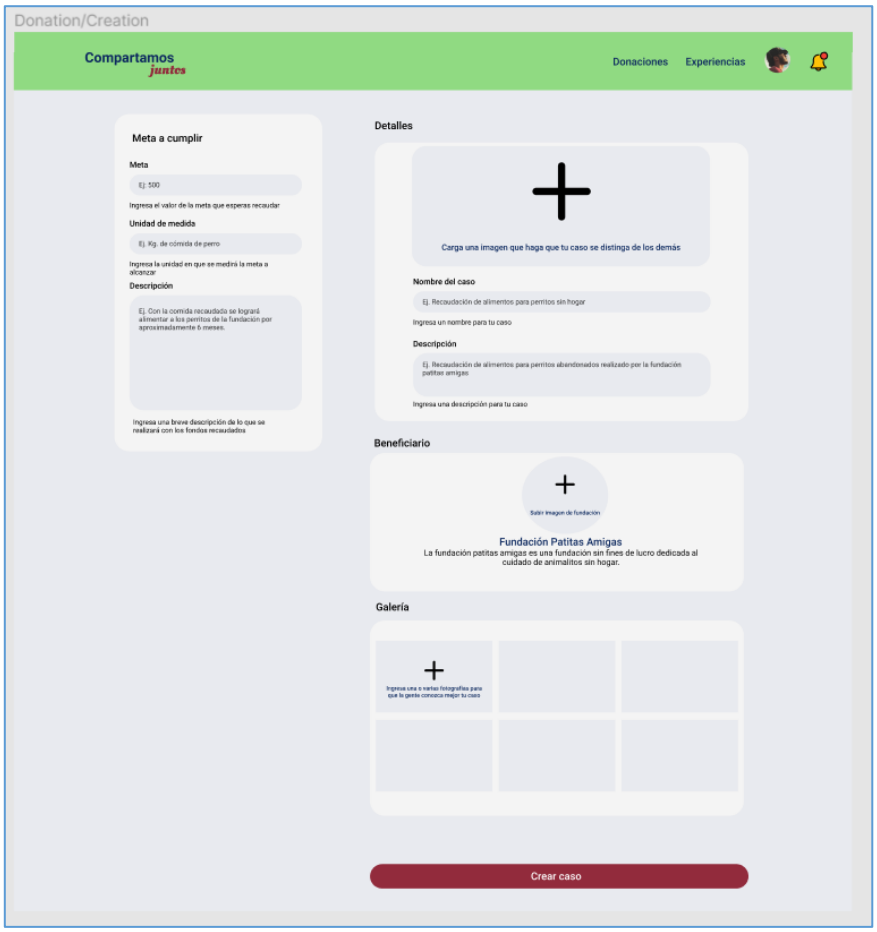

Ilustración 121 - Diseño UI - Pantalla de registro de casos

6.7 Anexo VII: Pruebas funcionales

Las pruebas funcionales fueron registradas haciendo uso de la herramienta Trello.

Ilustración 122 - Prueba funcional - Login de usuario

Ilustración 123 - Prueba funcional - Probar registro de un usuario

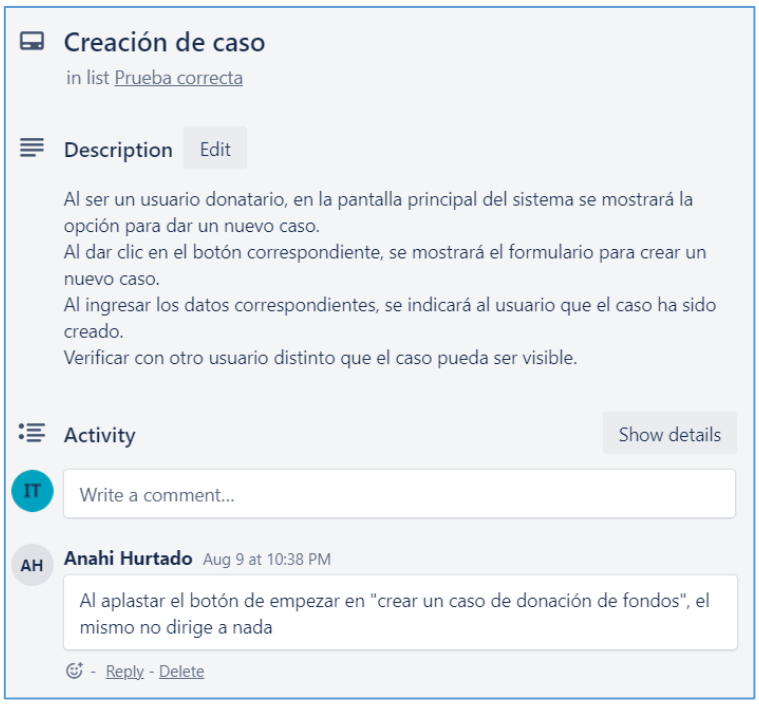

Ilustración 124 - Prueba funcional - Creación de caso

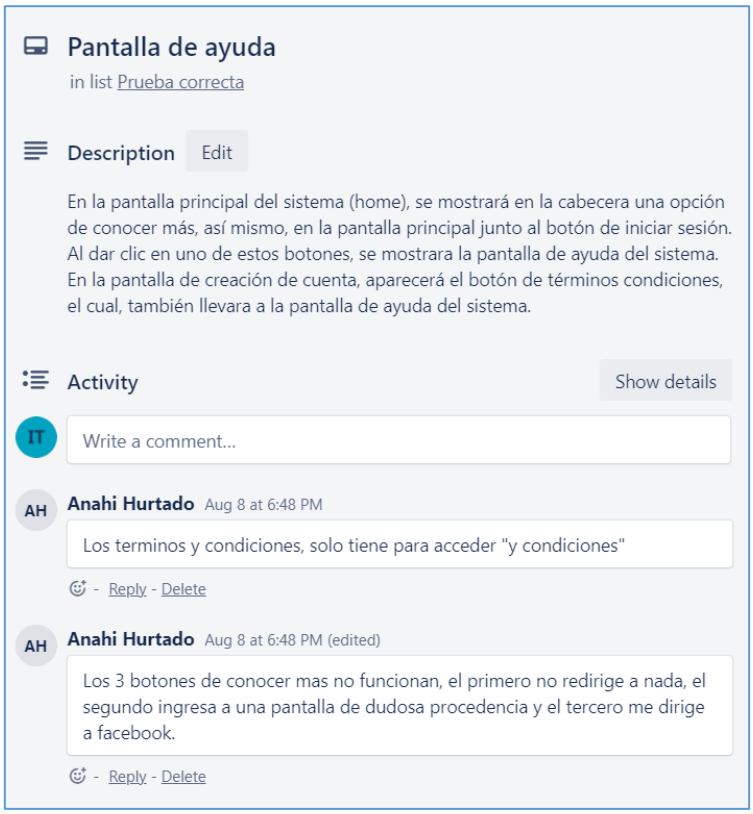

Ilustración 125 - Prueba funcional - Pantalla de ayuda

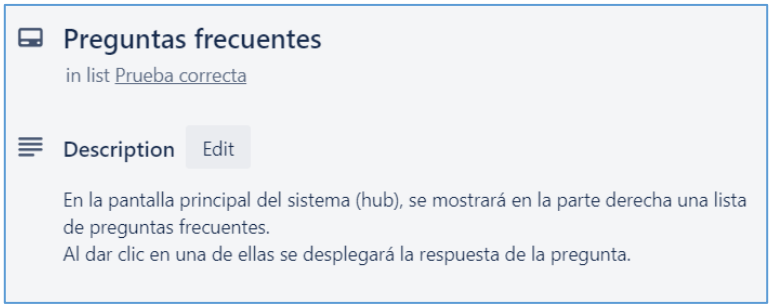

Ilustración 126 - Prueba funcional - Preguntas frecuentes

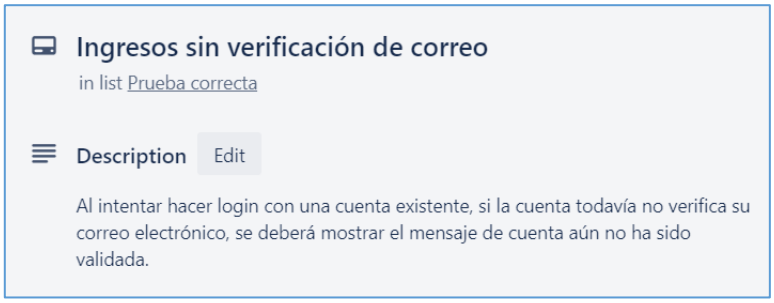

Ilustración 127 - Prueba funcional - Ingresos sin verificación de correo

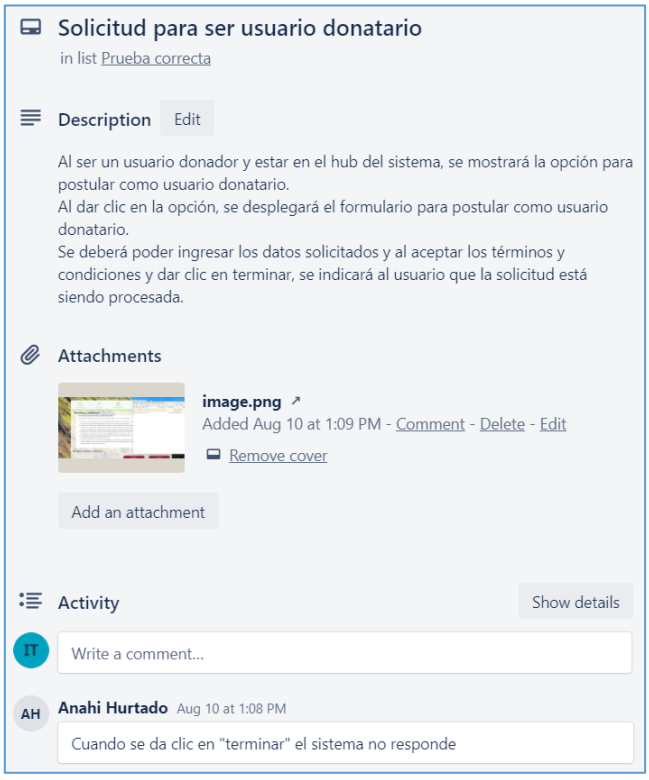

Ilustración 128 - Prueba funcional - Solicitud para ser usuario donatario

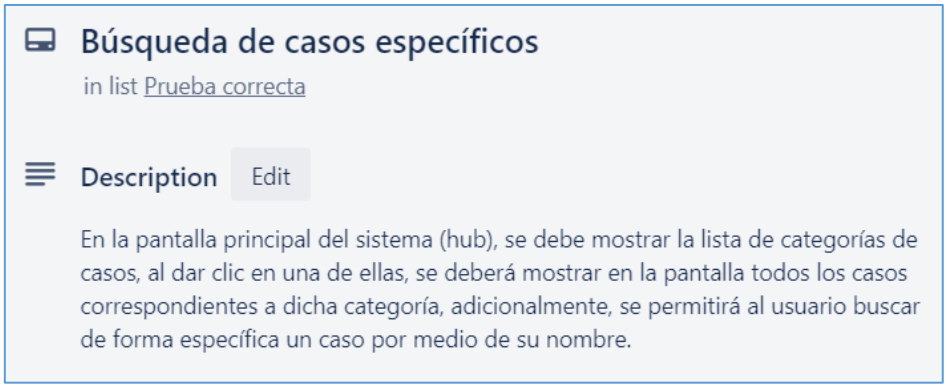

Ilustración 129 - Prueba funcional - Búsqueda de casos específicos

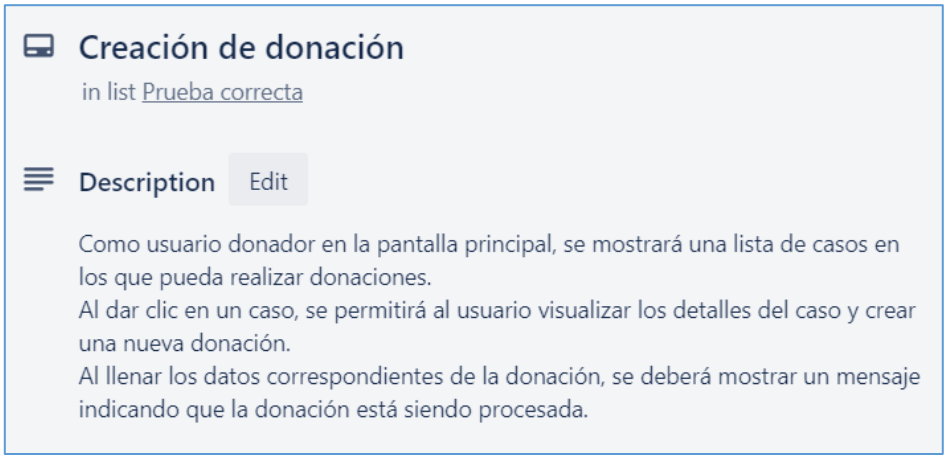

Ilustración 130 - Prueba funcional - Creación de donación

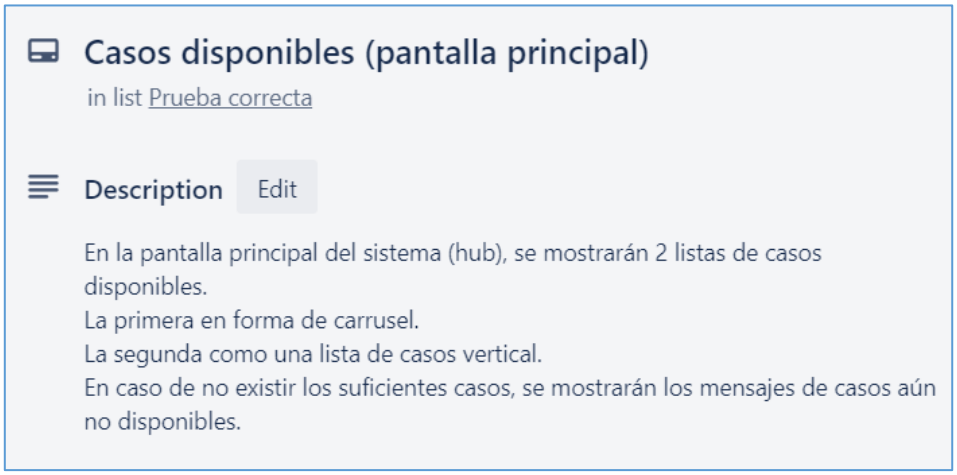

Ilustración 131 - Prueba funcional - Casos disponibles (pantalla principal)

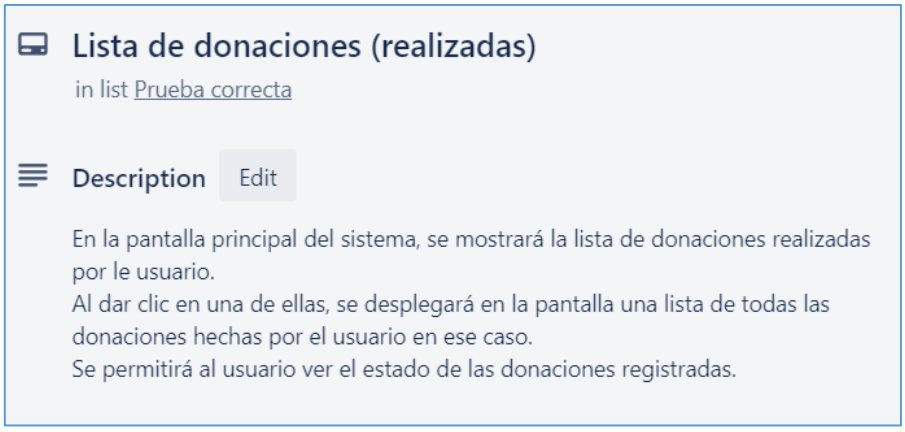

Ilustración 132 - Prueba funcional - Lista de donaciones (realizadas)

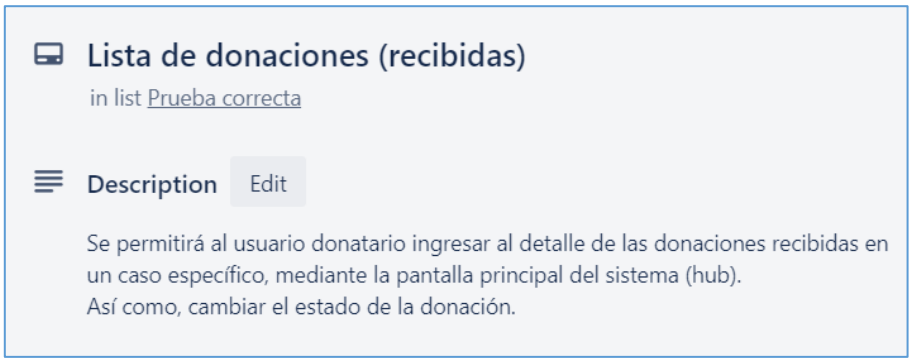

Ilustración 133 - Prueba funcional - Lista de donaciones (recibidas)

6.8 Anexo VIII: Encuesta de usabilidad

Las encuestas realizadas fueron del tipo escala lineal en donde el número 1 representaba que se percibió a la funcionalidad como muy difícil de entender y el número 5 como muy intuitivo de entender.

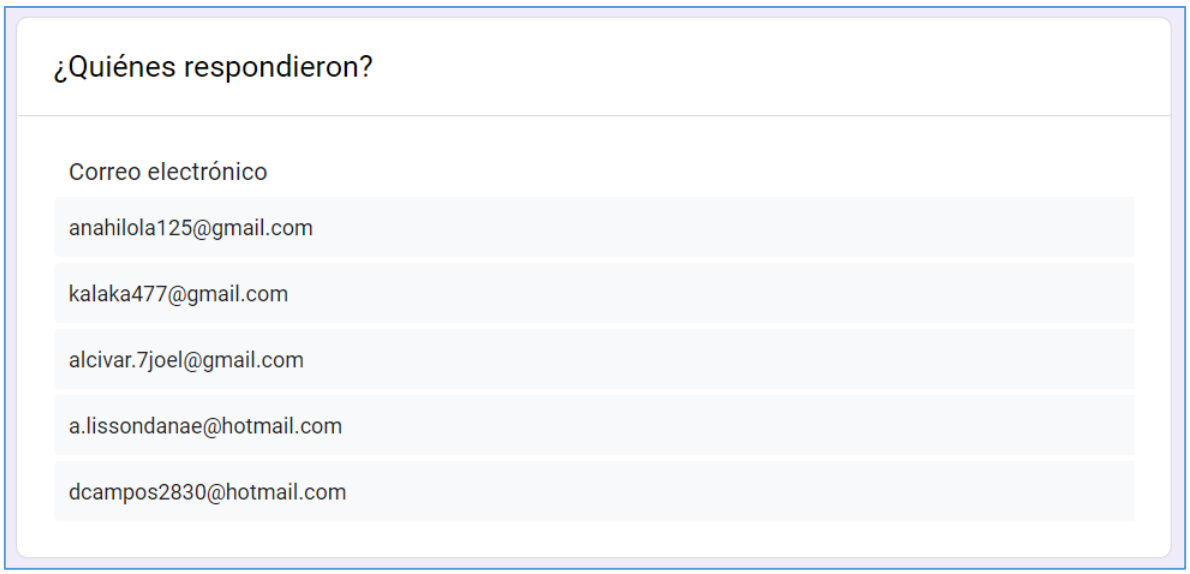

Ilustración 134 - Encuesta - Lista de usuarios encuestados

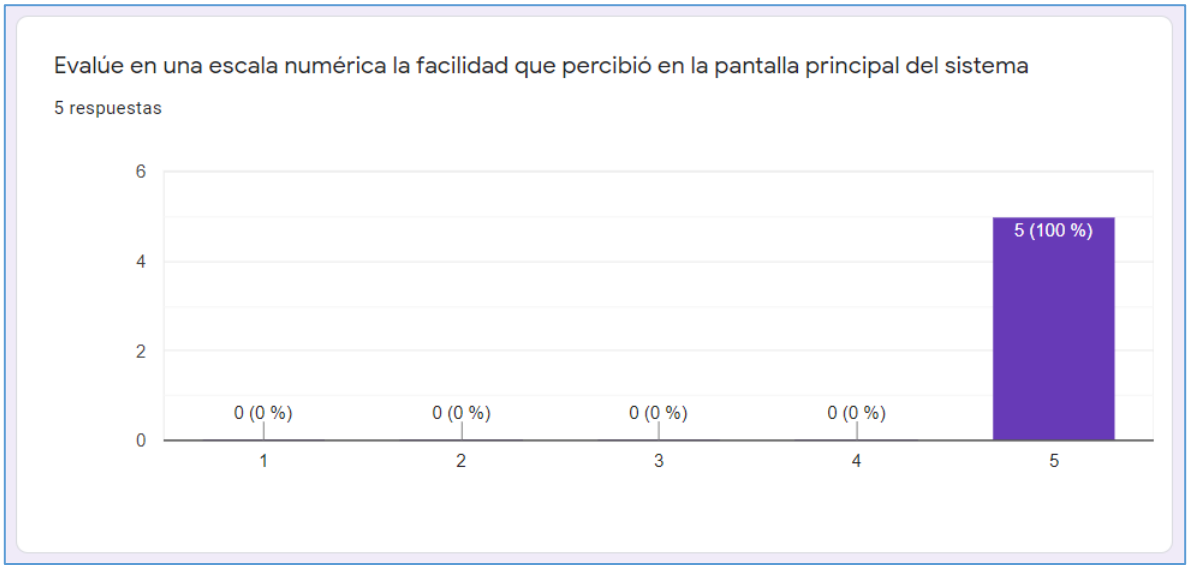

Ilustración 135 - Encuesta - Pregunta 1

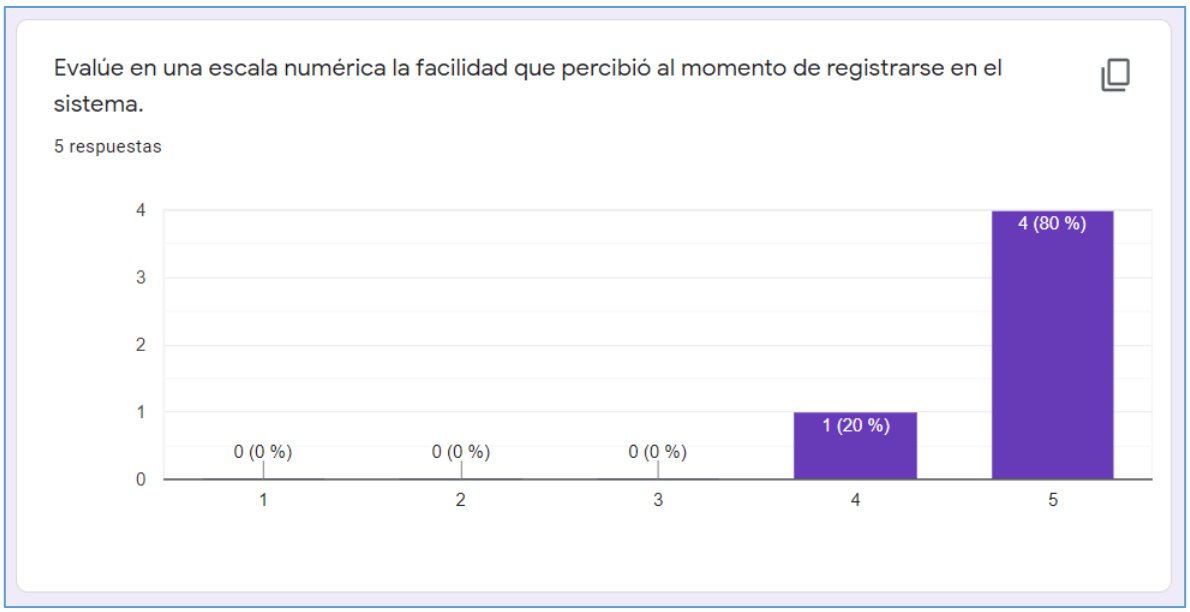

Ilustración 136 - Encuesta - Pregunta 2

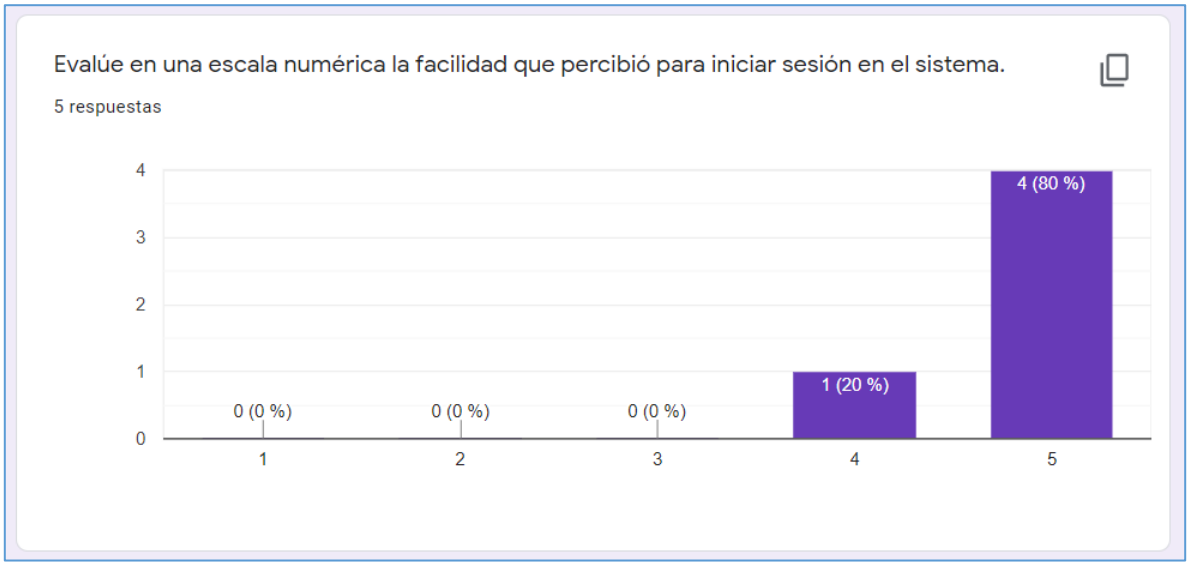

Ilustración 137 - Encuesta - Pregunta 3

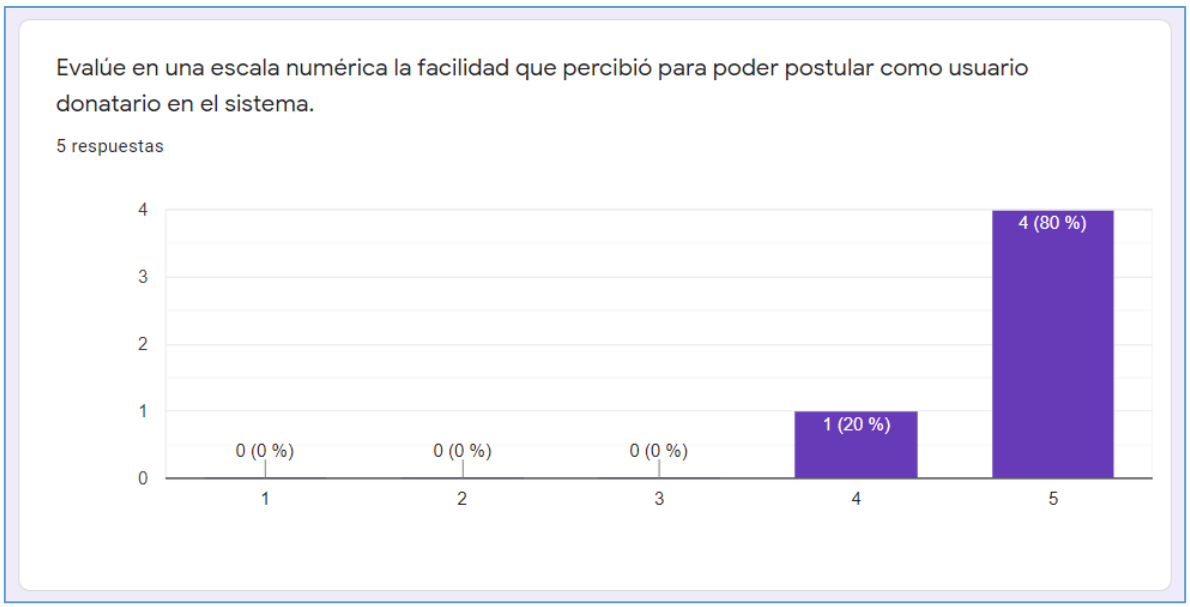

Ilustración 138 - Encuesta - Pregunta 4

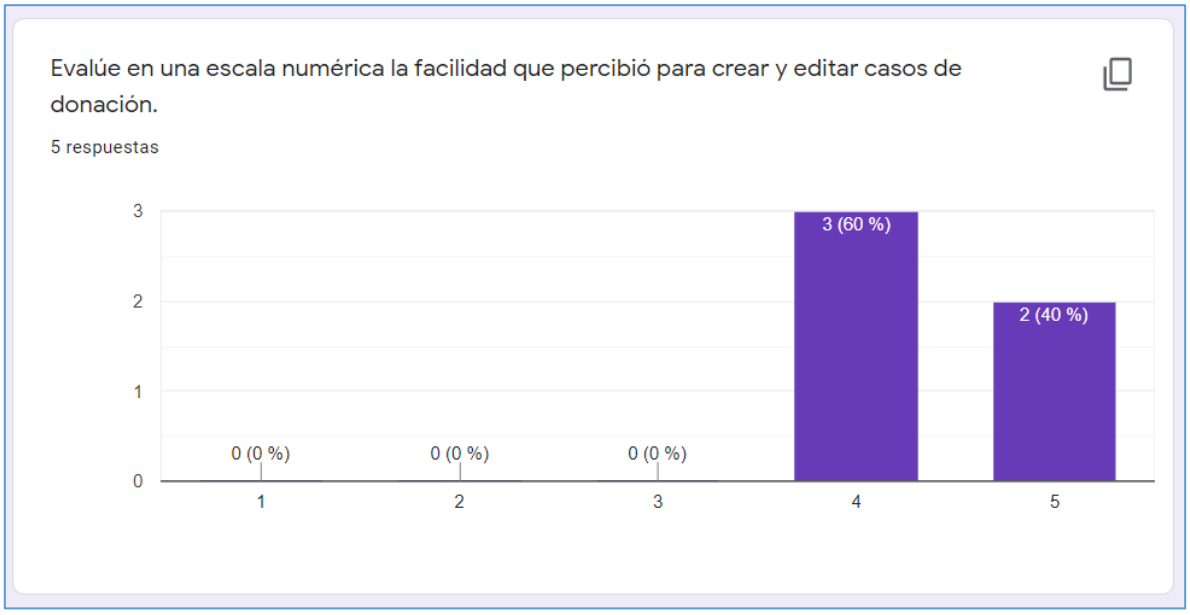

Ilustración 139 - Encuesta - Pregunta 5

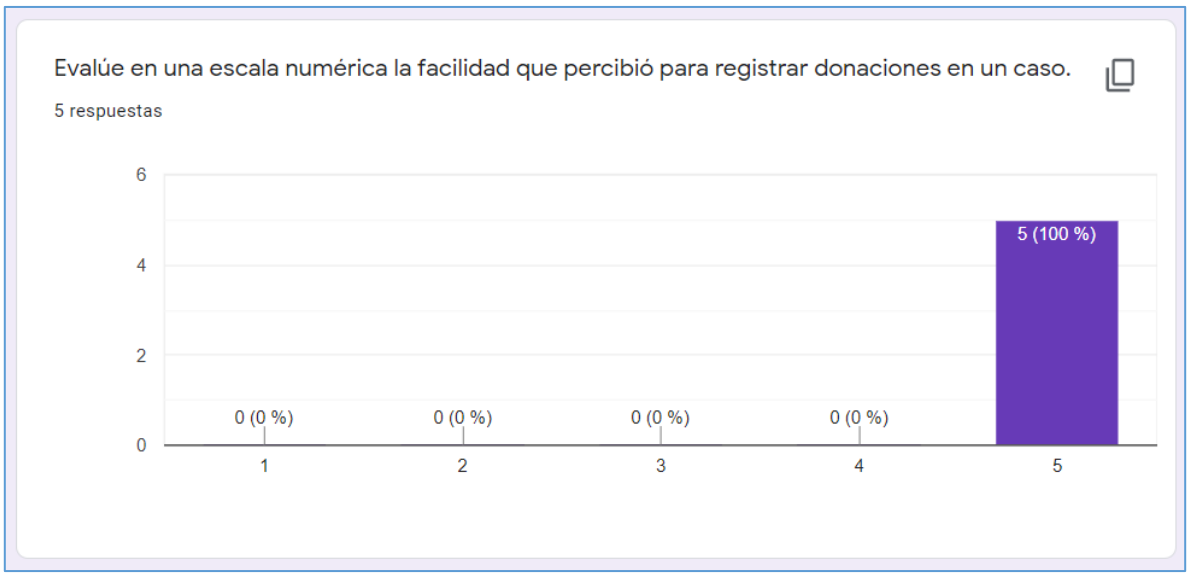

Ilustración 140 - Encuesta - Pregunta 6