UNIVERSIDAD PERUANA UNIÓN

FACULTAD DE INGENIERÍA Y ARQUITECTURA

Escuela Profesional de Ingeniería de Sistemas

Una Institución Adventista

Implementación de la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, para mejorar el control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.

Por:
Eliacer Fernandez Guevara

Heber Quelion Flores Chura

Asesor:
Mg. Joseph Ibrahim Cruz Rodríguez

Morales, noviembre 2019

DECLARACIÓN JURADA DE AUTORÍA DEL INFORME DE TESIS

Mg. Joseph Ibrahim Cruz Rodríguez, de la Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas, de la Universidad Peruana Unión.

DECLARO:

Que el presente informe de investigación titulado: ***"Implementación de la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, para mejorar el control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto"*** constituye la memoria que presenta los **Bachilleres Eliacer Fernandez Guevara y Heber Quelion Flores Chura;** para aspirar al título Profesional de Ingeniero de Sistemas, que ha sido realizada en la Universidad Peruana Unión, bajo mi dirección.

Las opiniones y declaraciones en este informe son de entera responsabilidad del autor, sin comprometer a la institución.

Y estando de acuerdo, firmo la presente declaración en Morales, a los 05 días del mes de diciembre del año 2019.

Mg. Joseph Ibrahim Cruz Rodríguez
Asesor

Implementación de la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, para mejorar el control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.

TESIS

Presentada para optar el título profesional de Ingeniero de Sistemas

JURADO CALIFICADOR

Mg. Danny Lévano Rodríguez
Presidente

Mg. Immer Elías Cuellar Rodríguez
Secretario

Ing. Cristian Werner García Estrella
vocal

Mg. Joseph Ibrahim Cruz Rodríguez
asesor

Morales, 29 de noviembre del año 2019

Dedicatoria

A mis padres Felipe y Amelida, mentores que sembraron en mi el deseo de superación, a mi hermana Kathya, por su gran dedicación al apoyarme en todo tiempo, y a mis demás hermanos que me dieron la motivación para cumplir las metas trazadas.

Eliacer Fernandez Guevara

A mis padres Isidro y Julia Isabel por brindarme todo el apoyo para construir uno de mis sueños, a mis hermanos: Edith y Jonathan por la confianza que me brindaron para salir adelante.

Heber Quelion Flores Chura

Agradecimiento

A Dios Padre, por darnos la oportunidad de seguir ampliando nuestros conocimientos y formar parte de una sociedad que ayude a sus semejantes.

A la Universidad Peruana Unión, filial Tarapoto por ser la casa de estudios que me formó, y asimismo, al área de Bienestar Universitario por brindarnos las facilidades para ejecutar la presente investigación.

Al Magister Joseph Cruz Rodríguez, por compartir su experiencia profesional en el desarrollo del presente proyecto de investigación, como asesor de tesis.

A la Magister Jessica Pérez Rivera, por su mentoría y colaboración en temas estadísticos aplicados en este proyecto.

Tabla de contenido

Índice de Tablas

Índice de Figuras

Índice de Gráficos

Índice de Anexos

Resumen

La presente investigación titulado: “Implementación de la aplicación “Control Residente”, bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, para mejorar el control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto”, posee como objetivo principal automatizar el control de servicios brindados a los residentes universitarios, y como retorno perfeccionar la eficiencia del uso de los recursos en la entidad anfitriona.

La investigación es aplicada, el diseño preexperimental, ya que se tomó un mismo grupo de estudio conformado por 4 personas encargadas del área Bienestar Universitario, específicamente, actores de las residencias universitarias, a los cuales se les aplicó el instrumento pre y postest.

En el proyecto se combinó el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming. SCRUM enfocándose en la organización y gestión del proyecto, y XP centrado en las prácticas de programación utilizadas en el desarrollo de los entregables. En este proyecto se creó un API REST en Python, un app móvil con IONIC Framework, y una aplicación web como soporte a las operaciones del cliente. Para ello, se utilizó el motor de bases de datos PostgreSQL, y otras herramientas tecnológicas.

Los resultados obtenidos, se realizaron mediante la prueba “*t – Student*” a través del paquete estadístico SPSS (22.0), el cual nos indica que existe diferencia significativa entre las variables analizadas el pre y postest aplicados al grupo de estudio. Esto significa que la aplicación “Control Residente”, mejora la eficiencia del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

Palabras claves: Metodologías ágiles, PostgreSQL, Python, Scrum, Base de Datos.

Abstract

This research entitled: "Implementation with "Residence Hall Control" application, under SCRUM framework and Extreme Programming, agile development methodology, for improving the control of services provided to resident students of Universidad Peruana Unión, site Tarapoto, San Martin – Peru, 2019", has a purpose, which is to automate the control of services provided to university residents, and as a return to improve the efficiency of the resources-use in the host entity.

This is an applied research, of pre-experimental design, since we took the same study group consisting of 4 people in charge of the University Welfare department, specifically, staff from the university residence halls, to which the pre and post tests were applied.

The project combined SCRUM framework and Extreme Programming. SCRUM focusing on the organization and management of the project, and XP focusing on the programming practices used in the development of deliverables. In this project, we created a REST API in Python, a mobile app using IONIC Framework, and a web application to support customer operations. For this, we used PostgreSQL database and some other technological tools.

The results were obtained using “t - Student” test through statistical package SPSS (22.0), which indicates that there is a significant difference between the variables analyzed, pre and post tests applied to the study group. This means the “Residence Hall Control” application improves the efficiency in the control of services provided to resident students of Universidad Peruana Unión, Site Tarapoto.

Key Words: Agile methodologies, PostgreSQL, Python, Scrum, Database.

Capítulo 1

El problema

1.1. Antecedentes de la investigación

Ruiz de la Peña & Cuba Céspedes (2010), realizaron un "Sistema de gestión de información para la Residencia Universitaria de la Universidad de Holguín - Oscar Lucero Moya", donde se abordó los procesos de matrícula y el inventario de medios básicos, con el objetivo de gestionar la información mediante el uso de una aplicación que benefició a administradores, directivos y trabajadores de las residencias universitarias. En la construcción de la solución tecnológica se usó el Sistema de Administración de Contenidos (CMS, por las siglas en inglés, Content Management System) Joomla, el lenguaje de programación PHP, motor de base de datos MySQL, y el servidor web Apache. Como metodología de desarrollo de software aplicaron RUP. El resultado fue una aplicación web Cliente – Servidor, que logró unificar la información favoreciendo la mantenibilidad, integridad, confiabilidad y accesibilidad por parte de los usuarios.

Rafael, Morales, Guzmán, & Marcial (2017), realizaron un proyecto titulado: "Sistema integral web para la gestión, control y seguimiento de residencias profesionales, servicio social y visitas a empresas", aplicado en el Instituto Tecnológico de Oaxaca, para la gestión, control y seguimiento de residencias profesionales, servicio social y visitas a empresas; con el objetivo de mejorar la eficiencia de las actividades administrativas del departamento, el control y seguimiento de expedientes y trámites de los estudiantes. Para la implementación se basaron en la metodología de desarrollo ágil Extreme Programming. Como lenguaje de programación usaron PHP, framework Slim, y MySQL como motor de base de datos.

1.2. Descripción del problema

El área de Bienestar Universitario (BU), es responsable en velar por la integridad emocional, física, mental y espiritual de toda la comunidad universitaria. Para esto, tiene que ejecutar muchas actividades distribuidas y categorizadas por tipos de usuarios. En la Universidad Peruana Unión, campus Tarapoto, cuenta con varios servicios, lo cual en este proyecto se abordará respecto a Residencias Universitarias. El campus cuenta con 2 residencias universitarias, una para varones y otro para mujeres. Atiende a un promedio de 34 varones y 52 mujeres por cada semestre, ver *Gráfico 1*. Dentro de las residencias se comparten varias tareas de vital importancia que benefician a estudiantes, apoderados, preceptores y responsables del área de Bienestar Universitario. Entre los estudios de análisis están los registros de permisos, entradas y salidas del campus; y seguimiento de los permisos.

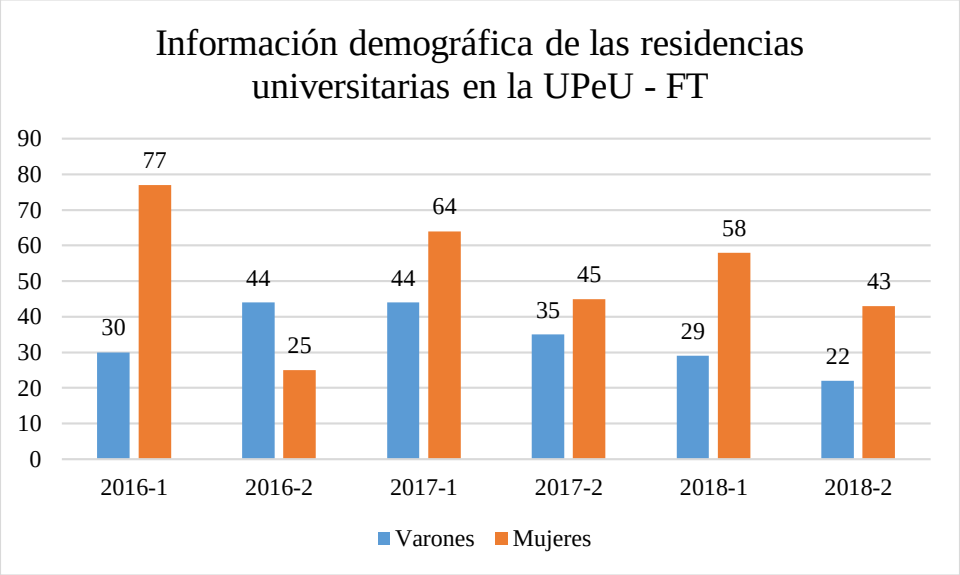


Gráfico 1. Datos históricos de residentes universitarios de los últimos seis semestres académicos.

En una entrevista realizada a la preceptora de damas Conqui (Entrevista Personal: octubre 08, 2018) mencionó los siguientes problemas:

- Deficiente gestión de los permisos asignados debido a las limitaciones de las herramientas utilizadas (procedimientos manuales) que dificultan el seguimiento adecuado por cada estudiante ocasionando y lentitud en el flujo de información.
- Dificultad para hacer reportes semanales y mensuales de permisos de los residentes, durante el ciclo académico.
- Información susceptible el deterioro.

Asimismo, en otra entrevista realizada a los preceptores encargados de las residencias, los Licenciados Perales y Fernandez (Entrevista Personal: octubre 17, 2018) expresaron los siguientes problemas:

- Dificultad para realizar un seguimiento personalizado de los residentes con respecto a sus permisos diarios. Dificultad en reconocer si este salió, ingresó, no ingresó, entre otros. Estos generan gastos en las operaciones de mano de obra y de uso de recursos por parte del personal de residencias.
- Conocer el comportamiento histórico de los permisos, de cada residente, al momento de realizar uno nuevo. Como el procedimiento es manual, dificulta la obtención en el momento determinado. Esto repercute en un excesivo uso de tiempo, además de imprecisión al momento de tomar una decisión como es el caso de otorgar o rechazar el nuevo permiso.
- Demoras en el registro de permisos, originado por diversos motivos: Ausencia del preceptor(a), masivas solicitudes para ser registrados por uno o dos preceptores (generalmente en días festivos, fines de semana, fines de semestre), entre otros.

- Frecuentes reclamos por parte de los apoderados. Los residentes sobrepasan el límite del permiso. En dichos casos se verificaba varias horas después de la hora de retorno del residente; y solo se observaba por un reclamo del apoderado o después de ocurrir algún tipo de incidencia.
- Inconsistencia de información en los reportes realizados de manera periódica de los permisos. En muchos de los casos, los residentes salían sin la aprobación previa de sus padres y preceptores, o terminaban haciendo caso omiso a las normas establecidas del Compromiso de Honor del residente universitario.

1.3. Formulación del problema

¿Es eficiente la aplicación "¿Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto?

¿Es eficiente la aplicación "¿Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos largos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto?

¿Es eficiente la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos cortos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto?

¿Es eficiente la aplicación "¿Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto?

1.4. Objetivo general

Determinar la eficiencia de la aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

1.5. Objetivo específico

- Determinar la eficiencia de la aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos cortos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.
- Determinar la eficiencia de la aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos largos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.
- Automatizar el control de servicios aplicando las guías del marco de trabajo SCRUM en la gestión del proyecto y Extreme Programming, en la construcción de la solución tecnológica.
- Determinar la eficiencia de la aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

1.6. Justificación

La solución desarrollada para resolver la problemática existente consistió en implementar un sistema de información multiplataforma: Entorno web y aplicación móvil, facilitando el control de las operaciones por parte de los beneficiarios descritos. Además, se muestran a continuación los beneficios detalladamente:

En ciencia y tecnología: aportará desde la recolección de los requisitos, aplicación de la metodología y demás herramientas, y la construcción de la solución tecnológica. Fortalecerá los conceptos de Ingeniería de Software y sus aplicaciones en problemas del mundo real, facilitando la elección de herramientas en los distintos escenarios que se puedan presentar.

En metodología: Sirve como referencia la combinación del marco de trabajo Scrum y la metodología ágil Extreme Programming, metodologías ágiles que guían el éxito de proyectos.

En el centro de aplicación: Se usa como una herramienta capaz de procesar las tareas que se realizan en el control de servicios, reduciendo significativamente los recursos utilizados, y favoreciendo la toma de decisiones de los responsables directos.

Los usuarios finales del sistema de información son: Preceptores, personales de seguridad, residentes y apoderados. Ayuda a controlar de manera holística la gestión de permisos mediante el monitoreo en cada eslabón o estado del permiso como: solicitud, salida, ingreso y tardanza, para que los actores (preceptores y personal de seguridad) puedan intervenir apropiadamente y en el momento oportuno. Con la implementación de una aplicación web y móvil se gestionó todas las actividades descritas, favoreciendo la elaboración de informes periódicos.

Finalmente, entre la comunidad beneficiaria están los directivos de Bienestar Universitario, área principal encargada de administrar los servicios que se brinda; quienes podrán registrar, actualizar y consultar la información en cualquier intervalo de tiempo, haciendo que esta fluya con facilidad, entre los responsables, para optimizar la toma de decisiones.

1.7. Presuposición filosófica

Creyentes en la existencia de un Dios de orden, teniendo como guía los principios establecidos en su Palabra; se reconoce que todo conocimiento proviene de Él. Se asume que la vida está dotada de dones y talentos para poder interactuar con nuestros semejantes; cuyas virtudes se despliegan en el espacio y tiempo, por medio del desarrollo y la aplicación de los conocimientos. Se considera al "tiempo" como el recurso más preciado concedido al hombre, y la forma de aprovechar este recurso ha permitido crear este software. Asimismo se reconoce al Creador, al momento de hacer el universo, puso cada cosa en su debido lugar magnificando el tiempo y el orden. Bajo este modelo nos enmarcamos buscando mejorar el orden de las actividades que desarrollan las organizaciones, optimizando los recursos valiosos que poseen.

Capítulo 2

Revisión de la literatura

2.1. Introducción

En este capítulo se abordará sobre los conceptos teóricos de la investigación; considerando la metodología de desarrollo ágil XP y el marco de trabajo Scrum utilizada para la implementación del proyecto. Por otro lado, es importante mencionar conceptos generales del área de Bienestar Universitario, sistemas de información, lenguaje de programación, base de datos y otras herramientas tecnológicas usadas en la industria del software.

2.2. Bienestar Universitario

Según la Ley Universitaria establecida por la Superintendencia Nacional de Educación Superior Universitaria (SUNEDU 2014), especifican que el área de Bienestar universitario tiene como función planificar, organizar, dirigir y ejecutar servicios asistenciales, de salud, alimentación, recreación, educación y otras actividades. Todo esto ha de realizarse en medida de las posibilidades y cuando haya casos que lo ameriten. En cuanto a los servicios asistenciales abarcan la formulación de becas parciales o totales, tutorías académicas, programas de ayuda económica en tareas formativas como materiales de estudio e investigación y otros. En Salud se especifica el seguro universitario, orientación psicológica, y otros. En alimentación, lo relacionado con becas en el comedor universitario. En actividades recreativas, está la promoción del deporte, actividades culturales y artísticas. En educación, actividades de formación profesional como investigación, adquisiciones de material bibliográfico, y demás relacionados con la misma.

Esta es la normativa que rige a todas las instituciones universitarias de nuestro país, haciendo que estas estén preocupadas en la formación integral de la comunidad universitaria.

2.3. Gestión de la información

Arévalo (2007), explica que el propósito de la gestión de la información es ofrecer elementos que ayuden a la organización para obtener, producir y transferir, al menor costo posible, datos e información actualizada, exacta y de calidad, que repercuta positivamente en los objetivos de la organización. Dicho de otro modo, es recolectar la información adecuada, para la persona que lo necesita, en el momento adecuado, con el menor costo posible y que a una mejor toma de decisiones.

Gestionar la información es un reto para todas las organizaciones, ya que se necesita de los recursos necesarios para hacer todo este proceso. Es bien sabido que, existe un flujo de información bien definido: Entrada, almacenamiento, procesamiento y salida de la información. Esto exige utilizar recursos para recolectar de manera ágil, segura y viable los datos; seguidamente, ser almacenados en repositorios, archivos, entre otros; luego procesar los datos y obtener información; para finalmente ser difundida en el momento adecuado para proceder a la respectiva toma de decisiones. Los procedimientos manuales no resultan muy bien especialmente en los procesos críticos de un área específica, tornándose tediosa e incómoda el debido tratamiento, además de no utilizar los recursos que se invierten adecuadamente.

Con la aparición de las Tecnologías de la Información (TI), se ha revolucionado la capacidad respecto al tratamiento, el uso de recursos, disponibilidad, y confiabilidad de la información. Esto ha enriquecido la forma en que responsables directos y altos directivos toman sus decisiones, y por ende el éxito de las organizaciones.

2.4. Sistemas de información

Fuster, Hormigo, Joana, & Rodríguez (2011), definen un sistema de información como "un conjunto de elementos interrelacionados que permiten transformar los datos en información y conocimiento, poniendo todo ello a disposición de los empleados y directivos de la organización para actuar en consecuencia".

Los sistemas de información (SI) están cambiando y mejorando la forma en que las organizaciones operan, con su uso, ayudan a optimizar los procesos operacionales que proveen información valiosa a sus directivos ayudando al proceso de toma de decisiones, logrando ventajas competitivas. Se dice que la información es uno de los activos más valiosos de la organización, y estos, son producto del procesamiento de todos los datos realizados por los sistemas de información. Bajo este contexto y desde el punto de vista de sus funciones en ella, Chatterjee (2017) define un sistema de información como un sistema bien coordinado de recopilación, procesamiento, almacenamiento y recuperación de información.

Chatterjee (2017), especifica que cualquier sistema de información tiene tres componentes principales: unidad de entrada, procesador y unidad de salida. La forma en que se recupera la información se muestra a continuación con la ayuda de un diagrama (ver Figura 1).

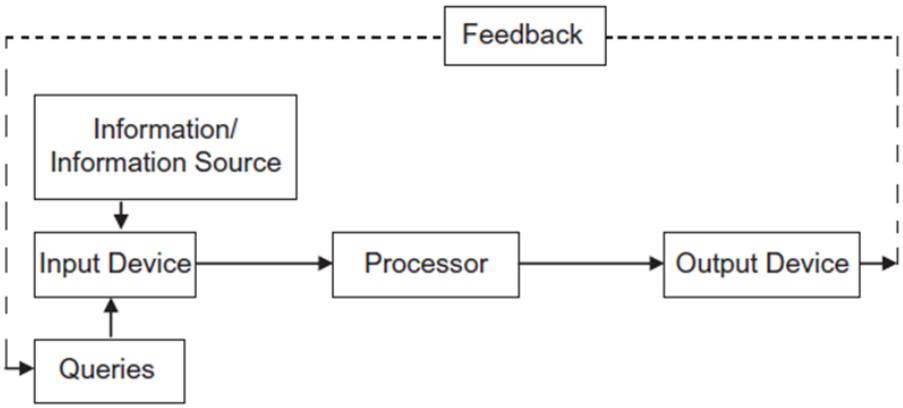


Figura 1. Diagrama de los componentes de un Sistema de Información (Chatterjee, 2017).

2.4.1. Tipos de sistemas de información

Los sistemas de información se clasifican de acuerdo a los niveles de organización de la empresa, siendo así que se deban desarrollar diferentes tipos de sistemas de información: sistemas para el procesamiento de transacciones, sistemas de información administrativa y sistemas de apoyo a la decisiones (Arjonilla & Medina, 2007).

2.4.1.1. Sistema de Procesamiento de Transacciones (TPS)

Estos sistemas de información son los que mantienen en marcha las operaciones de las empresas, ya que sustituyen los procedimientos manuales a tareas de ordenadores, haciendo que estas sean más eficientes, ordenadas y bien estructuradas. Recogen información de actividades rutinarias que son de vital importancia tales como operaciones de ventas, compras, pagos, pedidos, entre otros (Lapiedra, Devece, & Guiral, 2011).

2.4.1.2. Sistema de información administrativa

Son sistemas de información que tienen como objetivo principal brindar a directivos información necesaria (informes periódicos, resúmenes mensuales, anuales, entre otros) para que estos puedan tomar decisiones estructuradas y resolver problemas a tiempo. (Lapiedra et al., 2011)

2.4.1.3. Sistemas de apoyo a la decisión (DSS)

Estos tipos de sistemas tienen como objetivo ayudar al decisor durante el proceso de toma de decisiones, puesto que se requiere del poder de procesamiento de datos de los ordenadores. Estos sistemas no son estructurados, sino que sirven de apoyo y finalmente el criterio del decisor es lo que le lleva a tomar una decisión (Lapiedra et al., 2011). Por otro lado (Kyocera, 2017), explica que “Son un tipo de sistema computarizado de información organizacional que ayuda al gerente en la toma de decisiones cuando necesita modelar, formular, calcular, comparar, seleccionar la mejor opción o predecir los escenarios”.

2.4.1.4. Sistemas de información para ejecutivos (EIS)

Los sistemas de información ejecutiva (EIS por sus siglas en inglés) proporcionan de manera abstracta y simplificada la información interna y externa, presentada a menudo en formato gráfico. Son usados por ejecutivos y gerentes para tomar decisiones a nivel estratégico (Kyocera, 2016).

2.5. Metodologías Ágiles

Las metodologías ágiles surgen ante la necesidad de ver una nueva forma de desarrollar software, puesto que se encontraban muchas limitantes con las metodologías tradicionales, como es el caso del entorno cambiante de la industria y otros. Es así que por los años 90's empezaron a aparecer denominadas como metodologías livianas. Varios años después se oficializaron como metodologías ágiles, encontrándose Extreme Programming (XP), Scrum, Software Craftmanship, Lean Software Development, entre otros.

Palacia (2015), sostiene que "La gestión de proyectos ágiles no se formula sobre la necesidad de anticipación, sino sobre la de adaptación continua". Justamente esta es una razón fundamental por las que vio la necesidad de crear una nueva forma de gestionar proyectos.

En febrero de 2001, se marcó un momento sumamente importante en el desarrollo ágil de software, donde diecisiete profesionales reconocidos del desarrollo de software crearon The Agile Alliance, una organización sin fines de lucro, enfocada a la promoción de principios y valores ágiles ayudando a la adopción por parte de las organizaciones. Es así como se consolidó el Manifiesto ágil, como el cimiento ágil (Alaimo, 2013).

2.5.1. Manifiesto Ágil

Kent Beck, et al. (2001), especifican en El Manifiesto Ágil cuatro valores:

- **Los individuos y su interacción**, sobre procesos y herramientas.
- **El software que funciona,** sobre la documentación excesiva.

- **La colaboración con el cliente,** frente a la negociación contractual.
- **La respuesta al cambio,** sobre el seguimiento de un plan.

A continuación, detallaremos en específico sobre cada uno de estos cuatro valores.

1. **Los individuos y su interacción**, sobre procesos y herramientas.

   El factor de éxito en un proyecto de software son las personas que intervienen. Esto va a depender de acuerdo a la forma de seguir el proceso, en caso se fallar, las garantías del éxito son mínimas. Para esto es necesario que los integrantes construyan su entorno de desarrollo basándose en sus necesidades favoreciendo la adaptación del trabajo en equipo.

2. **El software que funciona,** sobre la documentación excesiva.

   Existe una regla fundamental de "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante" (P. Letelier & Penadés, 2006). Estos deben ser cortos y centrarse en lo más importante (diseño, manual del sistema y/o de usuario).

3. **La colaboración con el cliente,** frente a la negociación contractual.

   Aquí se tiene al cliente en constante interacción con el equipo de desarrollo, garantizando la buena marcha y éxito del proyecto.

4. **La respuesta al cambio,** sobre el seguimiento de un plan. La capacidad de adaptarse a los constantes cambios repercuten en el éxito del proyecto. A esto ayuda una buena planificación con duraciones cortas que sean flexibles y abiertas, capaces de adaptarse a los cambios que puedan surgir.

Los valores son las bases que cimientan los doce principios del Manifiesto ágil. De entre los doce, los dos primeros, resumen el espíritu ágil del desarrollo de software, mientras que los siguientes están orientados al proceso o al equipo de desarrollo. (Beck et al., 2001), listan los siguientes:

1. La prioridad es satisfacer al cliente a través de entregables funcionales en periodos cortos.
2. Aceptar los cambios en cualquier etapa del desarrollo. Estos, harán a nuestro cliente más competitivo.
3. Realizar entregas funcionales de software en intervalos de tiempos cortos, desde un par de semanas a un par de meses.
4. La intervención del cliente y los desarrolladores debe ser diariamente durante todo el proyecto.
5. La construcción del proyecto debe estar ligada a individuos motivados. Para ello, es necesario brindarles todo el apoyo que necesiten y confiar en ellos.
6. La comunicación es más efectiva si se mantiene al equipo de desarrollo en contacto o en el mismo ambiente de trabajo.
7. El indicador de progreso es un software que funciona.
8. Los procesos ágiles ayudan a tener un desarrollo sostenido. El cliente y desarrolladores deben lograr mantener un ritmo de trabajo constante.
9. La preocupación por la calidad técnica y el buen diseño favorece la agilidad.
10. Se debe realizar solo el trabajo que sea necesario.
11. Un equipo bien organizado define adecuadamente la estructura del proyecto y las herramientas a utilizar.

12. Es recomendable que el equipo se autoevalue con el fin de reconocer sus fortalezas y fortalecer los aspectos que debe mejorar, mediante técnicas que le permitan ser más efectivos.

2.5.2. Extreme Programming. (Programación Extrema - XP)

Letelier & Penadés (2006), mencionan que XP es una metodología ágil enfocada en mejorar las relaciones interpersonales como la clave del éxito en el desarrollo de software, promover el trabajo en equipo, cuidar el aprendizaje de los desarrolladores y fomentar un buen clima de trabajo.

2.5.3. Roles XP

Esta metodología de desarrollo de software tiene roles específicos dentro de cada proyecto. Según Letelier & Penadés (2002), mencionan en su artículo "Metodologías ágiles para el desarrollo de software: Extreme Programming (XP)" los siguientes roles:

2.5.3.1. Programador

El programador es considerado el miembro más importante del equipo, ya que su responsabilidad es escribir las pruebas unitarias y producir el código para el sistema. Debe haber una comunicación y coordinación apropiada entre los programadores y demás integrantes del equipo.

2.5.3.2. Cliente

El rol del cliente es importante como el rol del programador, ya que él es responsable de escribir las historias de usuario como también las pruebas funcionales para luego aceptar su implementación. Además, especifica las prioridades de cada historia de usuario y decide cuales se realizan en cada iteración enfocándose en agregar más valor al negocio. Debe haber una comunicación constante entre el cliente y el equipo.

2.5.3.3. Encargado de pruebas (Tester)

El encargado de pruebas es responsable de ayudar al cliente a seleccionar y escribir las pruebas funcionales para cada historia de usuario. Realiza las pruebas funcionales regularmente, expone los resultados obtenidos en el equipo, esto permite tener muy clara la visión del progreso del proyecto.

2.5.3.4. Encargado de seguimiento (Tracker)

Encargado de verificar las estimaciones realizadas y evaluar el proceso de cada iteración, comunicando los resultados para mejorar las estimaciones futuras. Mantiene contacto directo con el equipo de desarrollo y determina cuando es necesario realizar un cambio para lograr los objetivos de cada iteración.

2.5.3.5. Entrenador (Coach)

El rol de esta persona es múltiple, ya que tiene diferentes sub-roles que debe asumir, se le considera el "arquitecto principal" del proyecto porque su trabajo es asesorar / proporcionar guías a los miembros del equipo, ver una gran visión del proyecto y decidir la metodología que se aplicará en el proyecto.

2.5.3.6. Consultor

El consultor es un miembro externo del equipo con conocimientos específicos sobre el tema que es necesario para el proyecto, como también guía al equipo para resolver un problema determinado.

2.5.3.7. Gestor (Big boss)

Es el enlace que une al cliente y a los programadores, también ayuda al equipo a trabajar de manera eficaz para establecer las condiciones adecuadas. Su labor esencial es la coordinación.

2.5.4. Fases de XP

Según Letelier & Penadés (2002), mencionan que el ciclo de vida ideal de la Programación Extrema consta de seis fases, las cuales son:

2.5.4.1. Exploración

Fase en donde los clientes describen las historias de usuario que son de interés para la entrega inicial del producto. Al mismo tiempo, el equipo de desarrollo se adapta a las herramientas, tecnológicas y prácticas que se utilizarán en el proyecto. La tecnología se prueba y las posibilidades de la arquitectura del sistema se exploran mediante la construcción de un prototipo. La fase de exploración lleva de unas pocas semanas a unos pocos meses, dependiendo del tamaño y la familiaridad que los programadores tienen con la tecnología.

2.5.4.2. Planificación de la entrega

Durante esta fase, el cliente y el equipo de trabajo definen el nivel de prioridad de las historias de usuario más importantes, menos importante, también en esta fase se estima el esfuerzo necesario que se va a efectuar en cada entrega.

2.5.4.3. Iteraciones

En esta fase se definen las iteraciones que se realizarán en la construcción del producto, el tiempo de cada iteración es no mayor a tres semanas, teniendo en cuenta que la primera iteración es la base para las demás iteraciones.

2.5.4.4. Producción

En esta fase, se requiere realizar pruebas y revisiones de rendimiento de cada iteración antes de ser entregadas al cliente, además se deben tomar decisiones sobre la inclusión de las nuevas características del cliente a la versión actual.

2.5.4.5. Mantenimiento

Después de implementar la primera versión del sistema, XP debe mantener el sistema funcionando en paralelo a él, se desarrollan las otras iteraciones. Es importante mencionar que su principal objetivo es satisfacer al cliente.

2.5.4.6. Muerte del proyecto

Es la fase en la que el cliente no posee más historias de usuario para incluir en el sistema, es donde se da por terminado el sistema.

2.5.5. Valores de XP

Según Joskowicz (2008), menciona que la metodología XP se basa esencialmente en cuatro valores, que deben estar presentes en el equipo de desarrollo para que el proyecto tenga mayor éxito.

1. **Comunicación**

La comunicación es la clave del éxito, considerándose un elemento de vital importancia en XP. Puesto que se considera poca documentación, es sumamente necesario el diálogo frontal entre miembros del equipo, cliente, dueño del producto y los demás roles.

2. **Simplicidad**

La simplicidad o sencillez se basa en aplicar esto al código, proceso, diseño, etc. Favorece la respuesta ágil a posibles recodificaciones futura, calidad del producto, legible y entendible.

3. **Retroalimentación**

Se basa en recibir comentarios, aportaciones, críticas u observaciones tanto de los miembros del equipo como de parte del cliente. Esto ayuda a mejorar las relaciones y la calidad de los entregables en las siguientes iteraciones.

4. **Coraje**

Este valor fomenta la valentía y persistencia para resolver conflictos como: rediseñar, refactorizar, deshacer componentes de código sin tener en cuenta el tiempo invertido; de tal manera que favorezca a las modificaciones futuras.

2.5.6. Prácticas de XP

Según Beck (2000) menciona que la metodología XP tiene 12 prácticas que guían el desarrollo del proyecto, las cuales se evidencian en la Figura 2:

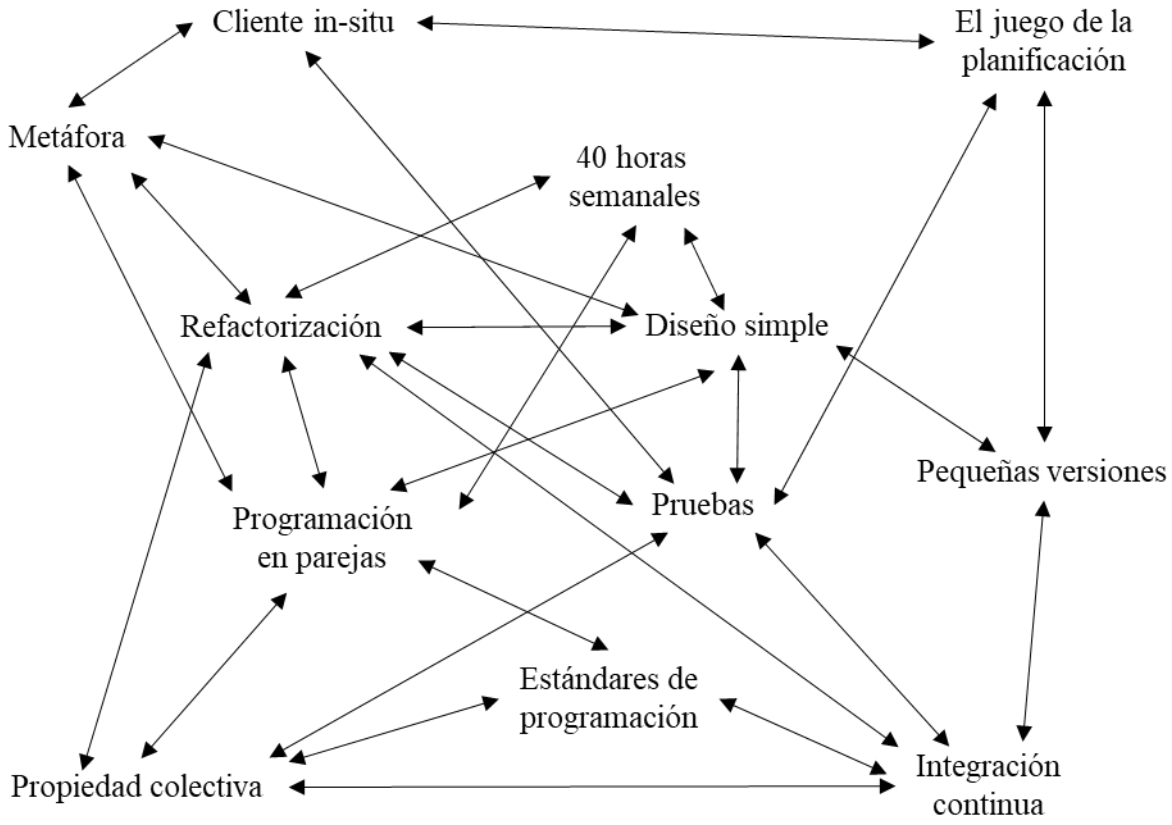


Figura 2. Buenas prácticas de XP (Beck, 2000).

1. **El juego de la planificación.** Se planifican las entregas en base a la estimación de las historias de usuario dadas por el equipo y el ratio de importancia asignados por el cliente.

2. **Pequeñas versiones.** Las versiones deben ser pequeñas, capaces de ser finalizadas en pocas semanas. Estas deben ser versiones funcionales para el usuario final.
3. **Metáfora.** Es una descripción simple que proporciona la información suficiente que sirve de guía capaz de ser entendida por el cliente y el equipo de desarrollo, sobre cómo funcionará el futuro sistema.
4. **Diseño simple.** Esto implica desarrollar solo lo necesario, no diseños de futuros no definidos. Debe evitar lógicas duplicadas, funciones no usadas y superar todos los testeos.
5. **Pruebas.** Son aquellas actividades que comprueban las funcionalidades del sistema. Con esto el encargado de pruebas verifica las pruebas de aceptación definidas por el cliente para cada historia de usuario.
6. **Refactorización.** Actividad constante que consiste en escribir nuevamente el código sin cambiar la funcionalidad del sistema, el objetivo es eliminar duplicidad de código, optimizar su legibilidad y hacerlo más simple.
7. **Programación en parejas.** Consiste en trabajar en parejas en un mismo ordenador. Uno de ellos es el que escribe los códigos, mientras que el otro el que piensa con más detenimiento. Cockburn & Williams (2001), mencionan que con la programación en pares se minimizan los errores, mejora la comunicación del equipo y se logran mejores diseños.
8. **Propiedad colectiva.** Cualquier programador puede realizar cambios en el código para corregir problemas, añadir funcionalidades o recodificar. Cada miembro del equipo debe poder visualizar los avances de los demás en cualquier momento.

9. **Integración continúa.** Consiste en unificar las diferentes partes del sistema, en cuanto se tenga una nueva funcionalidad compilada y probada. Esto favorece a los desarrolladores a poder trabajar siempre en con la "última versión". Se recomienda poder publicarlos diariamente.
10. **40 horas semanales.** Kendall & Kendall (2005), sostienen que el equipo de desarrollo debe trabajar un máximo de 40 horas por semana, esta práctica fundamental de XP tiene como propósito motivar y conservar un ritmo constante y razonable, sin sobrecargar al equipo de desarrollo.
11. **Cliente in-situ.** Implica tener al cliente disponible en cada momento que el equipo estime hacer preguntas, para de esta manera disolver dudas, resolver discusiones y fijar las prioridades.
12. **Estándares de programación.** Todo el código debe de tener un estándar de codificación implantado por el equipo para que el código sea más legible y fácil de mantener por todos los integrantes.

2.5.7. Herramientas de XP

2.5.7.1. Historias de usuario

Letelier & Penadés (2002), sostienen que las historias de usuario representan una breve descripción de las características que debe tener el sistema, ya sea funcional o no funcional.

2.5.7.2. Tareas de ingeniería

Priolo (2009), menciona que las tarjetas de ingeniería ayudan a simplificar la programación de una historia de usuario.

2.5.7.3. Tarjetas CRC (clase-responsabilidades-colaboradores)

Es una herramienta usada como metodología para el diseño de software, detallan las clases utilizadas en la programación de una historia. Chiluisa & Loarte (2014), mencionan que una tarjeta CRC permite conocer que clases compone el proyecto y cuales interactúan entre sí.

2.5.8. Ventajas y desventajas de XP

La ventaja de XP es la adaptabilidad al desarrollo de diferentes tipos de sistemas, además de optimizar el tiempo de desarrollo y también permite realizar el desarrollo en pares.

La desventaja es el tiempo para el desarrollo del sistema, la definición del costo y la disponibilidad del cliente para las entregas.

2.6. Marco de trabajo Scrum

Scrum es un marco de trabajo(framework) ágil que reduce la complejidad en el desarrollo de productos para satisfacer las necesidades de los clientes a través de un entorno de transparencia en la comunicación, responsabilidad y progreso continuo así como menciona Pries & Quigley (2011), en su libro titulado "Gestión de proyectos Scrum" ayuda al establecer un conjunto específico de actividades dentro de un lapso de tiempo relativamente corto.

2.6.1. Roles de Scrum

El equipo Scrum está formado por los siguientes roles:

2.6.1.1. Product Owner/Dueño del producto

El dueño del producto es la única persona responsable de desarrollar, mantener y priorizar las tareas en el backlog y de garantizar el valor del trabajo que se realiza el equipo, si en algún momento se desea modificar la prioridad de un elemento es necesario convencer al dueño del producto (Schwaber & Sutherland, 2010).

2.6.1.2. Scrum Master

Las funciones de Scrum Master es gestionar que el proceso Scrum se lleve a cabo y ayudar a eliminar progresiva y constantemente dificultades que van surgiendo en la organización. Según Bahit (2012), el Scrum Master tiene la responsabilidad de solucionar los obstáculos que impiden el progreso del proyecto, también el Scrum Master es el encargado de incentivar y motivar al equipo.

Según Schwaber & Sutherland (2010), menciona que el Scrum Master puede ser un miembro del equipo; por ejemplo, un desarrollador que realiza tareas de Sprint. Sin embargo, esto a menudo conduce a conflictos cuando el Scrum Master tiene que elegir entre eliminar obstáculos o realizar tareas. El Scrum Master nunca debe ser el Dueño del Producto.

2.6.1.3. El Equipo Scrum

Grupo de profesionales con conocimientos técnicos necesarios que se encargan de desarrollar cada Spring planificado para lograr entregar el producto a tiempo (Schwaber & Sutherland, 2010).

Según Mountain Goat Software (1998), da a conocer que en un equipo Scrum todos trabajan juntos para cumplir el conjunto de requerimientos que se han comprometido a completar dentro de un sprint.

2.6.2. Artefactos de Scrum

Bahit (2012), explica que en el marco de trabajo Scrum existen tres herramientas o artefactos los cuales ayudan a mantener organizados los proyectos, también ayudan a planificar y revisar cada sprint. Los artefactos en Scrum son: pila del producto, pila del Sprint y el Incremento.

1. **Pila del Producto (Product Backlog)**

También llamada pila de producto, es una lista priorizada de requerimientos que se pretende hacer durante el desarrollo del proyecto, solo el dueño del producto es el encargado de incluir, modificar y eliminar algún elemento que se requiera modificar (Schwaber & Sutherland, 2010).

2. **Pila del Sprint (Sprint Backlog)**

Según Mountain Goat Software (1998), refiere que las listas de tareas identificadas por el equipo se deben completar para entregar la funcionalidad que se comprometió a entregar durante el sprint, aunque las tareas giran en torno al conjunto de elementos de Product Backlog.

3. **Incremento**

Menzinsky, López, & Palacio (2019), mencionan que el incremento es la suma de todos los elementos de la pila del producto que se desarrolla en un sprint, con la condición de ser desplegada para el propietario del producto.

2.6.3. Etapas básicas de Scrum

La clave en Scrum es conocer cuáles son sus etapas básicas. Antes de mencionar las 5 etapas básicas definiremos que es un Sprint, ya que este es el corazón de Scrum como lo menciona Schwaber & Sutherland (2010) .

1. **Sprint**

El Sprint es una iteración de un ciclo de desarrollo continuo, es decir, su duración no cambia y nos permite reducir complejidad y comparar resultados entre diferentes Sprints, un Sprint tiene una duración de 1 mes o menos.

2. **Planificación del Sprint**

Durante esta actividad se realizan reuniones de planificación que puede durar hasta 8 horas para Sprints de un mes, para Sprints más cortos la reunión es proporcionalmente más corta.

3. **Ejecución del Sprint**

Esta actividad se realiza después de la planificación del Sprint, cada iteración tiene que proporcionar un resultado completo, un incremento del producto que sea potencialmente intragable.

4. **Reuniones diarias**

Se realizan reuniones diarias de planificación de 15 minutos con el equipo Scrum. Durante esta reunión se analizan cuáles son los elementos en lo que se está y que obstáculos encuentran.

5. **Revisión del Sprint**

La revisión del Sprint se realiza al final de cada Sprint, en donde el dueño del producto presenta a los Stakeholders el incremento terminado para su inspección y adaptación, se realiza esta actividad junto con la retrospectiva para lograr un entregable al cliente final

6. **Retrospectiva del Sprint**

El propósito de la retrospectiva de Sprint es hacer una revisión sobre el último Sprint e identificar posibles mejoras y crear un plan para la implementación.

2.6.4. Principios de Scrum

Como marco de trabajo de desarrollo ágil tiene como base seis principios que orientan la gestión de un proyecto a lo largo de todas sus fases; sin embargo, estos principios de Scrum no son negociables, es decir todos deben ser cumplidos y debe aplicarse como se describe en (Guía SBOK™, 2017).

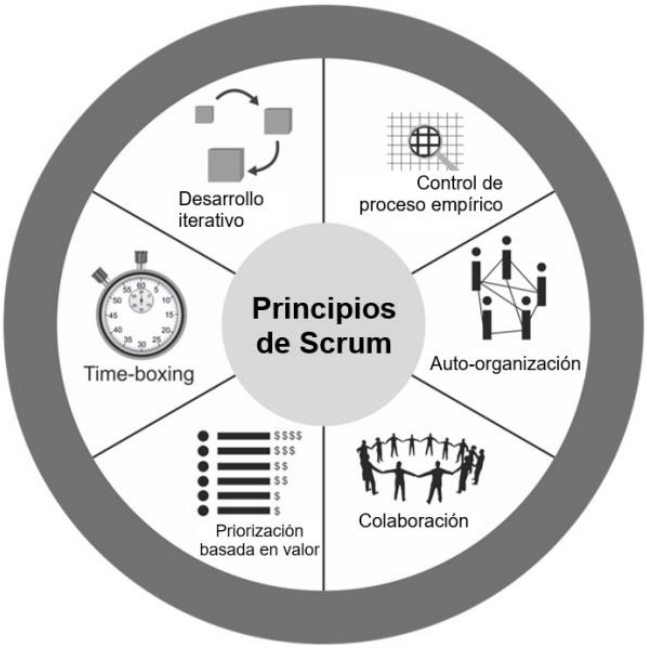


Figura 3. Principios de Scrum (Guía SBOK™, 2017).

1. **Control de proceso empírico**

Para Scrum más importante que seguir un plan es adaptarse a los cambios y mejorar de forma continua. Tres son los pilares que sustentan el marco de trabajo Scrum: transparencia, inspección y adaptabilidad.

2. **Auto-organización**

En un proyecto Scrum más importante que el mando y el control es ofrecer un ambiente con todo lo necesario para auto organizarse y trabajar de forma autónoma.

3. **Colaboración**

Se centra en las tres dimensiones fundamentales: conciencia, articulación y la apropiación. También se busca que el equipo trabaje en un solo lugar con el fin de facilitar la comunicación, la resolución de problemas y el aprendizaje.

4. **Priorización basada en el valor**

Este principio destaca en crear un producto que entregue el máximo valor comercial al negocio del cliente, de manera que, todo lo que es prioridad para el negocio del cliente, será de mayor importancia desde el principio del proyecto y continuando en todo momento.

5. **Time boxing**

En Scrum todas las actividades tienen un límite de tiempo sumamente importante, esto es con el fin de evitar que se emplee demasiado tiempo en una actividad o lo contrario.

6. **Desarrollo iterativo**

Este principio se enfoca en entregar a los interesados el máximo valor en un tiempo mínimo.

2.6.5. Valores de Scrum

Los valores de Scrum son los fundamentos, los cimientos, sobre los que se apoyan los principios y prácticas recogidos en este marco de trabajo.

Alaimo (2013), define en su libro "Proyectos Ágiles con Scrum: Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos" que Scrum se construye sobre cinco pilares, sus valores:

1. **Foco**

El equipo Scrum debe enfocarse en lo que es más importante, sin preocuparse por el futuro que puede ser incierto y cambiante. Esto permite que al final de cada Sprint se entregue un producto de alta calidad, en última instancia, permite cumplir los Sprint Goals.

2. **Coraje**

El equipo Scrum debe asumir retos, ser valientes ante los problemas y afrontarlos. Esto les permitirá crecer como profesionales y como equipo.

3. **Apertura**

El equipo Scrum debe tener una actitud abierta y proactiva para mejorar sus capacidades y competencias profesionales, también deben facilitar la transparencia en la comunicación y en la información para facilitar la colaboración dentro y fuera del equipo.

4. **Compromiso**

Cada uno de los miembros de equipo Scrum hará el máximo esfuerzo posible para lograr el éxito del equipo.

5. **Respeto**

El equipo Scrum respeta el conocimiento, las habilidades y la experiencia profesional no solo del resto del equipo, sino también con las personas que se relacionan. También comparten los éxitos y fracasos como equipo.

2.6.6. Ventajas y desventajas de Scrum

Las ventajas de Scrum son:

- El producto se lanza con mayor rapidez para que el cliente pueda utilizarlo.
- El cliente decide los nuevos objetivos a realizar.

Las desventajas de Scrum son:

- Tendencia a dejar una tarea sin terminar y empezar otra nueva por la exigencia del dueño del producto.
- Funciona bien solo en equipos pequeños.

2.7. Convergencia metodológica

En esta investigación se aplicará Scrum a nivel macro enfocándose en la organización y gestión del proyecto, y XP centrado en las prácticas de programación (entregables). Como está descrito en páginas anteriores, Scrum consta de cinco fases: iniciación, planificación y estimación, implementación, revisión, y lanzamiento. En cada una de estas fases se implementará los procesos de Scrum, y combinando las prácticas y estrategias de XP.

En la fase de iniciación se realizará los siguientes procesos de Scrum: Levantamiento de información, Identificar los interesados (Stakeholders), formación del equipo y definición de roles (Scrum Master, Team y Product Owner), y la creación de la pila del producto (Product Backlog) mediante una lista de historias de usuario (User Stories).

En la fase de planificación y estimación se realizará los siguientes procesos: definir las tareas de las historias de usuario, priorizarlas y estimarlas (Create and Estimate Tasks) considerando el factor dedicación del equipo, y elaboración de la lista de historias a incluir en el Sprint (Sprint backlog). Aquí se utilizará dos prácticas de XP: Metáfora y el Juego de la Planificación.

En la fase de implementación, se realizará la creación del entregable del Sprint, reuniones diarias de Scrum y el mantenimiento de la lista de pendientes del producto. En el desarrollo del entregable se utilizará las prácticas de XP: entregas pequeñas, diseño simple, programación en parejas, propiedad colectiva del código, cliente presente, y estándares de programación. En las reuniones diarias se evalúa el avance del Sprint, las actividades que se están desarrollando y el estado de estas, las dificultades encontradas, entre otros.

En la fase de revisión y retrospectiva, se hará la revisión del Sprint, es decir, la presentación de la Demo del Sprint; en donde se asegurará de haber cumplido con las historias de usuario seleccionadas en la etapa de planificación. Además, se efectuará las retrospectivas de Sprint, donde los miembros del equipo podrán contribuir y discutir las ideas para resaltar tres aspectos importantes: Actividades bien hechas, mejorables, y planes de mejoras. Es aquí donde se extraen las lecciones aprendidas durante el Sprint.

En la fase del lanzamiento se hará el despliegue de los entregables del Sprint aceptados por el cliente. Es aquí donde se hace el despliegue del entregable listo para ser usado por los usuarios finales. En la siguiente tabla se resume la convergencia metodológica de Scrum y XP.

Tabla 1

Convergencia Metodológica Scrum y XP – Elaboración propia.

Fases	**Scrum**	**XP**
Iniciación	Visión del Proyecto Identificar Interesados Formación del equipo y definición de roles (Scrum Master, Development Team y Product Owner) Creación de la pila del producto Crear tareas por historias de usuario Estimar historias de usuario	
Planificación y Estimación	Estimar Sprint Crear la lista de historias a incluir en el Sprint (Sprint Backlog) Definir lugar del Scrum diario	Buenas prácticas de XP: Metáfora, Juego de Planificación
Implementación	Desarrollar el entregable del Sprint	Buenas prácticas de XP: entregas pequeñas, diseño simple,

	Scrum diario	programación en parejas, propiedad colectiva del código, cliente presente y estándares de programación
	Actualización de los pendientes del Sprint	
Revisión	Presentación de la Demo del Sprint	
	Retrospectiva del Sprint	Plus and Delta, Regla de Pareto, y 5 ¿Por qué?
Lanzamiento	Despliegue de entregables	Integración definida por el equipo de desarrollo.

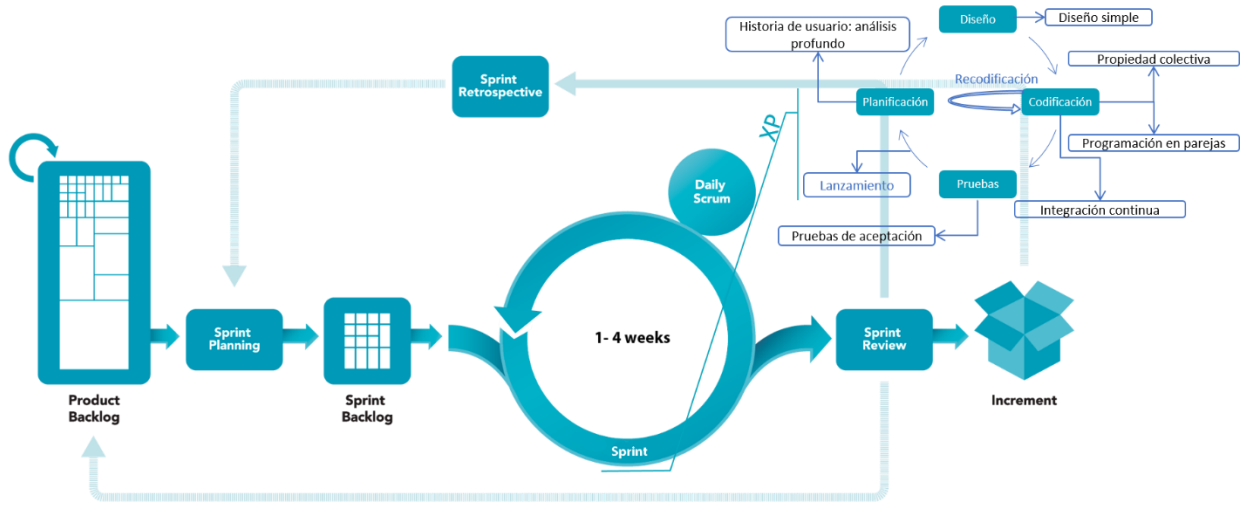


Figura 4. Convergencia metodológica - Elaboración propia.

En la Figura 4 se observa como marco central el flujo de trabajo Scrum, aplicando XP durante la ejecución del Sprint de la siguiente manera:

- En la etapa de planificación se define la lista de las historias de usuario.
- En el desarrollo del entregable se utilizaron las buenas prácticas de XP.

2.8. Lenguaje de programación

Un lenguaje de programación es una estructura utilizada para definir un conjunto de instrucciones que van a ser procesadas por un ordenador. “Es el medio de comunicación entre el programador y una computadora” (Buitrago, 2010) p.1, puesto que las computadoras no entienden el lenguaje humano, sino programas escritas en el lenguaje que estas usan. El término programación nos lleva a crear (o desarrollar) software, conocido también con el nombre de “un programa”. No existe un mejor lenguaje de programación, ya que todos tienen sus bondades como sus debilidades, va a depender de la experiencia del desarrollador para hacer que este funcione de la mejor manera en una situación específica (Liang, 2012).

Según informa el portal de (TIOBE, 2019) que Python ganó el título de lenguaje de programación del año 2018. Python ganó 3.62%, seguido de Visual Basic .NET y Java. En la Figura 5, se puede ver el ranking de estos lenguajes.

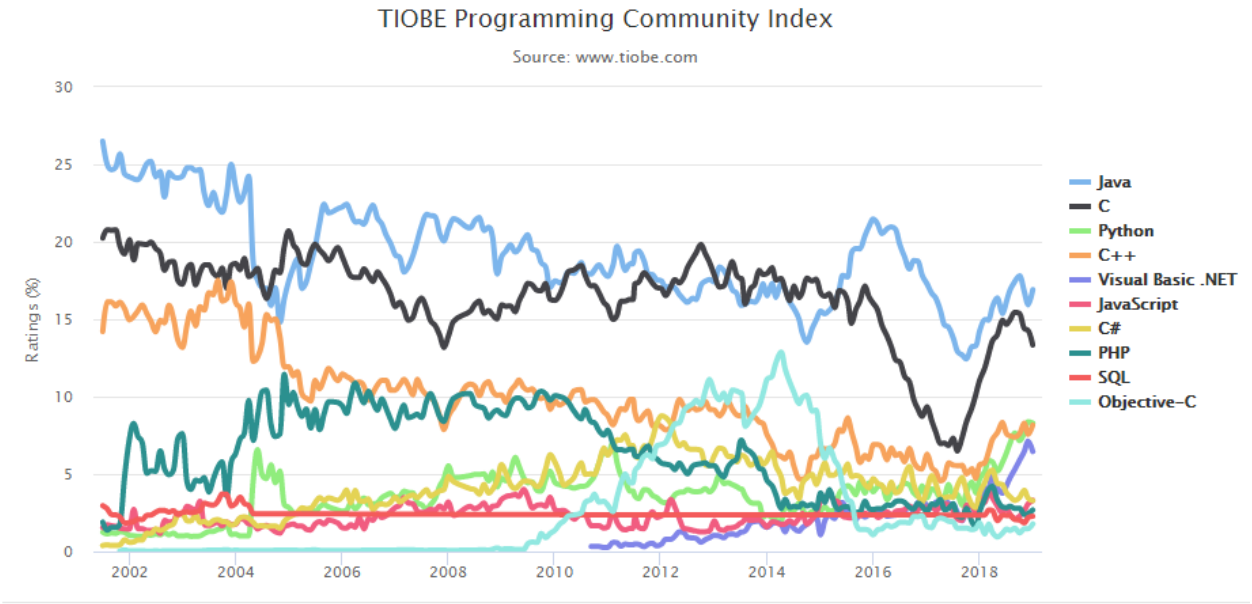


Figura 5. Ranking de lenguajes de programación (TIOBE, 2019).

A continuación, se explica brevemente las virtudes de alguno de ellos.

2.8.1. Java

Java diseñado en 1991 y conocido con el nombre de "oak", luego renombrado como tal en el año 1995, rediseñado para el desarrollo de aplicaciones web. Este, ha cobrado una enorme popularidad en el mundo del desarrollo gracias a las características que provee: simple, orientado a objetos, distribuido, interpretado, robusto, seguro, arquitectura neutra, alto rendimiento, multi-hilo y dinámico (Liang, 2012).

2.8.1.1. Spring

Es un framework de código abierto que se puede utilizar en cualquier aplicación desarrollada en java. Ocupa el primer lugar en el ranking de frameworks en el lenguaje Java (ver Figura 6).

Framework	Score
Spring	89
JSF	81
Google Web Toolkit	77
Vert.x	66
Dropwizard	66
Struts	64
Wicket	63
Vaadin	63
Restlet	55
Tapestry	53
Ninja	52
ZK	50
Stripes	43
Rapidoid	39
Cocoon	33

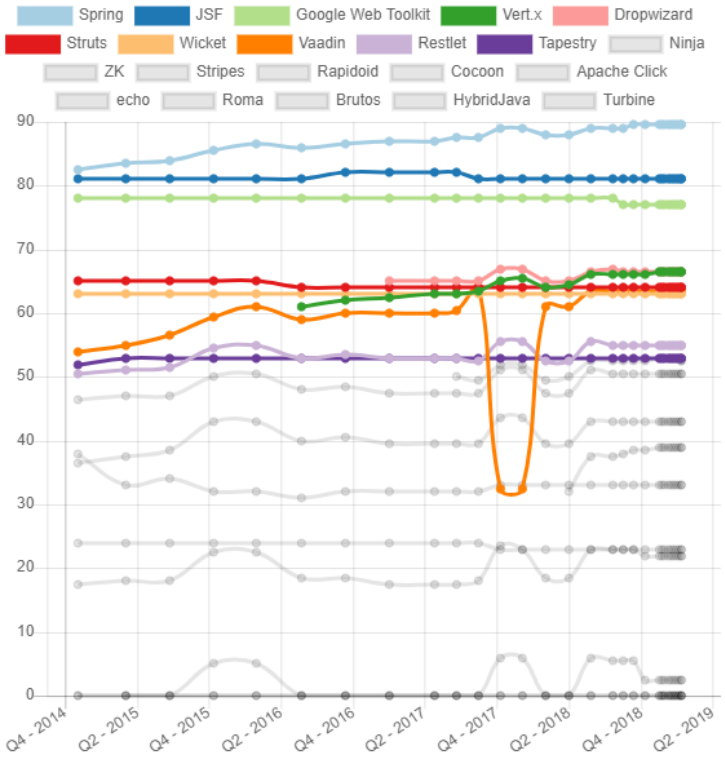


Figura 6. Ranking de frameworks de Java (HotFrameworks, 2019).

2.8.2. JavaScript

JavaScript es un lenguaje de programación interpretado, usado principalmente para la creación de sitios web dinámicos. No requiere de ninguna compilación y se ejecuta con facilidad en la mayoría de los navegadores web (Suehring & Valade, 2013). En base a JavaScript se han desarrollado muchas librerías tales como: TypeScript, Jquery, Ajax, Angular, React, entre otros.

2.8.2.1. AngularJS

Es un framework MVW (Model View Whatever) de Javascript construido por Google con el propósito de crear aplicaciones web de una sola página. Se puede usar para implementar Modelo Vista controlador u otro modelo de patrones. Puedes crear módulos, servicios, constantes, controladores, filtros, directivas, entre otras cosas (Grant, 2014). Favorece la simplicidad y agilidad de las aplicaciones web. Los últimos años se ha mantenido liderando respecto de los demás frameworks de Javascript (ver Figura 7).

Framework	Score
React	95
AngularJS	95
Angular	92
Vue.js	90
Express	87
Meteor	85
Ember.js	81
Polymer	77
Sails.js	76
Aurelia	70
Koa	69
Dojo	65
OpenUI5	65
Feathers	63
Riot.js	59

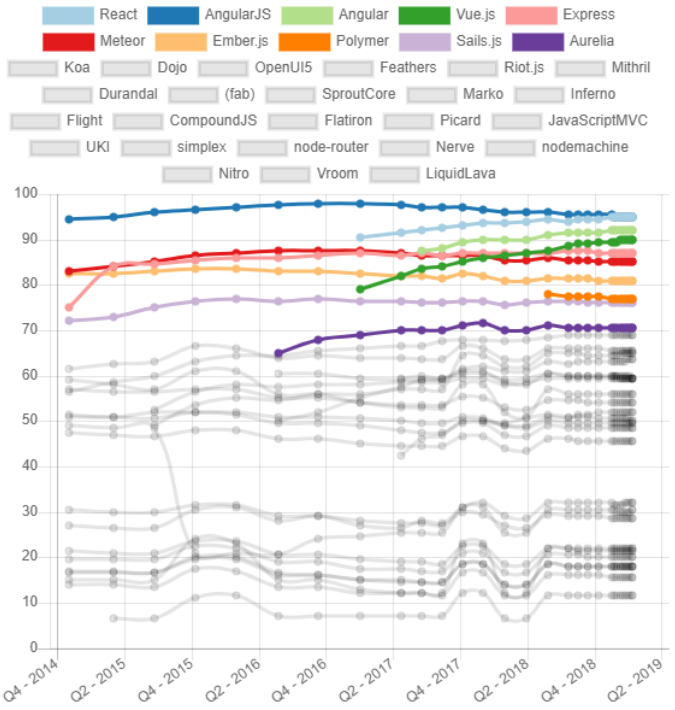


Figura 7. Ranking de frameworks de JavaScript (HotFrameworks, 2019).

2.8.3. Python

Python (2017), describe que Python es un lenguaje de programación claro y potente orientado a objetos con una sintaxis elegante que hace que los programas sean fáciles de leer. Algunas de sus notables características:

- Admite muchas tareas de programación comunes, como conectarse a servidores web, buscar texto con expresiones regulares, leer y modificar archivos.
- Puede ser incrustado en una aplicación para proporcionar una interfaz programable.
- Funciona en cualquier lugar, incluyendo Mac OS X, Windows, Linux y Unix, con versiones no oficiales también disponibles para Android y iOS.
- Es un software libre ya que no tiene costos de descarga ni tampoco en la inclusión de una aplicación. Puede ser modificado y redistribuido libremente.

2.8.3.1. Django

Garcia (2015), explica que Django es un framework creado en Python para el desarrollo Web de forma divertida y en el menor tiempo, puesto que requiere un mínimo esfuerzo. Diseñado para promover el acoplamiento débil y la estricta separación entre las piezas de una aplicación. Sigue un patrón MTV (Model – Template – View, por sus siglas en inglés), similar al MVC.

- Modelo: Capa de acceso a datos, donde se hacen validaciones y las relaciones entre ellos.
- Plantilla: Es la capa de presentación, la que es visible por el usuario final.
- Vista: Aquí se realizan todas las operaciones del negocio. Este accede al modelo y lo comunica con las vistas, es decir como un puente entre el modelo y la plantilla.

Con Django se pueden crear aplicaciones web APIs, con la ayuda de Django REST Framework. Django es asimismo el framework que lidera en el lenguaje de Python (ver Figura 8).

Framework	Score
Django	92
Flask	84
Tornado	73
Bottle	65
Pyramid	63
AIOHTTP	62
web.py	62
web2py	61
Falcon	56
CherryPy	56
Sanic	56
Grok	48
Zope	46
TurboGears	41
Tipfy	35

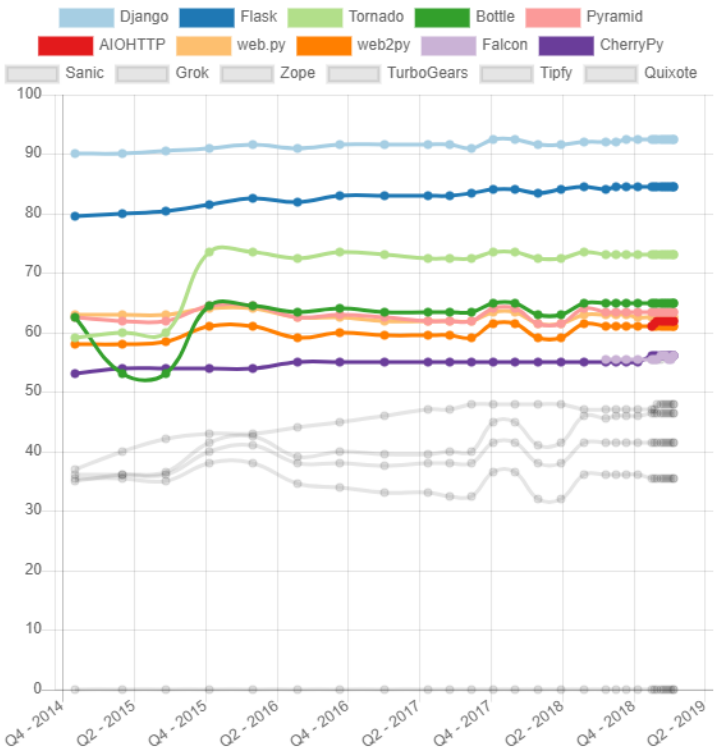


Figura 8. Ranking de frameworks de Python (HotFrameworks, 2019).

2.8.4. PHP

Es un lenguaje de programación tipo script, interpretado de alto nivel, orientado a objetos para entornos web. Es asimismo incrustado en páginas HTML, interpretado por servidores web (Millet, 2009). Se han ido creando una gran variedad de librerías en este lenguaje, como también frameworks tales como: Laravel, Symfony, CodeIgniter, entre otros. Además, existe un gestor de paquetes de PHP llamado Composer, el cual lista las dependencias de un proyecto en un archivo llamado composer.json y, con unos pocos comandos simples, este descarga automáticamente las dependencias del proyecto. (Sturgeon & Lockhart, 2015).

2.8.4.1. Laravel

Es el mejor y más popular framework de PHP en el mundo. Es de código abierto para crear aplicaciones web que permite el uso de código con una sintaxis elegante (Laravel, 2019), ver Figura 9.

Framework	Score
Laravel	90
Symfony	85
CodeIgniter	84
CakePHP	78
Zend	75
Yii	74
Phalcon	67
Kohana	61
SilverStripe Sapphire	56
PHP Fat-Free Framework	55
FuelPHP	55
Lithium	51
Horde	37
Aura	36
Prado	33

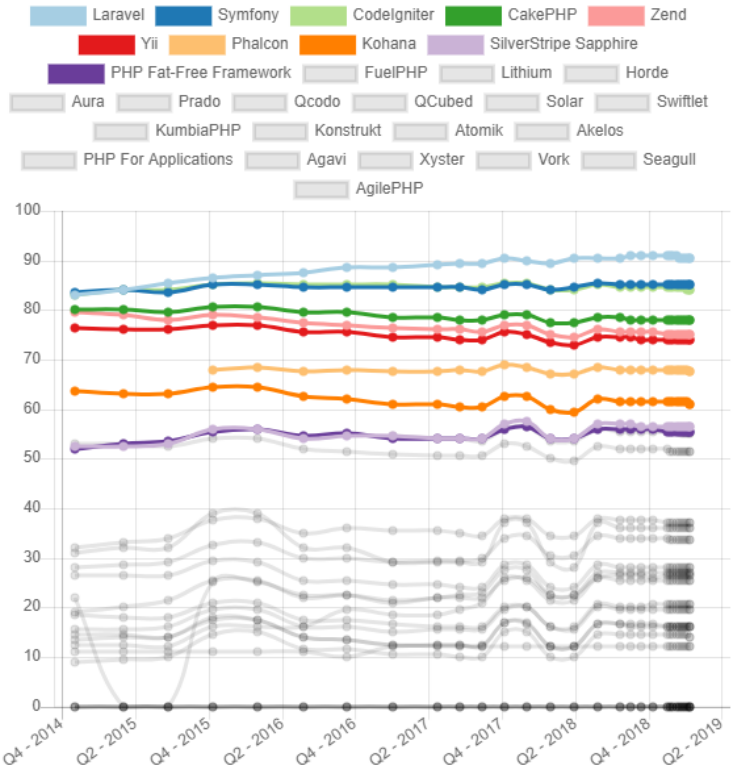


Figura 9. Ranking de frameworks de PHP (HotFrameworks, 2019).

2.8.5. HTML5

HTML5 (Lenguaje Marcado de Hipertexto, versión 5) "no se limita solo a crear nuevas etiquetas o atributos, sino que incorpora muchas características nuevas y proporciona una plataforma de desarrollo de complejas aplicaciones web (mediante los APIs)" (Garro, 2014, p.1).

Vega & Van Der Henst (2011), expresan que "HTML5 es el presente de la web y si no estás asimilando lo que está pasando ya eres parte de la vieja generación de desarrolladores. Eso tendría que tenerte preocupado." Esto nos muestra que debido a que los usuarios suelen conectarse a la web desde cualquier dispositivo móvil, nos insta a estar constantemente actualizados y obliga a los webmasters a usar estas herramientas que nos facilitan la vida.

2.8.6. CSS3

CSS es un lenguaje de diseño basado en hojas de estilos en cascada. Actualmente se cuenta con la versión 3. Con CSS puedes agregar estilos a tus páginas mejorando la presentación del contenido (color, fuente, tamaño del texto, entre otros.). Su estructura es proporcionada por XHTML: cada elemento designa una parte diferente del contenido, y los atributos transmiten más información acerca de esos elementos. Actúa como otra capa para influir en la presentación de esos elementos XHTML cuando se procesan (Schultz & Cook, 2007).

2.9. Sistema Gestor de Base de datos

Iruela (2016), define un Gestor de base de datos como "un sistema que permite la creación, gestión y administración de bases de datos, así como la elección y manejo de las estructuras necesarias para el almacenamiento y búsqueda de la información del modo más eficiente posible". Según el ranking que el portal web DB-Engines (2019) muestra, Oracle lidera, seguido por MySQL, SQL Server, PostgreSQL, y otros (ver Figura 10).

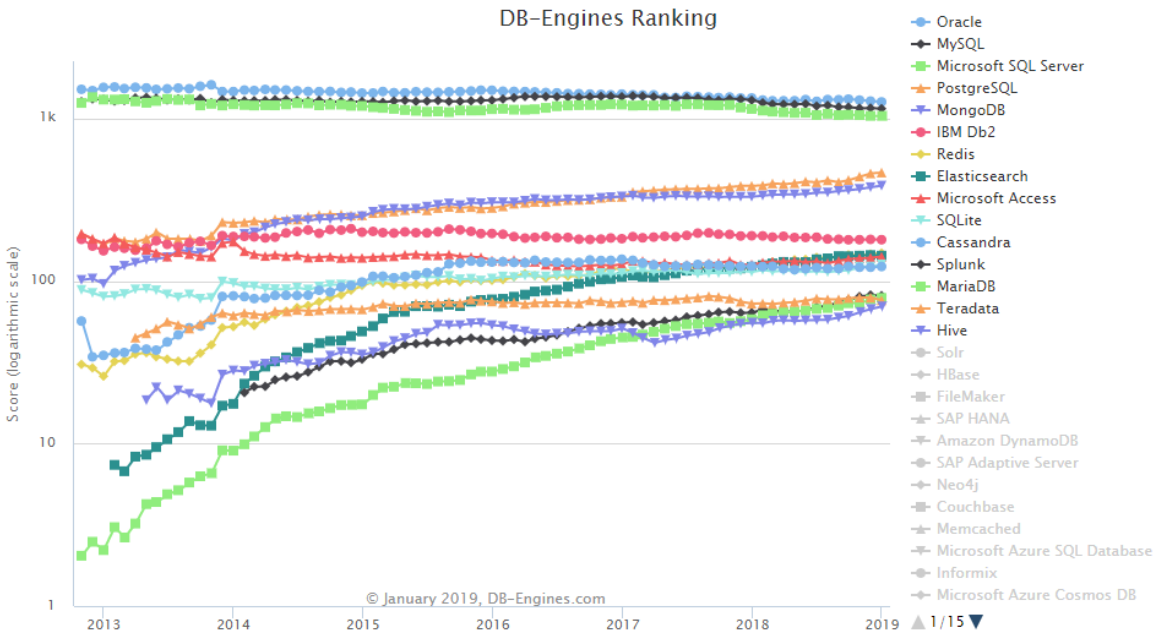


Figura 10. Ranking de Bases de Datos (DB-Engines, 2019).

2.9.1. Oracle

Es un gestor de base de datos relacional de propiedad de Oracle Corporation. Es considerado el gestor de base de datos por excelencia, el más completo y robusto, destacado por su soporte de transacciones, estabilidad, escalabilidad y multiplataforma. Asimismo, es considerado de los más caros, pero tiene una versión EXPRESS gratis para pequeñas instalaciones o usuarios personales (Iruela, 2016).

2.9.2. MySQL

Es un gestor de base de datos relacional, multihilo y multiusuario, desarrollado bajo licencia dual. Por un lado, se ofrece bajo la GNU GPL, pero, también ofrece licencias de uso a aquellas empresas que deseen incorporarlo en productos privados. Es considerada como la base de datos de código abierto más popular del mundo (Matthew Norman, 2004). Ocupa el segundo lugar, después de Oracle, indicando su usabilidad respecto de los demás.

2.9.3. Microsoft SQL Server

Es un gestor de bases de datos relacionales de Microsoft, está basado en el lenguaje Transact-SQL, capaz de soportar grandes cantidades de datos de manera simultánea por muchos usuarios. Entre sus principales características están la escalabilidad, estabilidad y seguridad, soporta procedimientos almacenados, incluye también un potente entorno gráfico de administración. Posee la capacidad de trabajar en modo cliente-servidor, es decir, conteniendo la información centralizada en servidores que luego serán accedidos por los clientes. Su adquisición es costosa para proyectos grandes, sin embargo, cuenta con una versión gratuita que permite usarlo en entornos pequeños (Iruela, 2016).

2.9.4. PostgreSQL

PostgreSQL es un gestor de base de datos relacional de código abierto. Ofrece muchas ventajas además de ser gratuita, soporta alta concurrencia de usuarios, estabilidad, confiabilidad, ahorros de costos de operación y una amplia variedad de tipos de datos (Iruela, 2016). Posee una capacidad de soportar múltiples esquemas al cual se puede administrar separadamente, además de permitir conectarse desde una amplia variedad de lenguajes de programación.

2.10. Herramientas de desarrollo de software.

2.10.1. Navicat

Navicat Premium fue considerado en el 2018 como el mejor administrador de bases de datos (DBA) (Navicat, 2018). Es una potente herramienta que permite manipular bases de datos como Oracle, PostgreSQL, MySQL, SQLite y MongoDB. Capaz de poder ser accedida de manera local y remota.

2.10.2. Herramientas colaborativas

2.10.2.1. Git

Git es un sistema de control de versiones distribuido de código abierto y gratuito, con proporción escalable desde un proyecto pequeño hasta otro de mayor magnitud, con rapidez y eficiencia (Software Freedom Conservancy, 2019). Fue desarrollado por Linus Torvalds y actualmente es uno de los más usados desde pequeñas empresas hasta grandes como Google. “El control de versiones es un sistema que registra cambios en un archivo o conjunto de archivos a lo largo del tiempo para que pueda recuperar versiones específicas más adelante” (Chacon & Straub, 2014).

2.10.2.2. GitHub

Chacon & Straub (2014) sostiene que "GitHub es el host más grande para los repositorios Git, y es el punto central de colaboración para millones de desarrolladores y proyectos". GitHub permite la creación de proyectos públicos (cuenta gratuita) y privados (cuenta premium, a menos que se use GitHub Education). Este repositorio provee de múltiples funcionalidades, entre las más usadas está la creación de "branches" o ramas, "releases" versiones, agregar colaboradores en el proyecto, visualización de la participación del equipo, entre otras. En el 2018 esta plataforma fue adquirida por Microsoft.

2.10.2.3. GitLab

Es un servicio de control de versiones y desarrollo de software colaborativo basado en Git. Permite el trabajo colaborativos gracias a un monitoreo integrado que posee (Chacon & Straub, 2014). "GitLab es la primera aplicación única para el desarrollo, la seguridad y las operaciones de software que habilita las DevOps simultáneas (Desarrollo colaborativo), lo que acelera el ciclo de vida del software y mejora radicalmente la velocidad de los negocios" (GitLab, 2019).

2.10.2.4. Asana

Asana es una plataforma de gestión de trabajos en equipos mejorando la comunicación y colaboración. Permite crear proyectos, agregar miembros al equipo para poder crear tareas y realizar el debido seguimiento de manera integral (Asana, 2019). Con una cuenta Premium o Business puedes contar con cronograma, dependencia de tareas, proyectos públicos y privados, portafolios, dependencia de tareas, entre otras cosas más.

2.10.3. Firebase

Es una plataforma para el desarrollo de aplicaciones web y móviles. Entre los productos que ofrece están base de datos en tiempo real, mensajería desde la nube, autentificación, almacenamiento en la nube, entre otros (Firebase, 2019).

2.10.4. Aplicaciones de desarrollo

2.10.4.1. PyCharm

PyCharm es un entorno de desarrollo integrado (IDE) robusto para la construcción de aplicaciones en el lenguaje de programación Python.

JetBrains (2019), empresa promotora de PyCharm, específica que

> Pycharm proporciona código inteligente, inspecciones de código, resaltado de errores sobre la marcha y soluciones rápidas, junto con refactorizaciones de código automatizadas y capacidades de navegación enriquecidas. Asimismo, facilita una gran colección de herramientas listas para usar: un depurador integrado y un corredor de prueba; Perfilador de Python; una terminal incorporada; e integración con VCS principales y herramientas de base de datos incorporadas. Ofrece soporte para varios frameworks de desarrollo web tales como Django, Flask, Pyramid, Google App Engine y web2py.

2.10.4.2. WebStorm

IDE robusto para la construcción de aplicaciones en JavaScript, TypeScript, lenguajes de hojas de estilo, y todos los frameworks más populares. WebStorm proporciona asistencia de codificación avanzada para Angular, React, Vue.js y Meteor respecto a web. En el desarrollo móvil, React Native, PhoneGap, Cordova e Ionic. En la parte del servidor, Node.js y Meteor. En escritorio, a Electrón (JetBrains, 2019).

2.10.4.3. Postman

Postman es el único entorno de desarrollo de API completo, para desarrolladores de API, utilizado por más de 5 millones de desarrolladores y 100,000 empresas en todo el mundo. Postman hace que el trabajo con las API sea más rápido y más fácil al apoyar a los desarrolladores en cada etapa de su flujo de trabajo, y está disponible para usuarios de Mac OS X, Windows y Linux (Postman Inc., 2019).

2.10.4.4. PuTTY

PuTTY es un cliente SSH y telnet, desarrollado por Simon Tatham para la plataforma Windows. Es un software de código abierto que permite conectarnos a servidores remotos y ejecutar comandos en un servidor VPS, realizando la actividad que el usuario desee conveniente, ya sea instalación de programas o configurar el servidor (PuTTY, 2019).

2.10.5. Servidores y herramientas de configuración

2.10.5.1. Gunicorn

Gunicorn es un servidor HTTP de Python Web Server Gateway Interface (WSGI, por sus siglas en inglés) para UNIX. Es ampliamente compatible con varios marcos web, con una implementación simple, ligero en los recursos del servidor y bastante rápido (Gunicorn, 2019).

En este servidor se especifica los metadatos y dependencias, el usuario y el grupo a procesar que va ser comunicado con Nginx. Posterior a ello se especifica el directorio del proyecto y el comando a iniciar el servicio.

2.10.5.2. CentOS Linux

Es una distribución de Linux gratuita y de código abierto lista para ser utilizada por cualquier persona. El código es proporcionado por Red Hat, Inc (The CentOS Project, 2019). Es utilizada como servidores web en empresas y organizaciones de gran tamaño. En la siguiente figura se muestra un diagrama de porcentajes del uso de las distribuciones de Linux (ver Figura 11).

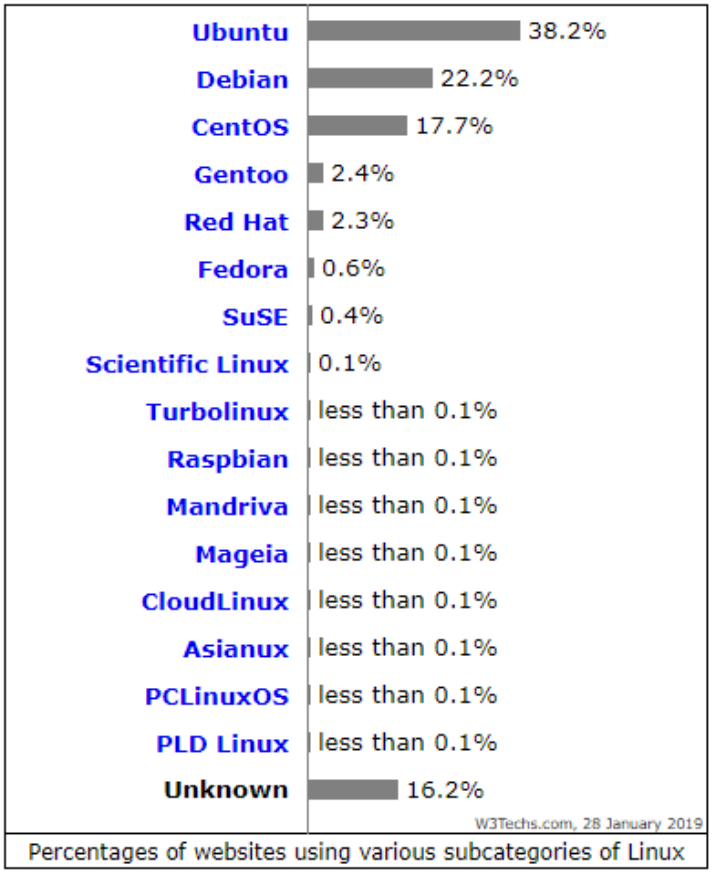


Figura 11. Porcentajes de sitios web que utilizan varias subcategorías de Linux (W3Techs Web Technology Surveys, 2019).

2.10.5.3. Supervisord

Supervisor es un sistema cliente / servidor que permite a sus usuarios controlar varios procesos en sistemas operativos similares a UNIX, compatibles con Linux, Mac OS X, Solaris y FreeBSD (Supervisor, 2019).

Los usuarios pueden configurar sus procesos estableciendo prioridades, agruparlos en grupo de procesos. Con esta herramienta permite tener activos los procesos, reiniciarlos, darlos de baja, y otras configuraciones. Facilita el arranque en caso de que el servidor se reinicie o se apague, haciendo que este inicie los procesos automáticamente.

2.10.5.4. Nginx

NGINX es un servidor HTTP de código abierto, alto rendimiento y proxy inverso, así como un servidor proxy IMAP / POP3. Es conocido por su alto rendimiento, estabilidad, características, configuración simple, bajo consumo de recursos en memoria y alta concurrencia (NGINX, 2019). Asimismo, varios sitios de alta visibilidad usan NGINX, como: Neflix, Pinterest, WordPress.com, GitHub, Heroku, entre otros.

Capítulo 3

Materiales y métodos

3.1. Introducción

En este capítulo se presenta brevemente la descripción del lugar de ejecución, el tipo de investigación y el diseño del experimento a realizar; donde se describirán los pasos que se seguirán para el desarrollo de la investigación.

3.2. Descripción del lugar de ejecución

La investigación a realizar se ejecutará en el área de Bienestar Universitario de la Universidad Peruana Unión, campus Tarapoto; específicamente en las residencias universitarias, ubicada en Jr. Los Mártires N° 340, Morales.

3.3. Tipo de investigación

La investigación presentada es de tipo aplicada, porque busca resolver un determinado problema focalizándose en la búsqueda y aplicación del conocimiento para enriquecer el desarrollo científico. Incluye como justificación adelantos y productos tecnológicos.

3.4. Diseño de la investigación

El diseño de la investigación a aplicar es preexperimental, direccionado a un mismo grupo de interés con en un pre y postest. En esta, se evaluará los efectos del tratamiento en comparación con una medición previa. Este instrumento direcciona al investigador mostrando las acciones a seguir, durante todo el proceso. A continuación, se muestra la Figura 12.

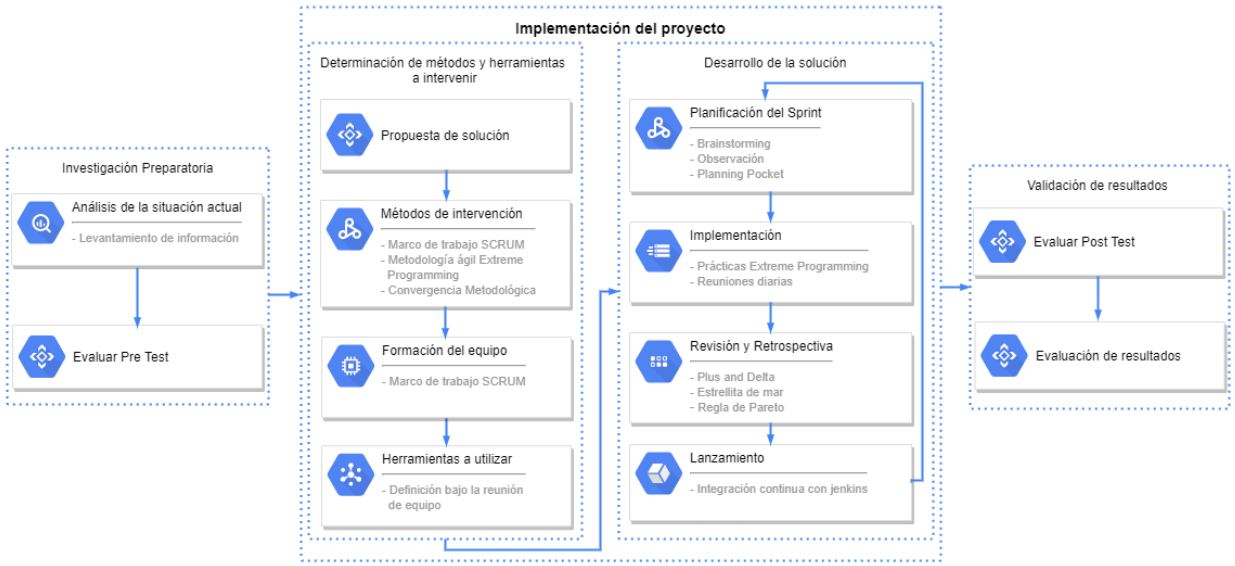


Figura 12. Diseño del proceso metodolódigo de la investigación – Elaboración propia.

3.4.1. Investigación preparatoria

3.4.1.1. Levantamiento de información.

Esta sección comprende la entrevista con el cliente para conocer la situación actual del problema a intervenir, así como también la recaudación de los documentos necesarios que sirvió para orientarnos y tener un mejor conocimiento del negocio.

3.4.1.2. Evaluar Pretest.

En este apartado se realizará la evaluación del pretest con un cuestionario y entrevistas directas a preceptores. Esta evaluación ayudará a diagnosticar la situación actual del problema.

3.4.2. Determinación de métodos y herramientas a intervenir

3.4.2.1. Propuesta de solución

Aquí se analizó la situación actual de cada una de las actividades que se realizan para luego proponer un nuevo modelo automatizado, que permita optimizar el uso de recursos, mejorar la eficiencia, facilitar el flujo de información y reducir el tiempo de procesamiento de las operaciones.

3.4.2.2. Métodos de intervención.

Aquí se hizo una combinación entre dos metodologías: Scrum, en la gestión del proyecto y XP en la construcción de la solución tecnológica. Se hizo una convergencia entre ambas metodologías aplicando las buenas prácticas que estas establecen para satisfacer las necesidades requeridas por el dueño del producto.

3.4.2.3. Formación del equipo de desarrollo.

Esta sección comprende la formación de todo el equipo de Scrum: Equipo de desarrollo, Scrum Master, y dueño del producto. Para esto se hizo la definición de roles de acuerdo al marco de trabajo Scrum.

3.4.2.4. Herramientas a utilizar

Aquí se seleccionó las herramientas de acuerdo a las condiciones del proyecto, metodología y equipo; permitiendo que los miembros puedan converger en uno para contribuir de la mejor manera en el éxito del proyecto. Dicha definición de herramientas se hará previa aprobación en reuniones de equipo.

3.4.3. Desarrollo de la solución

3.4.3.1. Planificación del Sprint.

Teniendo como base el levantamiento de información, se identificó los requisitos funcionales que marcan el ritmo del negocio. Estos a su vez, fueron categorizados y ordenados según la ratio de importancia consignado por el cliente. Estos requisitos fueron plasmados en historias de usuarios. Para esto aplicaremos Brainstorming y Observación.

Por consiguiente, el equipo realizó las estimaciones correspondientes para cada una de las historias de usuario y/o subdivisión de historias en caso sean necesarios; como también la cantidad de puntos (en días) de historias a abordar por sprint. Para las estimaciones usamos la técnica de Planning Pocket y Ojo de buen cubero.

Finalmente se diseñó un plan de entregas con un calendario de presentación de entregables.

3.4.3.2. Implementación.

En esta sección se procedió a transformar los requisitos o historias de usuario definidos para cada Sprint, en funcionalidades probadas y listas para ser usadas por los interesados (cliente). En la construcción de los entregables se usaron varias prácticas de la metodología ágil Extreme Programming: Entregas pequeñas, Diseño Simple, Programación en parejas, Propiedad colectiva del código, Cliente presente, y Estándares de programación.

Los Sprints tuvieron una duración máxima de 4 semanas, con reuniones diarias que permitieron monitorizar el avance de las tareas de las historias de usuarios.

3.4.3.3. Revisión y Retrospectiva.

Después de la implementación, se realizó un análisis de fortalezas y dificultades por cada Sprint, con el propósito de mejorar tanto a nivel de equipo como en cuestión de resultados en los entregables. Para esto, usamos tres técnicas: Plus and Delta, Regla de Pareto, y 5 ¿Por qué?

3.4.3.4. Lanzamiento

Para el lanzamiento de cada entregable se hizo varias configuraciones en coordinación con el área de Redes de la sede central de la universidad. Todos estos procedimientos fueron manuales.

3.4.4. Valoración de resultados

3.4.4.1. Evaluar postest

Esta etapa nos permitió conocer los resultados de los usuarios finales respecto la automatización del control de servicios. Para esto, se aplicó un instrumento validado por juicio de expertos.

3.4.4.2. Evaluación de resultados

Se validará y comparará los resultados obtenidos bajo la prueba estadística "*t de Student"*. Para la validación y análisis de datos obtenidos del cuestionario se realizará con ayuda del software estadístico IMB SPSS Statistics 22.

3.5. Formulación de hipótesis

3.5.1. Hipótesis general

La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

3.5.2. Hipótesis específica

a) La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión de permisos cortos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

b) La aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión de permisos largos brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

c) La aplicación “Control Residente” bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.

3.6. Identificación de variables

Variable Independiente: Aplicación “Control Residente”

Variable dependiente: Control de servicios.

3.6.1. Matriz de consistencia

Tabla 2

Matriz de consistencia – Elaboración propia.

PROBLEMA	OBJETIVO	HIPOTESIS	VARIABLE	DIMENSIÓN	INDICADOR	INSTRUMENTO	METODOLOGÍA
GENERAL: ¿Es eficiente la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto?	**GENERAL:** Determinar la eficiencia de la aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.	**GENERAL:** La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.	**V.D:** Control de servicios	Gestión de permisos cortos Gestión de permisos largos	Número de permisos cortos Número de permisos largos Tiempo de procesar un permiso.	Reporte manual. Reporte generado del software. Encuesta.	**TIPO DE INVESTIGACIÓN:** Aplicativo. **DISEÑO DE INVESTIGACIÓN:** Pre-experimental
				Tiempo	Tiempo de realizar un reporte de permisos	Encuesta. Sistema de información.	**POBLACIÓN Y MUESTRA:** Preceptores **TECNICAS:** Encuesta Entrevista
ESPECÍFICOS: ¿Es eficiente la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos largos brindados a los	**ESPECÍFICOS:** Determinar la eficiencia de la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos cortos	**ESPECÍFICOS:** La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión de permisos cortos brindados a los	**V.I:** Aplicación "Control Residente"	Sistema			

estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto?	brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.	estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.
¿Es eficiente la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos cortos brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto?	Determinar la eficiencia de la aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión de permisos largos brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.	La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión de permisos largos brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.
¿Es eficiente la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad Peruana	Determinar la eficiencia de la aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad	La aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, es eficiente en la mejora de la gestión del tiempo de los servicios brindados a los estudiantes residentes de la Universidad Peruana Unión,

Unión, Filial Tarapoto?	Peruana Unión, Filial Tarapoto.

3.6.2. Operacionalización de variables

Tabla 3

Operacionalización de variables – Elaboración propia.

Variable Independiente	**Objetivos**	**Contenido**	**Método/estrategia**	**Aplicación**
Sistema de información	Determinar la eficiencia de la aplicación "Control Residente" bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto.		SCRUM y Extreme Programming.	Se determinará si la implementación del sistema de información mejora la eficiencia de los recursos.
Variable Dependiente	**Dimensión**	**Indicadores**		
		Nombre	**Atributo**	**Unidad de medida**
Eficiencia de recursos	Gestión de permisos cortos.	Número de permisos cortos.	0,1, 2…, n	Número natural
	Gestión de permisos largos.	Número de permisos largos.	0,1, 2…, n	Número natural
	Tiempo	Tiempo de procesar un permiso.	1,2,3…, n	Segundos
		Tiempo de realizar un reporte de permisos	1,2,3…, n	Segundos

Capítulo 4

Desarrollo de la propuesta

4.1. Investigación preparatoria

4.1.1. Levantamiento de información.

Durante el levantamiento de información se inició por medio de una entrevista in-situ al Mg. Turpo (junio de 2018), encargado del Área de Bienestar Universitario, quién consideró la necesidad de la automatización de un proceso dentro del área: Gestión de permisos de residentes. El área cuenta con dos residencias universitarias y un promedio de 86 residentes por cada semestre.

Para realizar la entrevista respectiva al responsable directo del proceso, fue necesario lo siguiente:

Selección del personal: Teniendo en cuenta los procesos a intervenir, se seleccionó los actores que participarán en la entrevista.

Coordinación de lugar, fecha y hora: Una vez seleccionado a los actores, se coordinó el lugar, fecha y hora de la entrevista, según la disponibilidad de los mencionados.

Preparación del cuestionario: Se elaboró un cuestionario con preguntas de alto nivel para identificar las actividades principales y los problemas que se suscitan.

Ejecución de la entrevista: Llegado el día y hora coordinado, se llevó a cabo la entrevista.

En el levantamiento de información se encontró los siguientes documentos que respaldan las actividades que se realizan en la gestión de permisos.

- Reglamento de residencias universitarias:
- Reporte de permisos mensuales por residente: Formato Excel donde indica la cantidad de permisos de cada residente en un periodo mensual.
- Tarjeta de Permisos: Tarjeta física donde se registra el permiso manualmente. En ella se registra el nombre del residente, la fecha y hora de salida, la fecha y hora de regreso, motivo de salida y el lugar.

En esta sección se elaboró la Pila de Producto, que viene a ser la lista de requisitos, o historias; cosas que el cliente desea. Para la creación de las historias de usuario se tuvo en consideración los niveles de prioridad y los niveles de riesgo, los mismos que se especifican en cada una de ellas.

Tabla 4

Nivel de riesgo

Riesgo	**Valor**
Baja	1
Media	2
Alta	3

Tabla 5

Nivel de prioridad

Prioridad	**Valor**
Baja	1
Media	2
Alta	3

A continuación, se presenta la pila de producto mediante historias de usuario según el marco de trabajo Scrum. Tiene algunos atributos, tales como: Usuario, prioridad del negocio, riesgo, descripción y los criterios de aceptación.

Tabla 6

Historia de Usuario N° 1 – Elaboración propia.

Historias de Usuario					
Número: 01		**Usuario**: Preceptor (a)			
Nombre de la Historia: Registro de residentes					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como preceptor (a) deseo llevar un registro de los residentes y/o sincronizarlos del sistema académico, con los siguientes datos: nombres, apellidos, tipo documento, número de documento, código universitario, estado civil, genero, email, fecha de nacimiento, teléfono, datos del apoderado; para realizar las demás operaciones de permisos, seguimiento, entre otros.					
Criterios de aceptación:					
* Los datos pueden ingresarse o modificarse por los preceptores previa autenticación.					
* Los registros deben ser configurados semestralmente.					
* Los nuevos registros pueden ser registrados manualmente en caso de no encontrarse actualizado en el sistema académico para ser sincronizado a la aplicación.					
* Pueden darse de baja los residentes durante un determinado semestre.					
* Los residentes se agruparán por género (varones – preceptores, mujeres – preceptoras).					

Tabla 7

Historia de Usuario N° 2 – Elaboración propia.

Historias de Usuario					
Número: 02		**Usuario**: Preceptor (a)			
Nombre de la Historia: Registro de apoderados					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como preceptor (a) deseo tener un registro de los apoderados de cada residente, para facilitar la comunicación con ellos en el momento que se requiera.					
Criterios de aceptación:					
* Los datos pueden ingresarse o modificarse por los preceptores previa autenticación.					
* Los registros deben ser configurados semestralmente.					
* No todos los datos a ingresar son requeridos.					

Tabla 8

Historia de Usuario N° 3 – Elaboración propia.

Historias de Usuario					
Número: 03		**Usuario**: Director de Bienestar Universitario			
Nombre de la Historia: Registro y configuración de preceptores					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como Director de Bienestar Universitario deseo poder registrar y configurar a los preceptores encargados de las residencias de varones y de mujeres; con el propósito de que estos puedan tener acceso a los registros de los residentes a cargo para poder monitorizarlos constantemente.					

Criterios de aceptación:

* Se deben configurar semestralmente.

* Se pueden dar de baja en caso de que un preceptor no continúe laborando.

* Se deben registrar datos básicos como: nombres, apellidos, correo y teléfono.

* Solo deben poder registrarse y actualizarse por usuarios administradores.

Tabla 9

Historia de Usuario N° 4 – Elaboración propia.

Historias de Usuario					
Número: 04		**Usuario**: Preceptor (a)			
Nombre de la Historia: Registro de permisos					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como preceptor (a) deseo llevar un registro de los permisos que otorgo a los residentes con el propósito de llevar un control de los mismos. Estos deben contener la siguiente información: Motivo, lugar, fecha y hora de entrada, fecha y hora de salida.					
Criterios de aceptación:					
* Los permisos pueden ser registrados por preceptores y por parte de los residentes (en este caso se llaman solicitudes).					
* Los permisos deben ser aceptados mediante un botón de confirmación, previa visualización de los datos del permiso y un reporte histórico de sus permisos. Esta acción solo puede ser realizada por los preceptores encargados.					
* Los permisos pueden ser modificados por el residente, antes de ser aceptado. Luego de ser aceptado, solo el preceptor puede modificar y/o ampliar el permiso en caso que se requiera.					
* Los permisos podrán ser visualizados por preceptores y apoderados.					

Tabla 10

Historia de Usuario N° 5 – Elaboración propia.

Historias de Usuario					
Número: 05		**Usuario**: Preceptor			
Nombre de la Historia: Consultar los estados de los permisos					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como preceptor (a) deseo poder consultar los diferentes estados de los permisos en ejecución, agrupados en tres secciones: Solicitados, diarios, rechazados. Los solicitados, me permitirá monitorizar los solicitados para aceptarlos o rechazarlos; los diarios, visualizar los estados: Por salir, por ingresar, tardanza, prórroga, no ingresa para actuar en cada uno de ellos según se estime conveniente; y los rechazados, con el propósito de poner en espera alguna solicitud con observaciones.					
Criterios de aceptación:					
* Los reportes deben mostrar solo información necesaria que permita tomar decisiones con facilidad.					
* Debe proporcionar información del momento y permitir actualizar la lista en caso sea necesaria, ordenados por fecha y hora.					
* Los permisos rechazados deben tener la opción de restaurar en caso de no estar vencido.					

Tabla 11

Historia de Usuario N° 6 – Elaboración propia.

Historias de Usuario					
Número: 06		**Usuario**: Preceptor			
Nombre de la Historia: Consultar permisos de sus apoderandos					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta

Iteración Asignada:

Programador responsable: por definir

Descripción:

Como preceptor (a) deseo que el apoderado pueda visualizar los permisos de sus apoderandos con la finalidad de estar comunicado de las entradas y salidas del campus universitario.

Criterios de aceptación:

* Deben tener un acceso a la aplicación web para visualizar el historial de permisos de sus apoderandos.

Tabla 12

Historia de Usuario N° 7 – Elaboración propia.

Historias de Usuario					
Número: 07		**Usuario**: Personal de seguridad			
Nombre de la Historia: Registrar entradas y salidas de los residentes					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como personal de seguridad (a) deseo registrar las entradas y salidas de los residentes con permisos aceptados por sus preceptores encargados, para tener un mejor control de los mismos.					
Criterios de aceptación:					
* Deben acceder a la aplicación móvil con el rol de personal de garita.					
* Solo pueden registrar a aquellos que tienen permisos aceptados y no vencidos.					
* El registro debe ser mediante dos opciones: Lector de código QR o búsqueda personalizada.					
* Deben poder visualizar quienes van a salir e ingresar del campus universitario.					

Tabla 13

Historia de Usuario N° 8 – Elaboración propia.

Historias de Usuario					
Número: 08		**Usuario**: Preceptor(a)			
Nombre de la Historia: Registrar motivos de salida					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como Preceptor(a) deseo agregar, actualizar y eliminar los motivos de salida más recurrentes en caso sea conveniente.					
Criterios de aceptación:					
* Deben ser motivos generales y recurrentes.					

Tabla 14

Historia de Usuario N° 9 – Elaboración propia.

Historias de Usuario					
Número: 09		**Usuario**: Preceptor(a)			
Nombre de la Historia: Registrar lugares					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Yo como Preceptor(a) deseo agregar, actualizar y eliminar los lugares más recurrentes a los que los residentes frecuentan.					
Criterios de aceptación:					
* Deben ser lugares recurrentes.					

Tabla 15

Historia de Usuario N° 10 – Elaboración propia.

Historias de Usuario					
Número: 10		**Usuario**: Director de Bienestar Universitario			
Nombre de la Historia: Registrar usuarios del sistema					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como Director de Bienestar Universitario deseo poder crear usuarios y asignarles un rol específico para acceder al sistema (web y móvil) y realizar las operaciones correspondientes.					
Criterios de aceptación:					
* Se deben tener en cuenta los privilegios asignados a cada rol del sistema					
* Se deben poder actualizar las contraseñas de los usuarios en caso de olvido o solicitud directa.					

Tabla 16

Historia de Usuario N° 11 – Elaboración propia.

Historias de Usuario					
Número: 11		**Usuario**: Preceptor(a)			
Nombre de la Historia: Notificaciones					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como Preceptor(a) deseo que el sistema envíe notificaciones después de realizarse operaciones específicas: registro de solicitud, aceptación o rechazo de un permiso; con el propósito de estar constantemente alertado para tomar acciones de forma inmediata, y dar un buen servicio a los residentes.					

Criterios de aceptación:

* Las notificaciones deben llegar en tiempo real mediante la aplicación móvil.

* Al abrir una notificación debe redireccionar a la sección de la operación realizada, siempre y cuando el usuario se encuentre autenticado.

* Las notificaciones deben ser temporales, es decir, no es necesario tener un historial guardado.

Tabla 17

Historia de Usuario N° 12 – Elaboración propia.

Historias de Usuario					
Número: 12		**Usuario**: Preceptor(a)			
Nombre de la Historia: Enlazar número telefónico a llamadas y WhatsApp					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					
Programador responsable: por definir					
Descripción:					
Como Preceptor(a) deseo que la aplicación enlace los números telefónicos a llamadas y mensajes de WhatsApp para agilizar el hecho de comunicar a un contacto de la aplicación.					
Criterios de aceptación:					
* Cada operación de comunicación debe identificarse con su icono respectivo.					

Tabla 18

Historia de Usuario N° 13 – Elaboración propia.

Historias de Usuario					
Número: 13		**Usuario**: Preceptor(a)			
Nombre de la Historia: Presentar reportes mensuales de permisos					
Prioridad en el negocio:			**Riesgo**		
Media	Baja	Alta	Media	Baja	Alta
Iteración Asignada:					

Programador responsable: por definir

Descripción:

Como Preceptor(a) deseo contar con reportes mensuales de los permisos de cada residente agrupándolos semanalmente con el fin de tener un conteo exacto de cada uno para emitir a instancias superiores o a los interesados (apoderados).

Criterios de aceptación:

* Debe permitir descargar en PDF y EXCEL, además de poder visualizarse en línea.

4.1.2. Evaluar pretest.

Llegado el día y hora coordinado, se llevó a cabo la evaluación pretest con los actores directos que son los preceptores, en donde se les comunicó la finalidad, la estructura y el contenido de la entrevista.

4.2. Determinación de métodos y herramientas a intervenir

4.2.1. Propuesta de solución

Teniendo el panorama de las necesidades del cliente mediante el levantamiento de la información, se consideró necesario la construcción de dos aplicaciones: web y móvil. A continuación, se especifica a nivel de detalle cada una de ellas:

La aplicación web tiene como propósito brindar las configuraciones básicas y necesarias para que entre en función la aplicación móvil. Además, brindará reportes periódicos a administradores y preceptores.

La aplicación móvil será la responsable de cada una de las operaciones: Solicitud, aprobación, seguimiento, control de entradas y salidas de cada uno de los permisos, y otras. Estas operaciones serán en tiempo real, facilitando la intervención de los usuarios e interesados claves en cada uno de estas.

4.2.2. Métodos de intervención

Aquí se hizo una convergencia entre el marco de trabajo Scrum y la metodología ágil XP, garantizando el éxito del proyecto y la calidad del producto final.

Scrum como marco de trabajo prioriza el flujo de trabajo entre sus fases: Inicio, Planificación, Implementación, Revisión, y Lanzamiento. Como Scrum nos dice que hacer, mas no el cómo; muestra flexibilidad al permitir integrar diferentes técnicas y procesos.

XP metodología centrada en aplicar las prácticas de programación. De tal manera, se hará uso de las 12 prácticas repartidas entre las fases de Scrum; además de otras técnicas que ayudan al equipo en la planificación y retrospectiva.

4.2.3. Formación del equipo de desarrollo.

Para la formación del equipo seguiremos la guía de Scrum. El equipo Scrum consta de un Dueño de Producto, Equipo de Desarrollo, y un Scrum Master. Asimismo, es necesario mencionar algunas características que se deben tener en cuenta.

- Deben ser multifuncionales, es decir, que estos poseen todas las competencias que se requieren para llevar a cabo todas las tareas del proyecto.
- Deben ser capaces de entregar productos de forma iterativa e incremental.
- Debe oscilar entre 5-9 integrantes.
- No existen jerarquías entre los miembros del equipo.

En este proyecto todos los miembros del equipo estuvieron en el mismo lugar, donde se compartió todos los eventos de manera presencial con constantes interacciones entre ellos.

Teniendo en cuenta este marco de trabajo se procedió a formar el equipo que iba a estar a cargo de todo el proyecto (ver *Tabla 19*).

Tabla 19

Distribución del equipo Scrum - Elaboración propia.

División del Equipo	**Responsable**	**Cargo**
Dueño del Producto	Josué Turpo Chaparro	Jefe de Bienestar Universitario
Scrum Master	Eliacer Fernandez Guevara	Desarrollador Full Stack
	Ulices Julca Huancas	
Equipo de Desarrollo	Pedro Joel Gómez Rengifo	Desarrollador Full Stack (Back-end y Front-end)
	Jhan Arly Sánchez Tarrillo	
	Heber Quelion Flores Chura	

A continuación, se presentará una descripción breve de las habilidades y responsabilidades de cada uno de los miembros del equipo.

Eliacer Fernandez Guevara: Egresado de la carrera de Ingeniería de Sistemas, experimentado en el área de desarrollo de software. Posee habilidades de varios lenguajes de programación, entre ellos, Java, Python, PHP, JavaScript y frameworks frontend y backend que agilizan el proceso de desarrollo. Además, se destaca por su excelente desempeño dirigiendo equipos de desarrollo.

Ulices Julca Huancas: Egresado de la carrera de Ingeniería de Sistemas, experimentado en el lado del backend del desarrollo de software. Sus diferentes habilidades en varios lenguajes de programación y diferentes frameworks, le permiten un excelente desempeño en el equipo reflejándose en la calidad de los productos que implementa.

Pedro Joel Gomez Rengifo: Egresado de la carrera de Ingeniería de Sistemas, experimentado en el desarrollo de software, del lado del frontend y administración de bases de datos. Además, es un aficionado de las aplicaciones móviles, hecho que le llevó a conocer Ionic Framework, una herramienta que permite construir aplicaciones móviles para IOS y Android.

Jhan Arly Sánchez Tarillo: Egresado de la carrera de Ingeniería de Sistemas y con experiencia en frontend y aplicaciones móviles. Posee excelentes habilidades duras y blandas para trabajar coordinadamente en trabajo colaborativo. Entre sus destrezas, está el diseño de interfaces de usuario, y la integración de APIs con el lado del cliente.

Heber Quelion Flores Chura: Egresado de la carrera de Ingeniería de Sistemas con habilidades en el diseño de interfaces de usuario y en la gestión de bases de datos.

4.2.4. Herramientas a utilizar.

En esta sección se describe todas las herramientas tecnológicas que el equipo consideró utilizar para el cumplimiento de los requisitos solicitados por el cliente. Se detalla en la Tabla 20.

Tabla 20

Herramientas utilizadas en la creación de la solución tecnológica – Elaboración propia.

Clasificación	Nombre de la herramienta	Descripción
IDEs	PyCharm	Editor para codificar código escrito en Python. Usado para la construcción del backend del proyecto.
	WebStorm	Editor para codificar JavaScript, TypeScript, y lenguajes de hojas de estilo. Usado en la construcción de la app móvil.
	Postman	Entorno que sirve para hacer pruebas a las rutas que provee la API REST.
	Navicat	Administrador de base de datos. Con esta herramienta se manipuló la base de datos de manera local y remota.
	PuTTY	Programa que permite acceder mediante SSH a servidores de manera remota. Con esta herramienta se configuró el servidor, clonó el proyecto, instaló las dependencias, y otros comandos que pusieron en marcha el software.
Lenguajes de programación, estilos, entre otros.	Python	Lenguaje de programación usado en la construcción del API REST.
	JavaScript	Lenguaje de programación usado entre el API REST y las vistas. Se usó muchos eventos y funciones para enriquecer la interacción con el usuario.

	CSS3	Hojas de estilo que embellece las plantillas HTML. Se usó creando estilos personalizados para el proyecto.
	TypeScript	Lenguaje de programación usado en la construcción de la app móvil en el framework Angular 5.
	HTML5	Lenguaje de etiquetas usadas en la construcción de aplicaciones web.
Bases de datos	PostgreSQL	Motor de base datos que dio soporte a todas las transacciones de información procesados por la aplicación. Se crearon múltiples esquemas, teniendo una proyección escalable (un esquema por cada sede).
	SQLite	Motor de base datos usado en el almacenamiento de información básica que se guarda en los dispositivos móviles (datos de logueo).
	Firebase	De esta herramienta se usó el servicio de mensajería desde la nube, para el envío de notificaciones desde la aplicación central hacia los dispositivos móviles.
Herramientas colaborativas y control de versiones	GitHub	Herramienta colaborativa que ayudó al equipo a subir los cambios del proyecto y poder trabajar siempre en la última versión. Fue usado a nivel de equipo.
	GitLab	Herramienta colaborativa usado a nivel institucional para actualizar las aplicaciones desarrolladas (en nuestro caso, web y móvil). Desde este repositorio se hicieron los respectivos “pulls” al servidor donde la aplicación está en ejecución.

	Git	Sistema de control de versiones que permitió manejar con facilidad todas las modificaciones del proyecto por cada usuario.
Frameworks para el desarrollo web	AngularJS versión 1.7.2	Framework usado para interactuar entre la vista y los datos provenientes del API REST. Se crearon directivas, controladores, servicios, entre otros.
	Bootstrap	Framework de plantillas usado para la creación de las vistas en la versión web de la aplicación.
	Django	Framework escrito en Python. Se usó como base de toda la construcción de la API REST, con la ayuda de django-rest-framework. Se crearon los modelos, vistas, rutas, serializers, y otros propios del framework.
Frameworks para el desarrollo móvil	Ionic	Framework que posee utilidades para desarrollar aplicaciones móviles nativas e híbridas. Permite con una sola aplicación generar aplicaciones para diferentes sistemas operativos: Android, IOS, entre otros.
	Angular 5	Framework para crear aplicaciones web y de escritorio. En este proyecto se usó para crear las funciones que obtienen los datos para mostrarlo en la vista de la app.
	Node.js	Entorno en tiempo de ejecución multiplataforma que permitió gestionar los paquetes que se necesitó en la aplicación, mediante su gestor de paquetes “NPM”.
Herramientas para configurar el servidor	CentOS 7	Se usó como servidor de producción donde se alojó todas las dependencias del proyecto, como el proyecto mismo.

	Nginx	Usado como servidor proxy para llamar a una aplicación en ejecución mediante un protocolo específico.
	Gunicorn	Se usó como servidor HTTP para Python.
	Supervisord	Se usó para controlar los procesos de gunicorn. Favoreció en la supervisión del inicio, reinicio y otros eventos en caso de que la aplicación o el servidor fallaren.

La arquitectura planteada para abordar este proyecto de investigación es como se muestra en la Figura 13. Se muestra de manera global como van a funcionar las aplicaciones (web y móvil), procesando información en JSON brindada por la API Django.

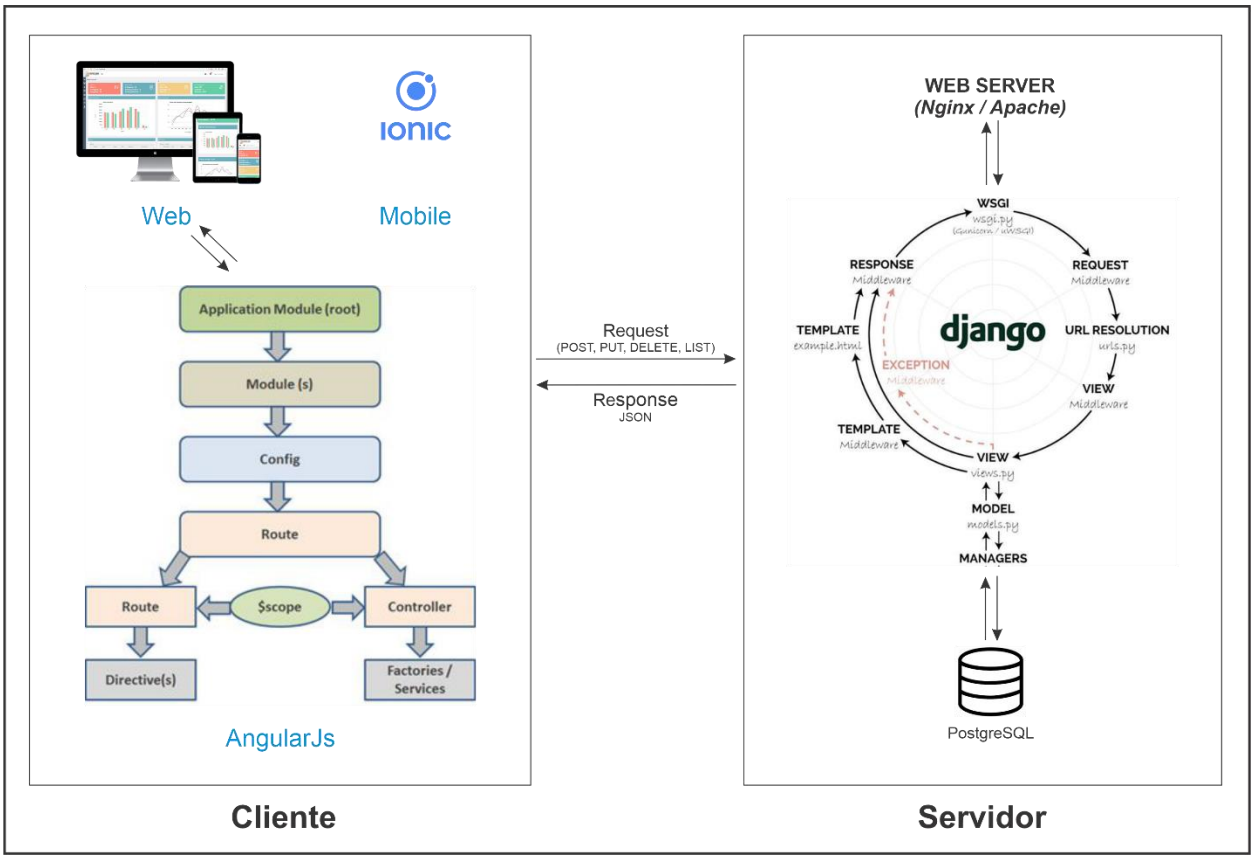


Figura 13. Arquitectura de la solución planteada – Elaboración propia.

La aplicación web como sistema cliente es accedida mediante un navegador web. En ella se pueden realizar las operaciones previa autentificación. Cada una de las solicitudes que el usuario final realiza son interceptadas por la API Django, que responde según sea conveniente, los cuales pueden ser: POST, PUT, DELETE, GET; crear, actualizar, eliminar, ver, respectivamente.

Los dispositivos móviles como Android y IOS, necesitan ser autentificados para acceder a realizar las operaciones; de acuerdo a cada perfil de usuario. Algunas de estas operaciones incluyen envío y recepción de notificaciones, usando así la plataforma de Firebase.

Asimismo, cabe resaltar que el motor de toda la aplicación radica en la API Django, que gestiona la información en coordinación con el motor de Base de Datos.

4.3. Desarrollo de la solución

4.3.1. Planificación del Sprint.

Teniendo como premisa la pila del producto, el equipo de desarrollo analizó las historias de usuarios mediante el juego de planificación y metáfora, en inglés, Planning Pocket. En la Tabla 21 se muestra un resumen de las historias de usuario.

Tabla 21

Resumen de Historias de Usuario – Elaboración propia.

N°	Nombre	Descripción
01	Registro de residentes	Como preceptor (a) deseo llevar un registro de los residentes y/o sincronizarlos del sistema académico, con los siguientes datos: nombres, apellidos, tipo documento, número de documento, código universitario, estado civil, genero, email, fecha de nacimiento, teléfono, datos del apoderado; para realizar las demás operaciones de permisos, seguimiento, entre otros.
02	Registro de apoderados	Como preceptor (a) deseo tener un registro de los apoderados de cada residente, para facilitar la comunicación con ellos en el momento que se requiera.
03	Registro y configuración de preceptores	Como Director de Bienestar Universitario deseo poder registrar y configurar a los preceptores encargados de las residencias de varones y de mujeres; con el propósito de que estos puedan tener acceso a los registros de los residentes a cargo para poder monitorizarlos constantemente.
04	Registro de permisos	Como preceptor (a) deseo llevar un registro de los permisos que otorgo a los residentes con el propósito de llevar un control de

		los mismos. Estos deben contener la siguiente información: Motivo, lugar, fecha y hora de entrada, fecha y hora de salida.
05	Consultar los estados de los permisos	Como preceptor (a) deseo poder consultar los diferentes estados de los permisos en ejecución, agrupados en tres secciones: Solicitados, diarios, rechazados. Los solicitados, me permitirá monitorizar para aceptarlos o rechazarlos; los diarios, visualizar los estados: Por salir, por ingresar, tardanza, prorroga, no ingresa para actuar en cada uno de ellos según se estime conveniente; y los rechazados, con el propósito de poner en espera alguna solicitud con observaciones.
06	Consultar permisos de sus apoderandos	Como preceptor (a) deseo que el apoderado pueda visualizar los permisos de sus apoderandos con la finalidad de estar comunicado de las entradas y salidas del campus universitario.
07	Registrar entradas y salidas de los residentes	Como personal de seguridad (a) deseo registrar las entradas y salidas de los residentes con permisos aceptados por sus preceptores encargados, para tener un mejor control de los mismos.
08	Registrar motivos de salida	Como Preceptor(a) deseo agregar, actualizar y eliminar los motivos de salida más recurrentes en caso sea conveniente.
09	Registrar lugares	Como Preceptor(a) deseo agregar, actualizar y eliminar los lugares más recurrentes a los que los residentes en caso sean convenientes.
10	Registrar usuarios del sistema	Como Director de Bienestar Universitario deseo poder crear usuarios y asignarles un rol específico para acceder al sistema (web y móvil) y realizar las operaciones correspondientes.
11	Notificaciones	Como Preceptor(a) deseo que el sistema envíe notificaciones después de operaciones específicas: registro de solicitud, aceptación o rechazo de un permiso; con el propósito de estar constantemente alerta para intervenir de inmediato.

12	Enlazar número telefónico a llamadas y WhatsApp	Como Preceptor(a) deseo que la aplicación enlace los números telefónicos a llamadas y mensajes de WhatsApp para agilizar el hecho de contactar a un contacto de la aplicación.
13	Presentar reportes mensuales de permisos	Como Preceptor(a) deseo contar con reportes mensuales de los permisos de cada residente agrupándolos semanalmente con el fin de tener un conteo exacto de cada uno para emitir a instancias superiores o a los interesados (apoderados).

Asimismo, es necesario conocer las prioridades que el dueño del producto ha asignado a las historias de usuario. Estas se muestran a continuación en la Tabla 22.

Tabla 22

Priorización de historias de usuario – Elaboración propia.

Prioridad	**Historia**
A	Registro de residentes
I	Registro de apoderados
B	Registro y configuración de preceptores
C	Registro de permisos
D	Consultar los estados de los permisos
J	Consultar permisos de sus apoderandos
G	Registrar entradas y salidas de los residentes
E	Registrar motivos de salida
F	Registrar lugares
H	Registrar usuarios del sistema
K	Notificaciones
L	Enlazar número telefónico a llamadas y WhatsApp
M	Presentar reportes mensuales de permisos

Una vez que se tiene listo la pila del producto, se comienza a realizar las estimaciones de las historias de usuario con el fin de estimar el tiempo, costo y alcance del proyecto. Las estimaciones en cuanto a tiempo, ayudarán a definir el número de *Sprints*, y que historias incluir en cada una.

Para esto, hemos utilizado dos técnicas: Ojo de buen cubero y cálculos de velocidad. En la Tabla 23, se ve reflejada la primera técnica por puntos de historia (punto = día).

Tabla 23

Estimación de historias de usuario – Elaboración propia.

N°	Historia	Ulices	Pedro	Jhan	Eliacer	Heber	Estimación
01	Registro de residentes	8	10	10	8	10	**10**
02	Registro de apoderados	5	6	6	5	7	**6**
03	Registro y configuración de preceptores	5	7	7	5	7	**7**
04	Registro de permisos	10	12	13	11	12	**12**
05	Consultar los estados de los permisos	7	9	8	8	9	**9**
06	Consultar permisos de sus apoderandos	5	6	6	5	6	**6**
07	Registrar entradas y salidas de los residentes	7	9	10	8	9	**9**
08	Registrar motivos de salida	4	4	4	4	4	**4**
09	Registrar lugares	4	4	4	4	4	**4**
10	Registrar usuarios del sistema	14	15	16	14	17	**16**
11	Notificaciones	6	8	8	6	9	**8**
12	Enlazar número telefónico a llamadas y WhatsApp	3	4	3	3	3	**4**
13	Presentar reportes mensuales de permisos	6	6	7	5	8	**7**
	Sumatoria						**102 puntos**

Usando cálculos de velocidad como técnica, constó de dos pasos: Decidir la velocidad, calcular el número de historias a añadir. La fórmula que usaremos es la siguiente:

(Días-hombre disponibles) x (Factor de dedicación) = Velocidad estimada

Tabla 24

Días hombres por mes – Elaboración propia.

Miembro	**Total Días**
Eliacer	20
Ulices	10
Arly	12
Pedro	20
Heber	12
Total días hombre	74

Tabla 25

Variables para definir los Sprints – Elaboración propia.

Variables	**Valores**
Velocidad estimada	60% = 0.6
Numero de desarrolladores	5
Total de días hombre al mes	74
Total de puntos al (60%)	74 * 60% = 44 puntos

Considerando los valores anteriores, al aplicar la formula, tenemos los siguientes resultados:

74 días-hombre x 60% = 44.4 = 44 Puntos.

Con estos resultados se pueden tomar las siguientes decisiones:

- Por cada iteración la velocidad máxima será de 44 puntos de historia como máximo.
- Se tendrá un máximo de 03 *Sprints* o iteraciones.

Tabla 26

Definición de Sprints – Elaboración propia.

N° Sprint	Orden	Nombre	Puntos	Puntos Sprint
1	A	Registro de residentes	10	38
	B	Registro y configuración de preceptores	7	
	C	Registro de permisos	12	
	D	Consultar los estados de los permisos	9	
2	E	Registrar usuarios del sistema	16	33
	F	Registrar entradas y salidas de los residentes	9	
	G	Registrar motivos de salida	4	
	H	Registrar lugares	4	
	I	Registro de apoderados	6	
3	J	Consultar permisos de sus apoderandos	6	32
	K	Notificaciones	8	
	L	Enlazar número telefónico a llamadas y WhatsApp	4	
	M	Presentar reportes mensuales de permisos	8	

4.3.2. Implementación.

4.3.2.1. Introducción

Esta es una de las fases más importantes y divertidas porque es aquí donde los requisitos o historias de usuario se implementan convirtiéndose así en entregables funcionales de software. Sin embargo, es necesario mencionar que la etapa de planificación es la clave del éxito de los proyectos, es allí donde se orquesta el plano de toda construcción, es decir, se establecen las reglas de qué y cómo se va a implementar los requisitos con el fin de cumplir las expectativas del cliente.

¿Existe alguna opción de que pueda salvaguardar el proyecto si es que la etapa de planificación fue deficiente? Sin lugar a duda es evidente el fracaso del proyecto de acuerdo a la experiencia de los desarrolladores.

4.3.2.2. Arquitectura de la solución

Este proyecto tiene un alcance para dos plataformas: web y móvil. Con el fin de satisfacer ambas necesidades, sumado los conocimientos del equipo, se planteó usar la arquitectura Cliente – Servidor, donde separamos el *Back-end* del *Front-end*.

El *Back-end* es eje central que satisface las necesidades tanto de los clientes web, como de los clientes móviles. Este apartado es un proyecto por separado construido en el lenguaje de programación Python, con el framework Django 1.11.

El *Front-end* web, es un cliente que solicita la información al *Back-end* y los muestra al usuario mediante un navegador web. Este apartado fue construido con los frameworks de desarrollo *AngularJS* y *Bootstrap. AngularJS,* para la creación de las páginas y su interacción con los datos; y, por otro lado, *Bootstrap,* para diseñar las interfaces del usuario.

El *Front-end* móvil es un resultado a concretizarse gracias a *Ionic Framework*. Esta herramienta nos permite construir aplicaciones móviles basada en tecnologías web, con la ayuda de *Angular framework*. Una vez finalizada la aplicación con ionic se compila para dispositivos Android y IOS, ver Figura 14.

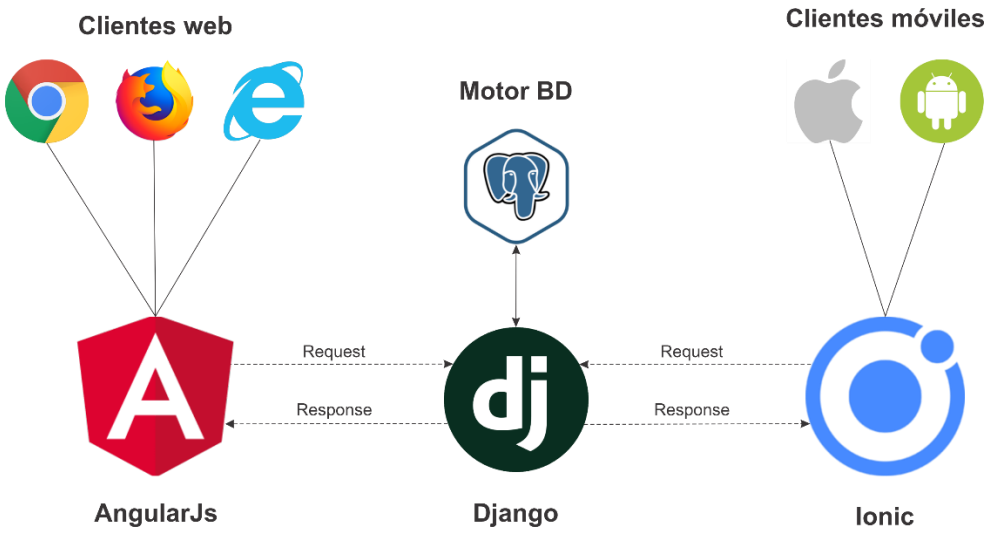


Figura 14. Vision general de la arquitectura de la solución – Elaboración propia.

4.3.2.3. Modelo de base de datos

Los sistemas de información requieren de un gestor de bases de datos para almacenarlos y luego procesarlos. Para este proyecto se utilizó PostgreSQL, gestor de bases de dato relacional orientado a objetos y de código abierto. En un gestor de este tipo, la información se almacena en tablas relacionadas entre sí.

Este proceso de definir que tablas se van a necesitar requiere conocimientos de modelado de datos. Para esto el equipo primeramente analizó los requerimientos del cliente, para identificar qué datos serán almacenados.

Como resultado de este análisis, se obtuvo un esquema de modelado de base de datos según se muestra en la Figura 15.

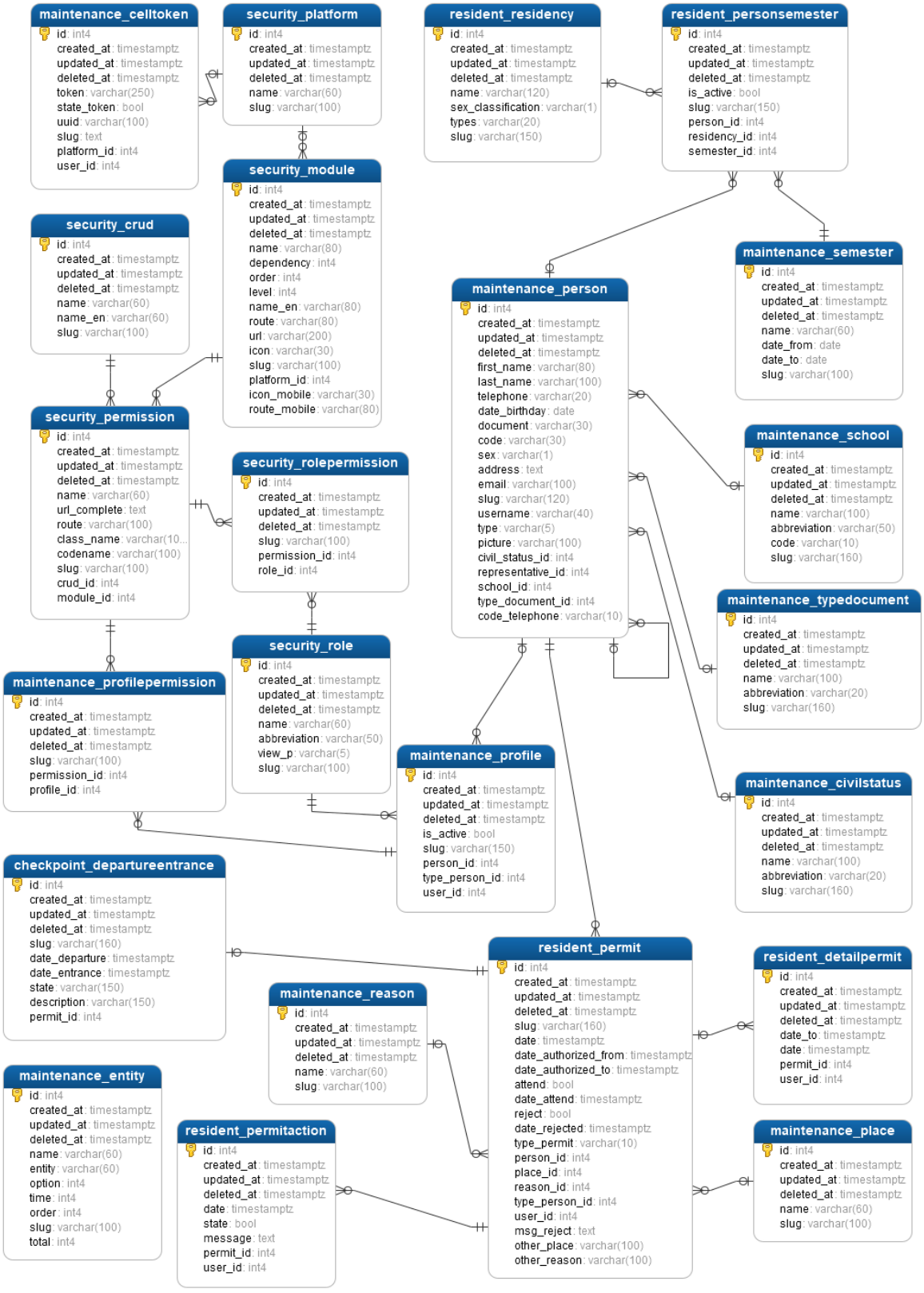


Figura 15. Modelo Entidad Relación del proyecto - Elaboración propia.

4.3.2.4. Construcción

Es en esta sección donde se utilizan la mayoría de la lista de herramientas descritas en secciones anteriores, y las buenas prácticas de XP.

Se realizó *entregas pequeñas* dado que el proyecto se dividió en Sprints cortos que producen entregables funcionales.

Con respecto al *diseño simple* se procuró cumplir con las necesidades del cliente haciéndolo lo más sencillo posible, es decir, interfaces interactivas y fáciles de comprender. No solo se abordó a las interfaces, sino también a la parte del código, ordenado, simple y fácil de mantener.

En cada iteración, se tuvo que utilizar la *programación en parejas*, básicamente en aquellas actividades que implicaban cierta complejidad de análisis y desarrollo. Asimismo, el equipo necesitaba estar pendiente de los avances de cada miembro, en caso de dependencia de tareas. Para esto se definió usar herramientas colaborativas: Git, GitHub para el equipo en etapa de desarrollo, y GitLab en etapa de producción. Estas actividades responden a la práctica de XP llamada *Propiedad colectiva del código.*

Por cada término de la iteración se realizaron pruebas a las nuevas funcionalidades. Para ello se usó el debugger de PyCharm, postman, entre otros.

En el último Sprint, con los nuevos conocimientos aprendidos en el proyecto se vio conveniente optimizar algunas secciones del código que eran claves en el tiempo de respuesta al cliente. Es así como se usó la práctica de *refactorización.*

Con la ayuda de los frameworks tanto en backend como en frontend, se usó algunos estándares de programación, puesto que estos, a la vez, estos ya lo integran. Sin embargo, el equipo mantuvo esas buenas prácticas, como también definió la forma de escribir código.

4.3.2.4.1. Backend

En el lado del Backend, el cual fue escrito con Python, fue necesario usar algunas librerías escritas en Python. La dinámica del trabajo se basó en construir un API Rest con la ayuda de Django, Django Rest framework y demás librerías que se mencionan en la Tabla 27.

Tabla 27

Librerías escritas en Python usadas en el Backend – Elaboración propia.

Nombre de la librería	Versión
Django	1.11.11
django-cors-headers	2.2.0
django-cors-middleware	1.3.1
django-filter	1.1.0
djangorestframework	3.7.7
Pillow	5.0.0
psycopg2	2.7.4
Qrcode	6.0
virtualenv	15.2.0

Las librerías en Python se instalan y administran por un sistema de gestión de paquetes llamado "pip". Es de la siguiente forma: **pip install** Django=1.11.11 o **pip install** Django.

Al momento de instalar se puede indicar la versión de la librería a instalar, en caso de no especificarlo, se instalará la última versión.

Con la finalidad de evitar incompatibilidades en los cambios de versiones de Python y sus librerías, y con el propósito de garantizar el funcionamiento de las diferentes versiones del proyecto, Python ofrece la creación de entornos virtuales independientes por cada proyecto. Consiste en crear un directorio adicional y configurarlo para la instalación de las librerías a usar en un determinado proyecto. A continuación, se muestra un ejemplo de uso en comando de consola.

```
$ mkdir myapp && cd myapp
$ git clone repostorio_url myproject
$ virtualenv -p python3.4 virtual_dir
$ source virtual_dir/bin/activate ó virtual_dir/Scripts/activate
$ (virtual_dir) cd myproject
$ pip install -r requirements.txt
```

Django trabaja con una gran aproximación al patrón de diseño MVC, llamado MTV (Modelo, plantilla, vista). En este proyecto se usó las bondades de Django, creando en cada módulo los archivos *models.py, serializers.py, views.py* y *urls.py.* A continuación, se describe rápidamente de cada uno.

- El archivo *models.py* contiene una descripción de la tabla de la base de datos, como una clase Python. A esto se lo llama el modelo. Usando esta clase se pueden crear, buscar, actualizar y borrar entradas de tu base de datos usando solo código Python en lugar de escribir declaraciones SQL repetitivas. En este proyecto se creó una clase abstracta llamada *TimeStampedModel* donde le indica a la clase que va a ser un modelo de Django; además adicionamos en esta clase algunas configuraciones propias, ver Figura 16 y Figura 17.

```python
class TimeStampedModel(models.Model):
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
    deleted_at = models.DateTimeField(blank=True, null=True)
    objects = ManagerMain()
    objects_all = ManagerAll()
    objects_deleted = ManagerDeleted()

    class Meta:
        abstract = True
```

Figura 16. Clase abstracta del modelo – Elaboración propia.

```python
class TypeDocument(TimeStampedModel):
    name = models.CharField(max_length=100, blank=False, null=False)
    abbreviation = models.CharField(max_length=20, blank=True, null=True)
    slug = models.CharField(max_length=160, unique=True, blank=True, null=True)

    def __str__(self):
        return self.name

    def save(self, *args, **kwargs):
        self.slug = slugify(self.name)
        super(TypeDocument, self).save(*args, **kwargs)
```

Figura 17. Modelo en Django – Elaboración propia.

- EL archivo serializers.py permiten que los datos complejos, tales como QuerySets y instancias de modelo a ser convertidos a los tipos de datos de Python nativas que luego pueden ser fácilmente transmitidas en JSON, XMLu otros tipos de contenido, ver Figura 18.

```python
class TypeDocumentSerializer(serializers.ModelSerializer):
    class Meta:
        model = TypeDocument
        fields = ('id', 'name', 'abbreviation', 'slug', 'created_at', 'updated_at',
                  'deleted_at')
```

Figura 18. Serializer básico en Django – Elaboración propia.

- El archivo views.py contiene la lógica de la página, ya sea basada en clases y/o funciones. A esta funciones y clases se les denominan vistas. En la API desempeñan el papel de gestionar las diferentes acciones al modelo: POST, PUT, DELETE, LIST , Figura 19.

```python
class TypeDocumentViewSet(DefaultViewSetMixin, ModelViewSet):
    queryset = TypeDocument.objects.all()
    queryset_deleted = TypeDocument.objects_deleted.all()
    serializer_class = TypeDocumentSerializer
    search_fields = ('name', 'created_at')
    lookup_field = 'slug'
    ordering_fields = '__all__'

    def get_queryset(self):
        queryset = get_queryset_override(self.queryset, self.queryset_deleted, self.request)
        return queryset

    @transaction.atomic
    def post(self, request, *args, **kwargs):...
```

Figura 19. Vistas del API Rest en Django – Elaboración propia.

- El archivo urls.py específica qué vista es llamada según el patrón URL (ver Figura 20).

```python
router = routers.DefaultRouter(trailing_slash=True)
router.register(r'person', PersonViewSet)
router.register(r'profiles', ProfileViewSet)
router.register(r'people', PersonlViewSet)
router.register(r'sync_person', SyncPersonlViewSet)
router.register(r'shifts', ShiftViewSet)
router.register(r'semesters', SemesterViewSet)
router.register(r'type-shifts', TypeShiftViewSet)
```

Figura 20. Urls en Django – Elaboración propia.

Django tiene mucha documentación de las que no se abordará en este proyecto. Se procuró usar algunas de las múltiples bondades que ofrece, según las necesidades de este proyecto.

Una práctica propia del equipo respecto a la forma de codificar el código ha sido en inglés, incluso los nombres de los modelos, funciones, vistas, entre otros. Una de las razones ha sido evitar el corrector de subrayados de Pycharm IDE.

4.3.2.4.2. Frontend web

En este proyecto tenemos dos tipos de clientes: los aplicativos móviles y los usuarios web. Dado el caso se tuvo que codificar por separado.

La aplicación web se construyó con el apoyo y soporte de AngularJS en su versión 1.6.9, y Bootstrap 4, para el diseño de las interfaces. La aplicación web configura todo el panorama para que el control de permisos se realice exitosamente en ambas plataformas. Veamos a continuación, la estructura del directorio del proyecto en angular usado en este proyecto, ver Figura 21.

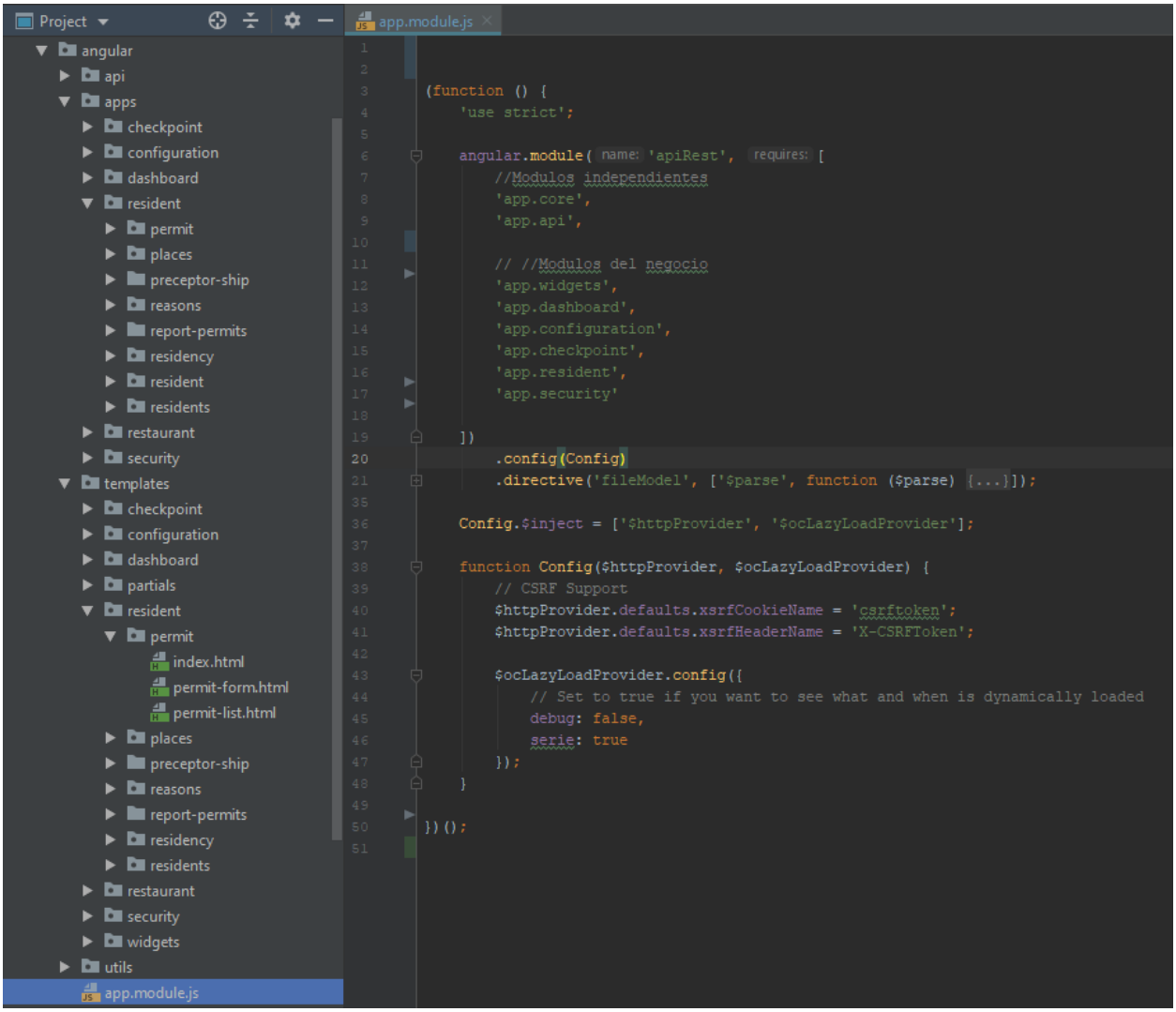


Figura 21. Estructura del directorio del proyecto en AngularJS – Elaboración propia.

Con AngularJS se crearon directivas, servicios, controladores, filtros, rutas, entre otros. Básicamente es el encargado de solicitar los datos al Backend y mostrarlos en las interfaces. Por otro lado, también obtiene los datos de parte del cliente, y los envía al Backend para su respectivo tratamiento (Crear, actualizar, eliminar, reporte, entre otros). En la Figura 22 se ve reflejado lo mencionado.

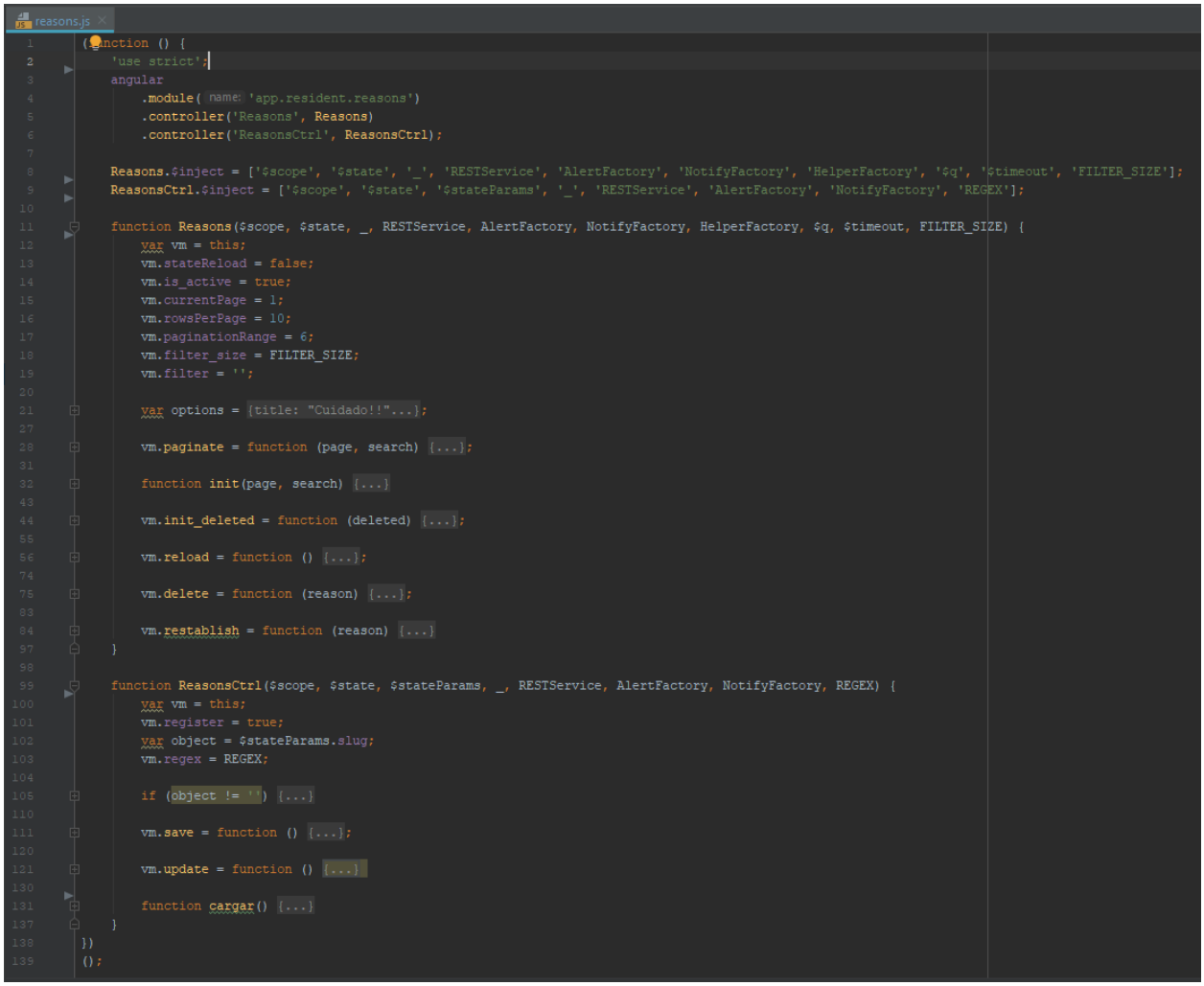


Figura 22. Vista general de las operaciones con AngularJS – Elaboración propia.

Asimismo, la forma de visualizar los datos se realizó en plantillas HTML. Dentro de cada una de estas se inyectaron las directivas que se crearon con la finalidad de reutilizar funcionalidades y reducir código. En la Figura 23 se refleja lo indicado.

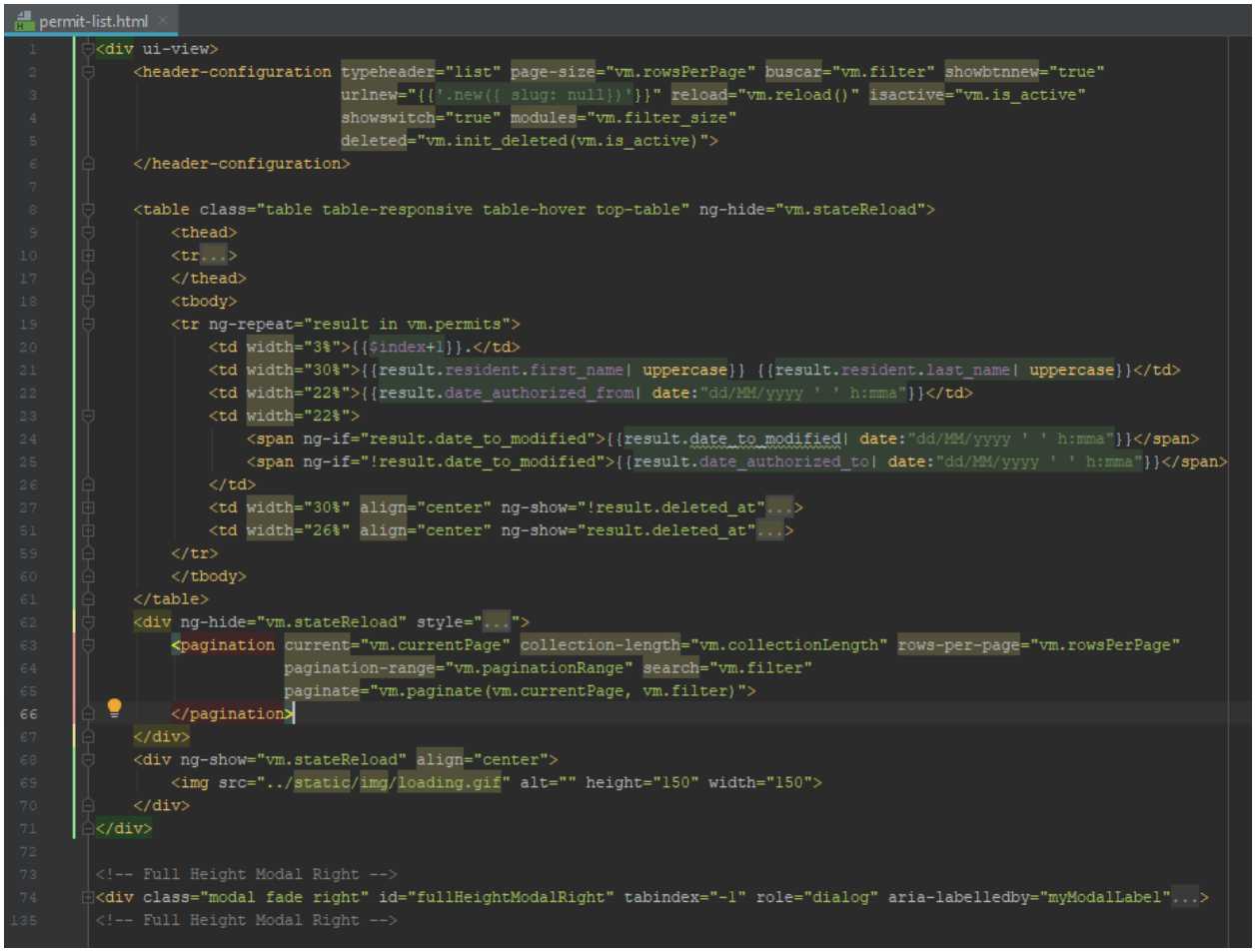


Figura 23. Vistas HTML usando directivas AngularJS – Elaboración propia.

En la Figura 23, se puede observar dos directivas en su interior: <header-configuration></ header-configuration> y <pagination></pagination>. La primera es la encargada de mostrar los filtros de búsqueda y botones que indican acciones (Nuevo, actualizar, entre otros). La segunda permite distribuir la lista de los datos de un modelo en páginas, favoreciendo la rapidez al momento de consultar gran cantidad de datos y haciendo que el cliente pueda solicitar más información en caso requiera.

Todas las listas tienen la misma estructura de como mostrar los datos para hacer que el usuario final se familiarice rápidamente con el aplicativo.

En seguida, pasamos a mostrar las interfaces de la aplicación web agrupados por módulos.

Módulo configuración

En este módulo se configura algunos registros básicos. Entre los más principales destaca las opciones: semestre y sincronización. En la primera, se establece el semestre actual al que se asignaran los residentes y preceptores. En la segunda, se migra la lista de residentes matriculados desde el sistema académico hacia la aplicación "Control Residente", ver Figura 24.

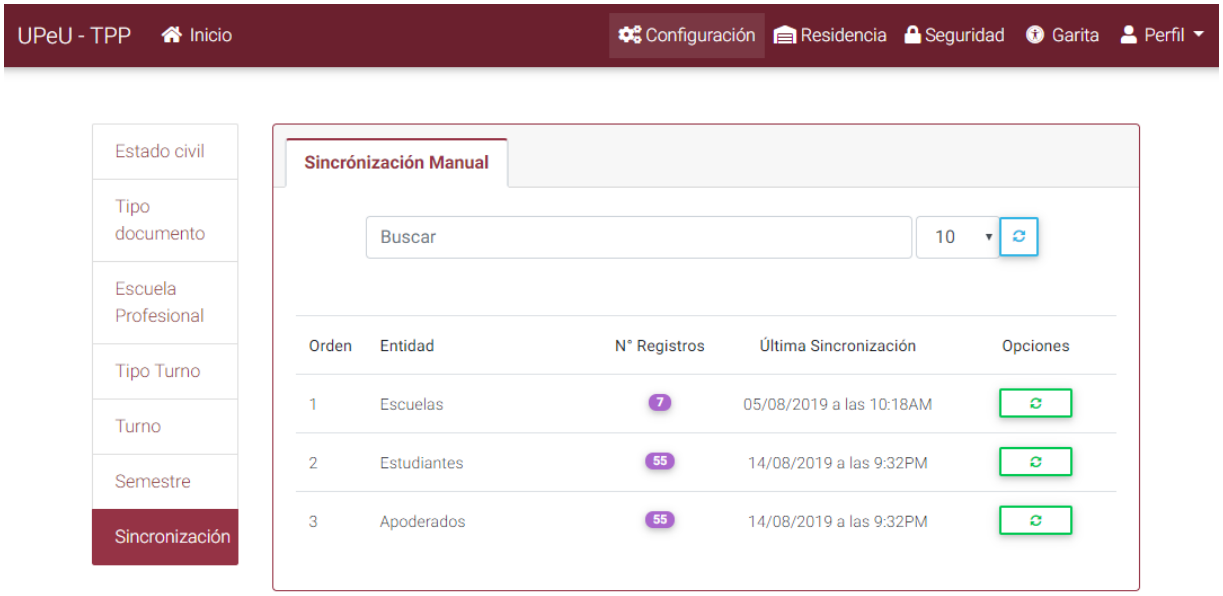


Figura 24. Módulo configuración y submódulos – Elaboración propia.

Modulo Residencia

En este módulo se configura las residencias, los preceptores, lugares de salidas frecuentes, motivos por los que suelen pedir permisos, reportes (periódicos mensuales agrupados por semanas), registro de permisos (en caso que el/la preceptor no cuente con la app móvil), y registro de residentes (en caso que no esté matriculado como residente o que después de matricularse como externo, decida ingresar a las residencias universitarias).

A continuación, en la Figura 25 se muestran las vistas del módulo de residencias.

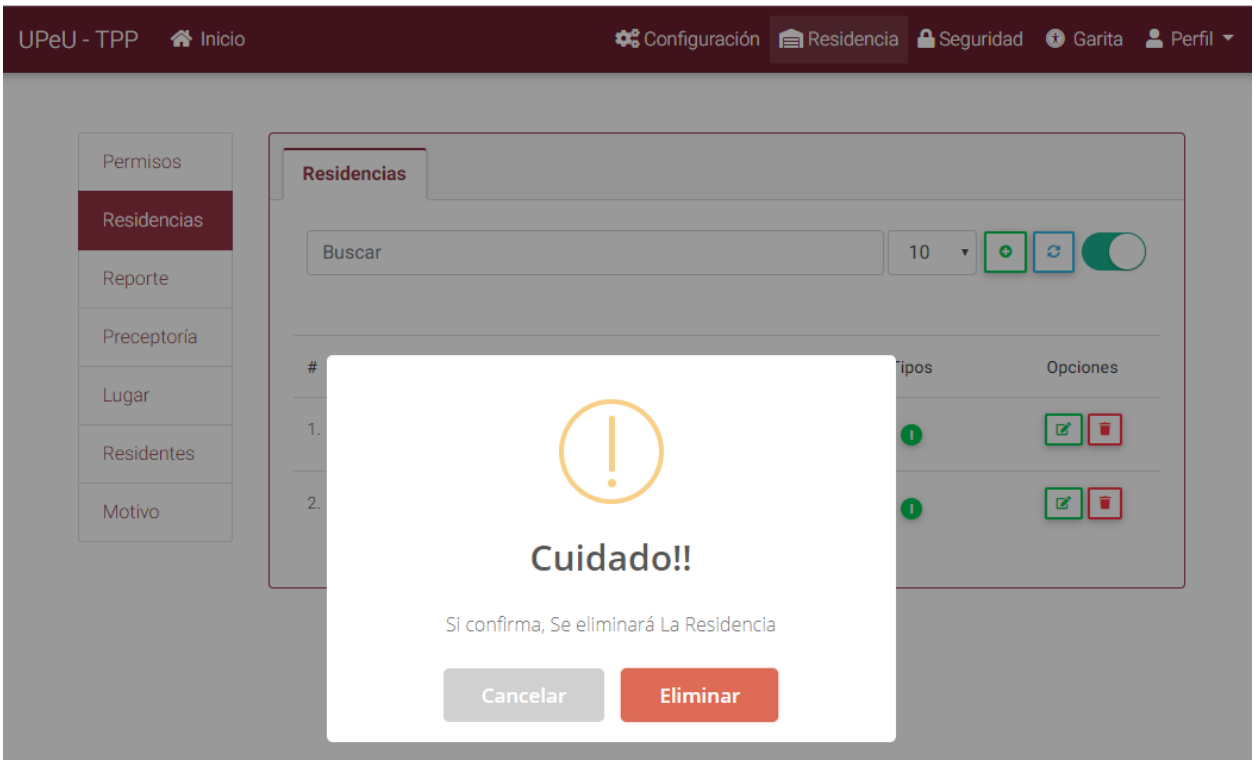


Figura 25. Registro y configuración de residencias – Elaboración propia.

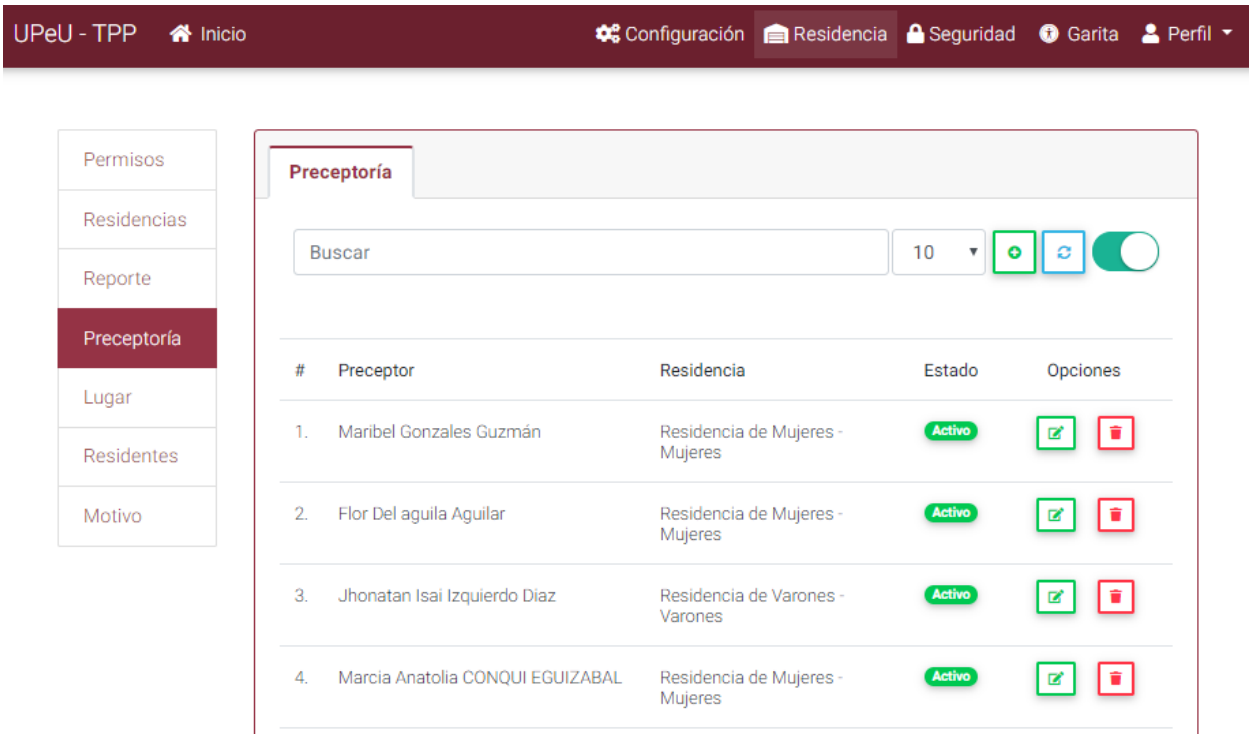


Figura 26. Asignación de preceptores por residencias – Elaboración propia.

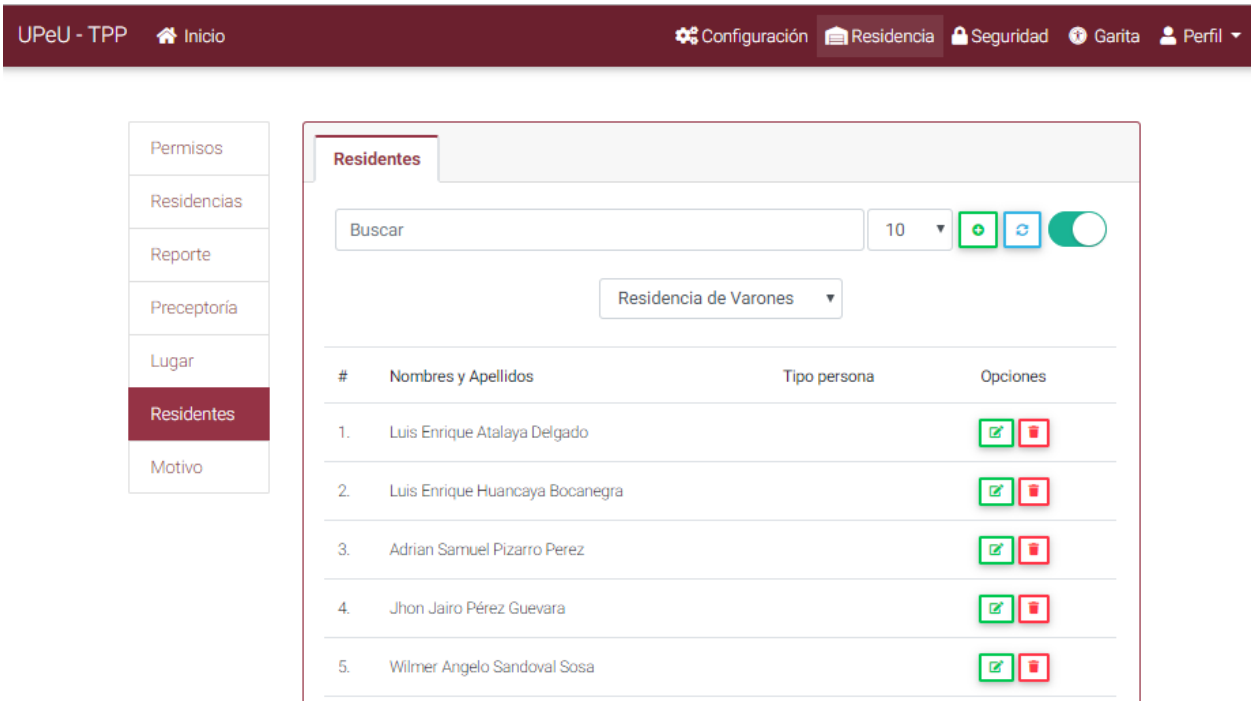


Figura 27. Registro y configuración de residentes – Elaboración propia.

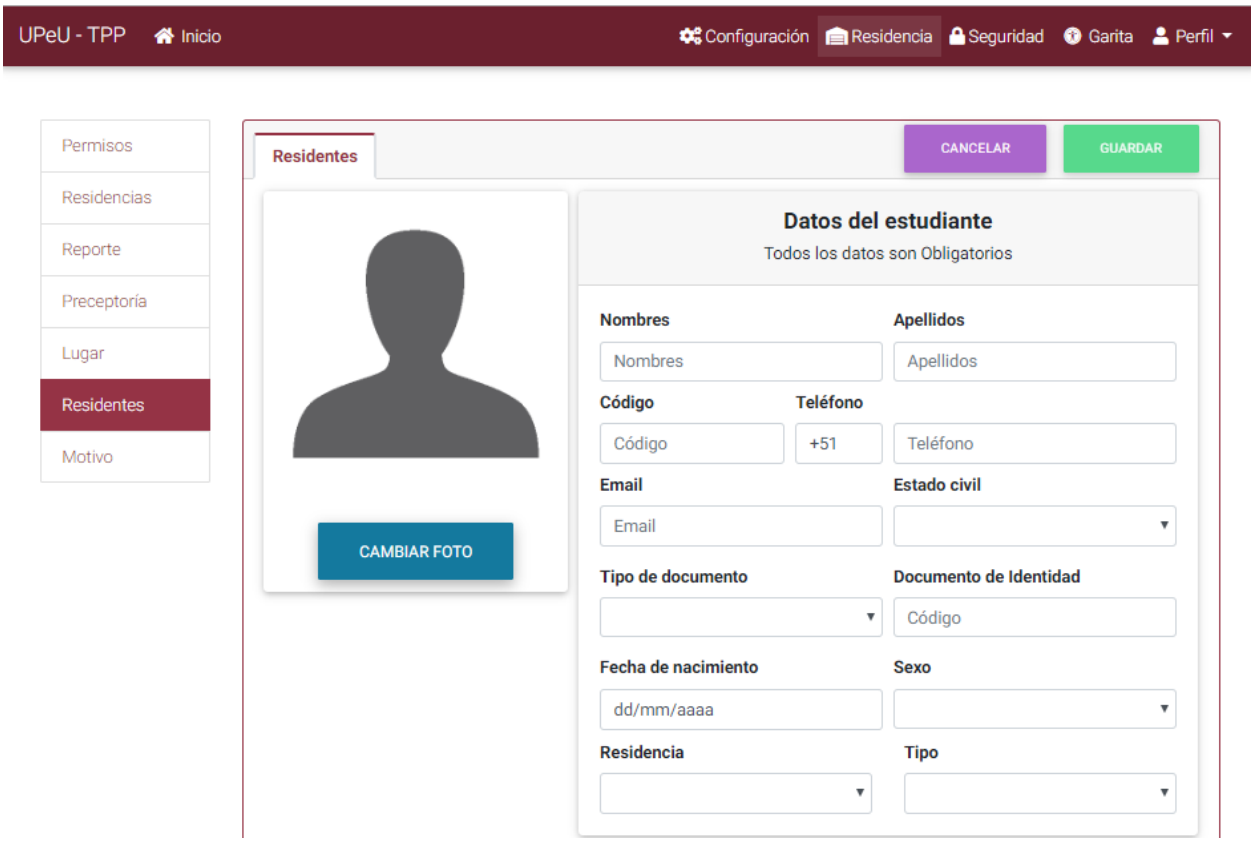


Figura 28. Formulario de registro de residentes – Elaboración propia.

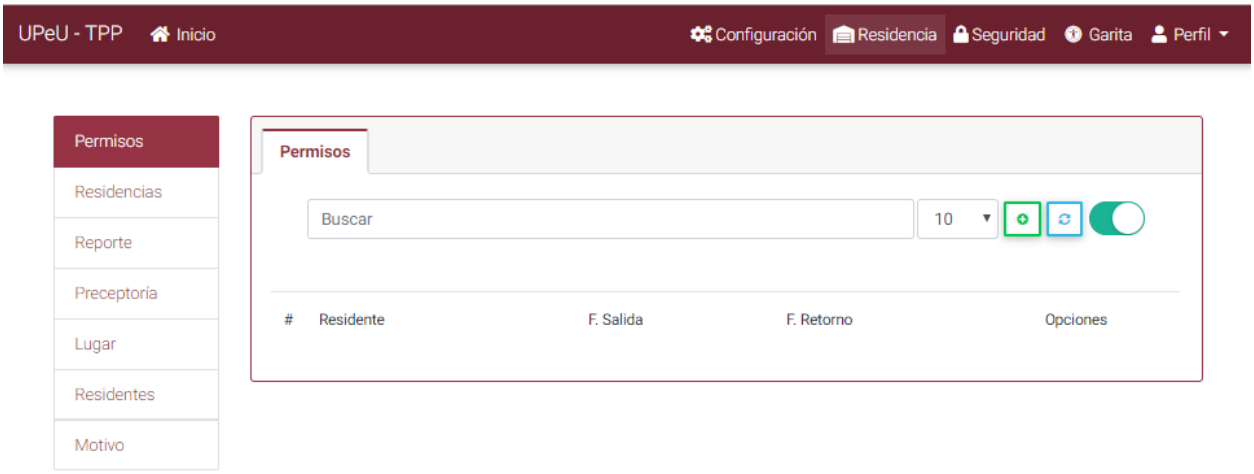


Figura 29. Lista de permisos – Elaboración propia.

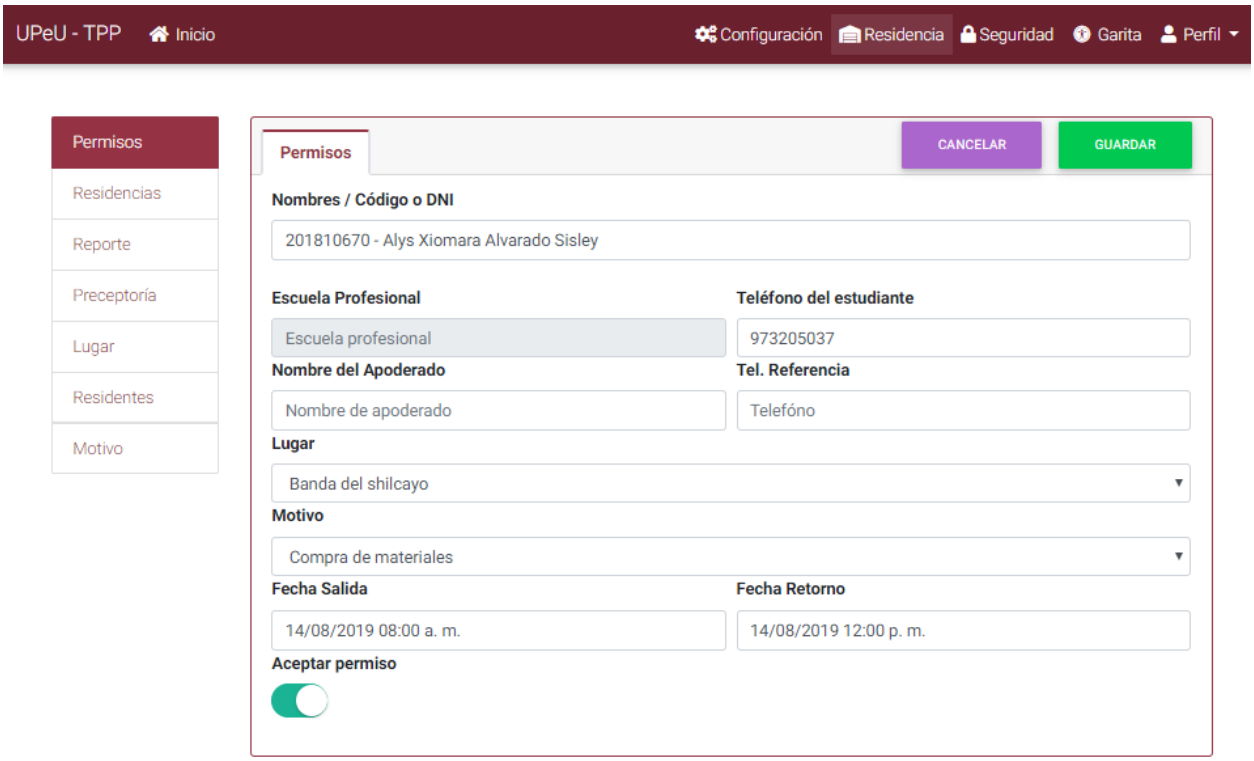


Figura 30. Formulario de registrar de un permiso – Elaboración propia.

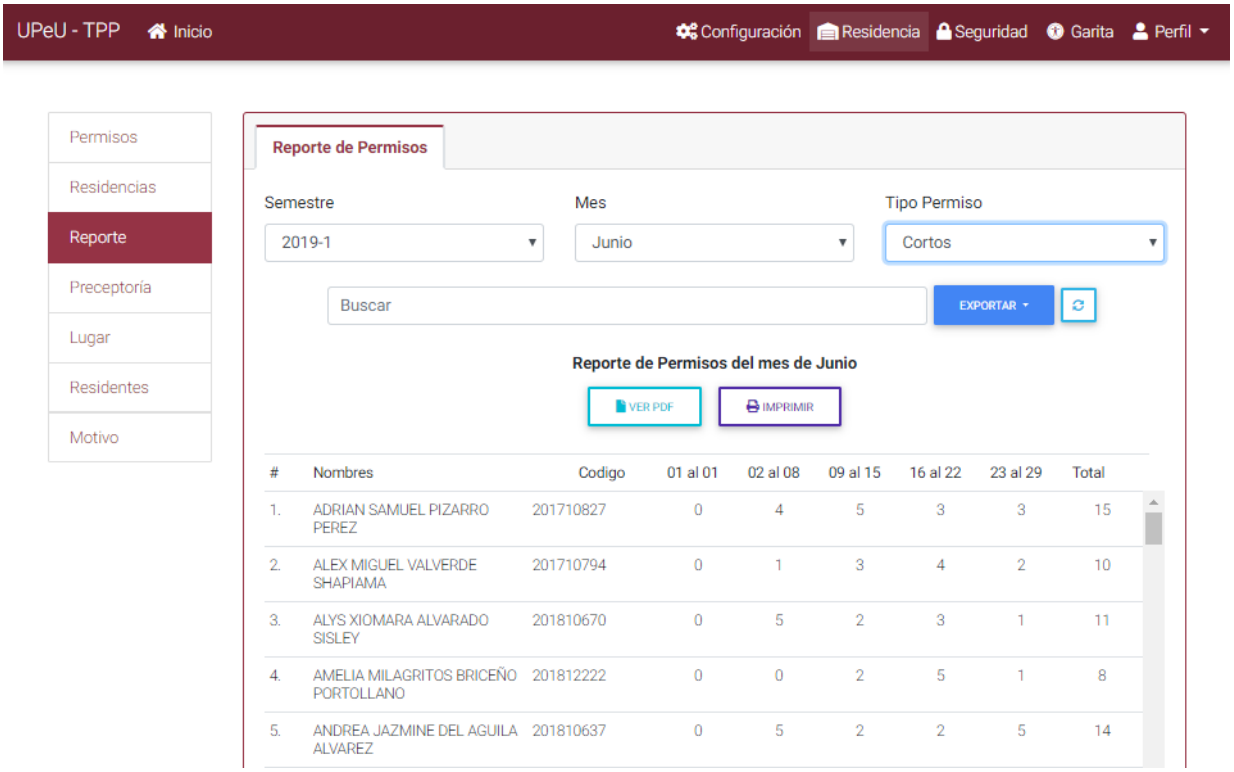


Figura 31. Reporte de permisos cortos – Elaboración propia.

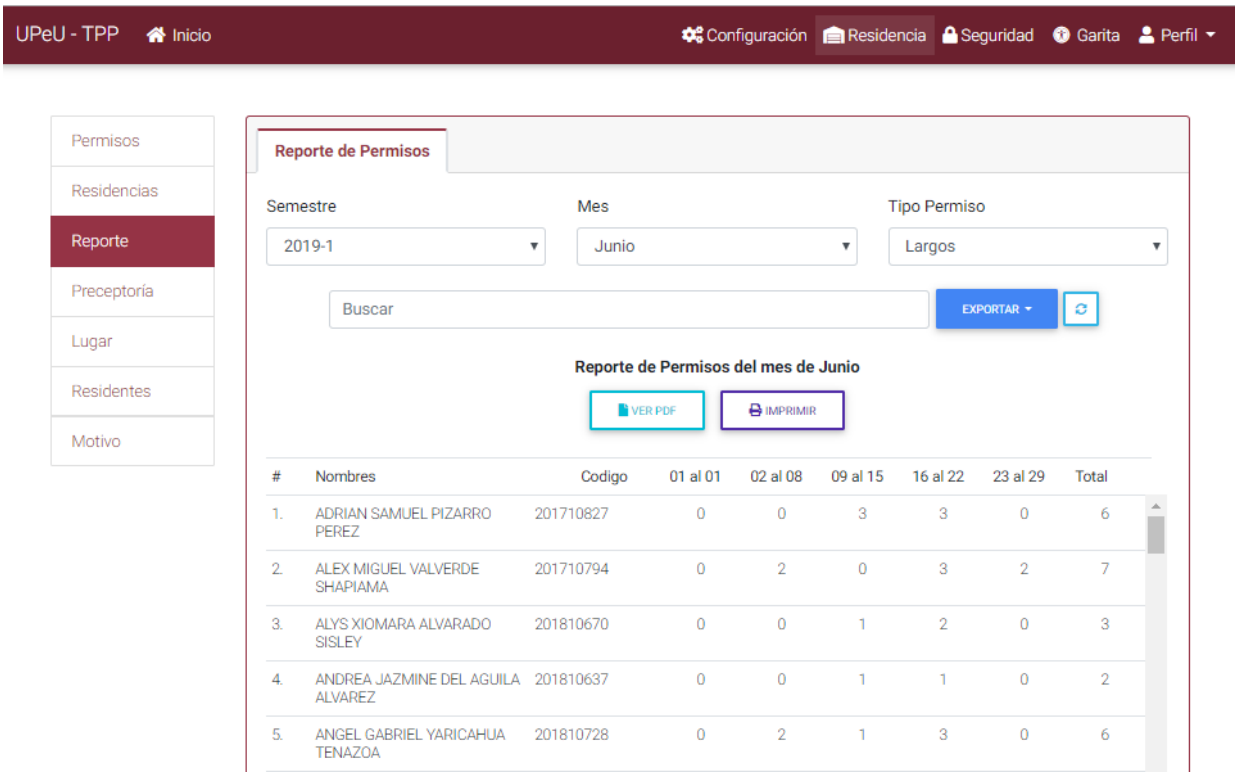
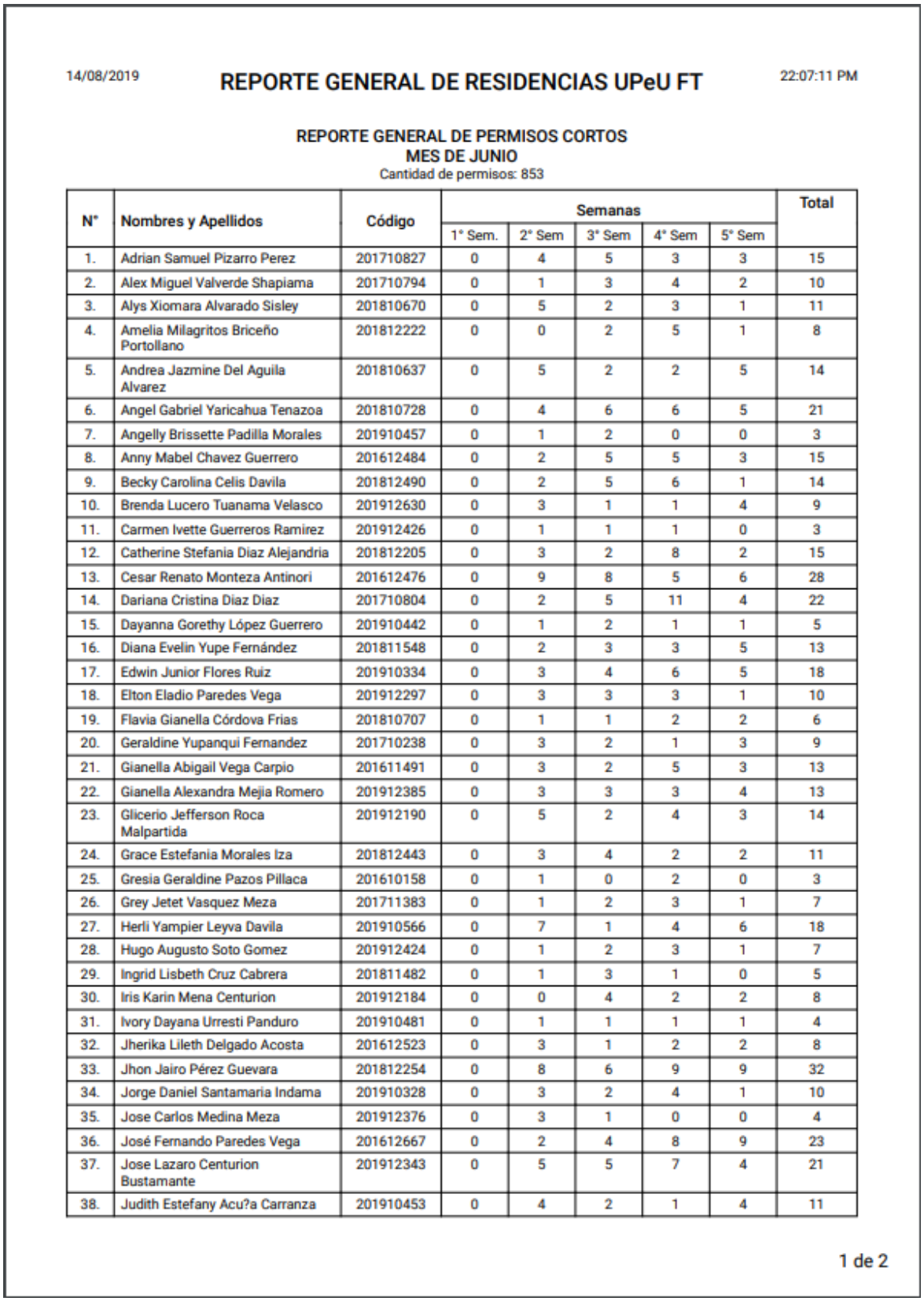


Figura 32. Reporte de permisos largos – Elaboración propia.



14/08/2019 **REPORTE GENERAL DE RESIDENCIAS UPeU FT** 22:07:11 PM

REPORTE GENERAL DE PERMISOS CORTOS
MES DE JUNIO
Cantidad de permisos: 853

N°	Nombres y Apellidos	Código	Semanas					Total
			1° Sem.	2° Sem	3° Sem	4° Sem	5° Sem	
1.	Adrian Samuel Pizarro Perez	201710827	0	4	5	3	3	15
2.	Alex Miguel Valverde Shapiama	201710794	0	1	3	4	2	10
3.	Alys Xiomara Alvarado Sisley	201810670	0	5	2	3	1	11
4.	Amelia Milagritos Briceño Portollano	201812222	0	0	2	5	1	8
5.	Andrea Jazmine Del Aguila Alvarez	201810637	0	5	2	2	5	14
6.	Angel Gabriel Yaricahua Tenazoa	201810728	0	4	6	6	5	21
7.	Angelly Brissette Padilla Morales	201910457	0	1	2	0	0	3
8.	Anny Mabel Chavez Guerrero	201612484	0	2	5	5	3	15
9.	Becky Carolina Celis Davila	201812490	0	2	5	6	1	14
10.	Brenda Lucero Tuanama Velasco	201912630	0	3	1	1	4	9
11.	Carmen Ivette Guerreros Ramirez	201912426	0	1	1	1	0	3
12.	Catherine Stefania Diaz Alejandria	201812205	0	3	2	8	2	15
13.	Cesar Renato Monteza Antinori	201612476	0	9	8	5	6	28
14.	Dariana Cristina Diaz Diaz	201710804	0	2	5	11	4	22
15.	Dayanna Gorethy López Guerrero	201910442	0	1	2	1	1	5
16.	Diana Evelin Yupe Fernández	201811548	0	2	3	3	5	13
17.	Edwin Junior Flores Ruiz	201910334	0	3	4	6	5	18
18.	Elton Eladio Paredes Vega	201912297	0	3	3	3	1	10
19.	Flavia Gianella Córdova Frias	201810707	0	1	1	2	2	6
20.	Geraldine Yupanqui Fernandez	201710238	0	3	2	1	3	9
21.	Gianella Abigail Vega Carpio	201611491	0	3	2	5	3	13
22.	Gianella Alexandra Mejia Romero	201912385	0	3	3	3	4	13
23.	Glicerio Jefferson Roca Malpartida	201912190	0	5	2	4	3	14
24.	Grace Estefania Morales Iza	201812443	0	3	4	2	2	11
25.	Gresia Geraldine Pazos Pillaca	201610158	0	1	0	2	0	3
26.	Grey Jetet Vasquez Meza	201711383	0	1	2	3	1	7
27.	Herli Yampier Leyva Davila	201910566	0	7	1	4	6	18
28.	Hugo Augusto Soto Gomez	201912424	0	1	2	3	1	7
29.	Ingrid Lisbeth Cruz Cabrera	201811482	0	1	3	1	0	5
30.	Iris Karin Mena Centurion	201912184	0	0	4	2	2	8
31.	Ivory Dayana Urresti Panduro	201910481	0	1	1	1	1	4
32.	Jherika Lileth Delgado Acosta	201612523	0	3	1	2	2	8
33.	Jhon Jairo Pérez Guevara	201812254	0	8	6	9	9	32
34.	Jorge Daniel Santamaria Indama	201910328	0	3	2	4	1	10
35.	Jose Carlos Medina Meza	201912376	0	3	1	0	0	4
36.	José Fernando Paredes Vega	201612667	0	2	4	8	9	23
37.	Jose Lazaro Centurion Bustamante	201912343	0	5	5	7	4	21
38.	Judith Estefany Acu?a Carranza	201910453	0	4	2	1	4	11

1 de 2

Figura 33. Reporte de permisos. Vista PDF y/o impresión – Elaboración propia.

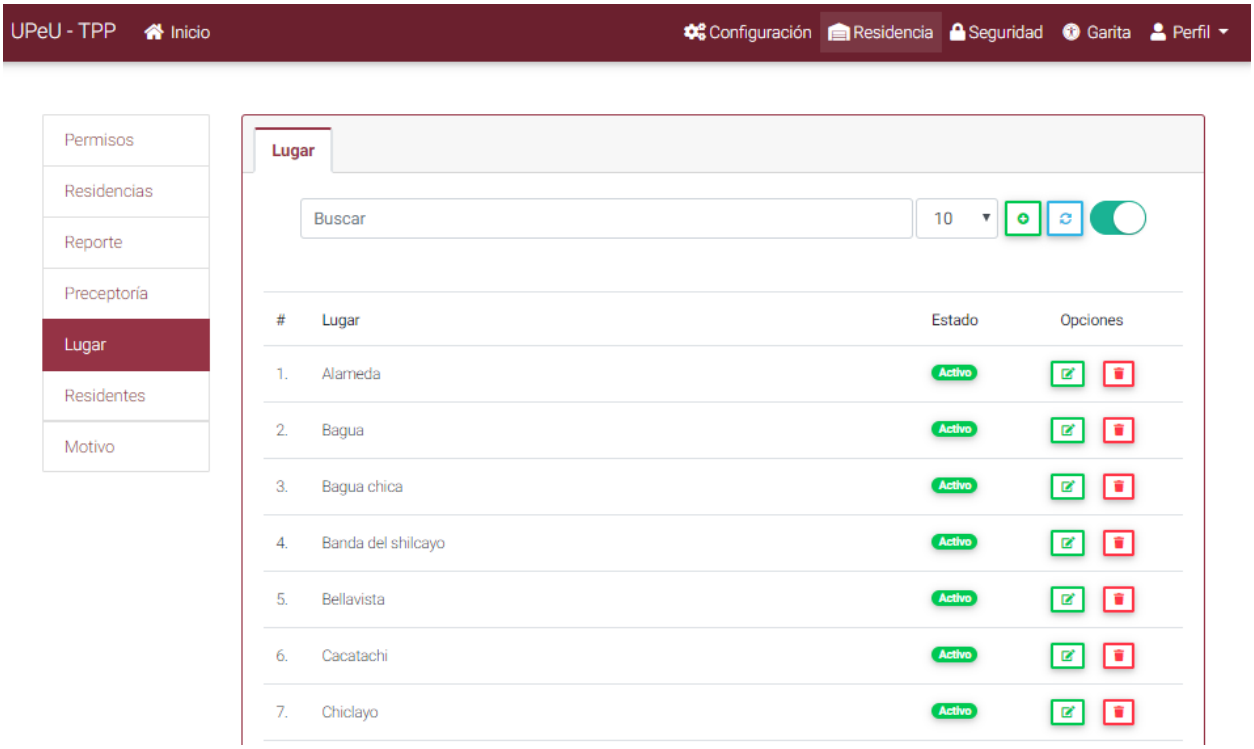


Figura 34. Registro de lugares – Elaboración propia.

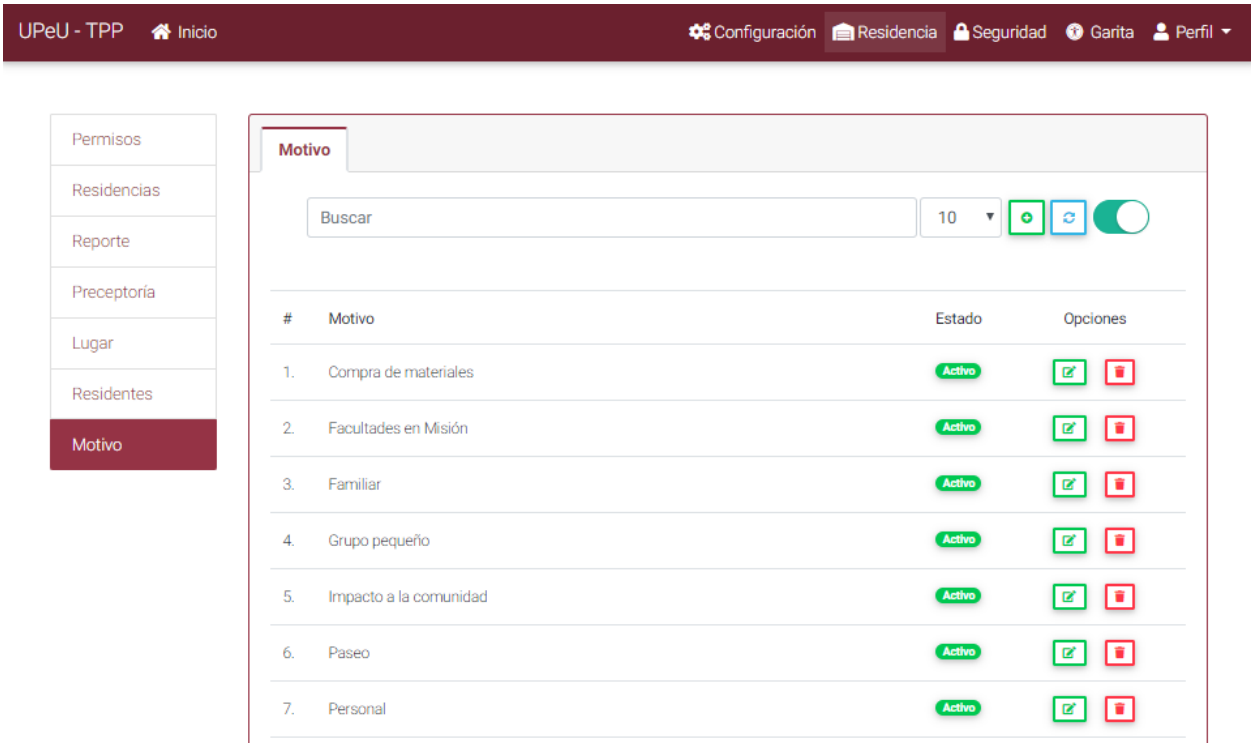


Figura 35. Registro de motivos de salidas – Elaboración propia.

Módulo Garita

En la plataforma web solo se muestra una lista de los estados de los permisos (Ingresó, no ingresó, salió, por salir, y prórroga), ver Figura 36.

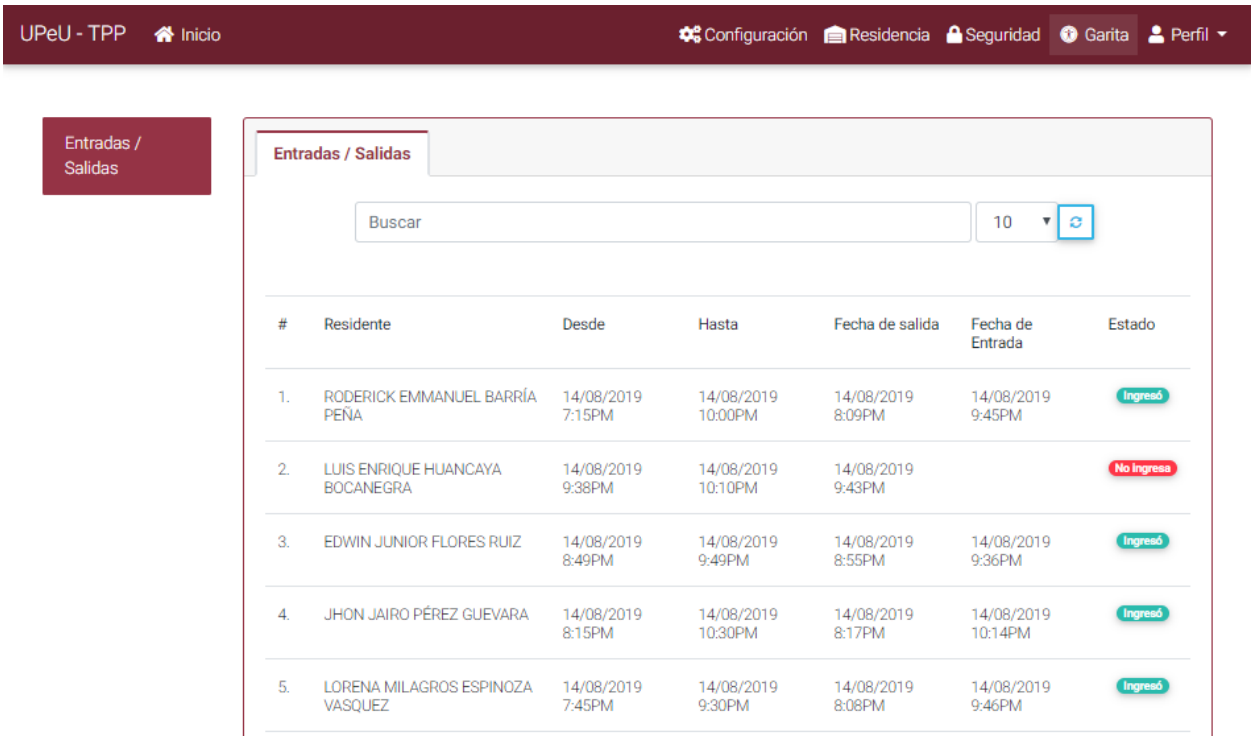


Figura 36. Reporte de estados de permisos – Elaboración propia.

Módulo Seguridad

En este módulo es donde se administran los accesos a las dos plataformas que aborda este proyecto (web y móvil). Aquí se crean los perfiles, usuarios, roles, accesos, módulos del sistema, entre otros.

Cabe mencionar que la renderización de las vistas de los módulos va a depender de los accesos del usuario que se ha autentificado en cada una de las aplicaciones. Dicho de otro modo, las vistas son dinámicas.

A continuación, se muestran las vistas de usuario de este módulo. Ver Figura 37, Figura 38, Figura 39, Figura 40, y Figura 41.

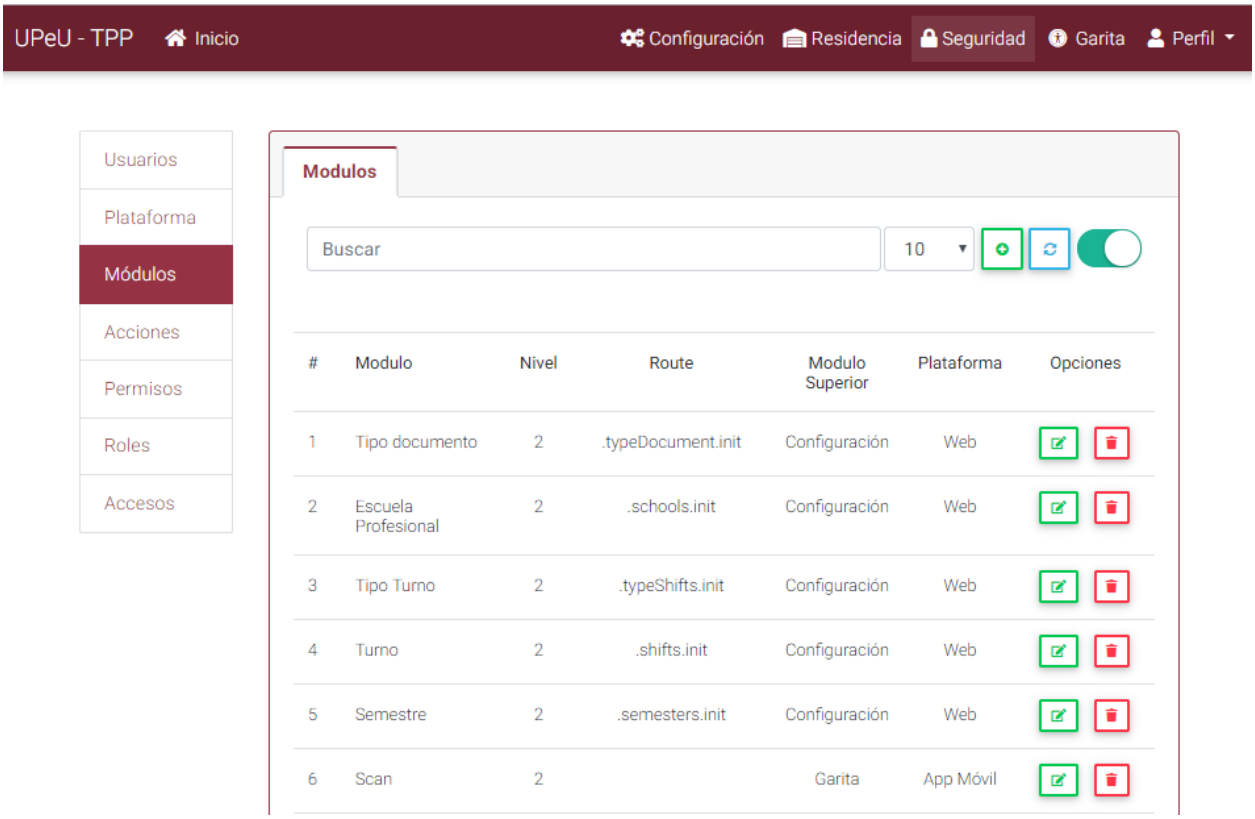


Figura 37. Registro de módulos por tipos de plataforma – Elaboración propia.

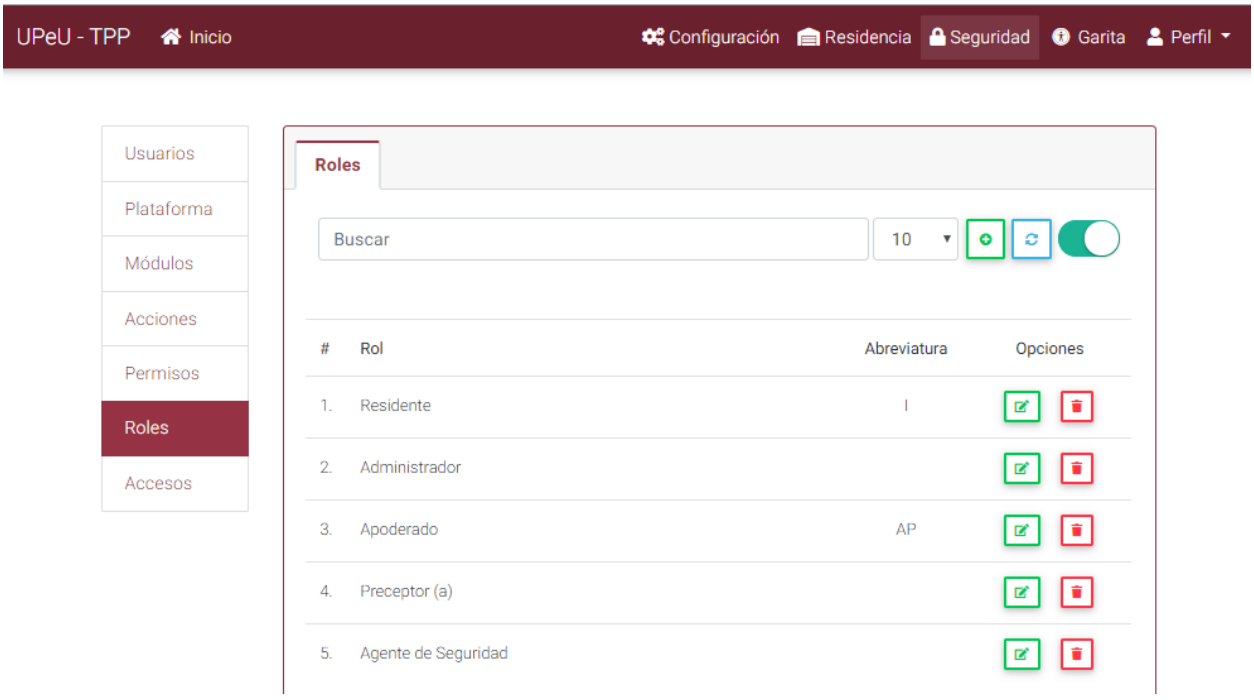


Figura 38. Roles de acceso al sistema – Elaboración propia.

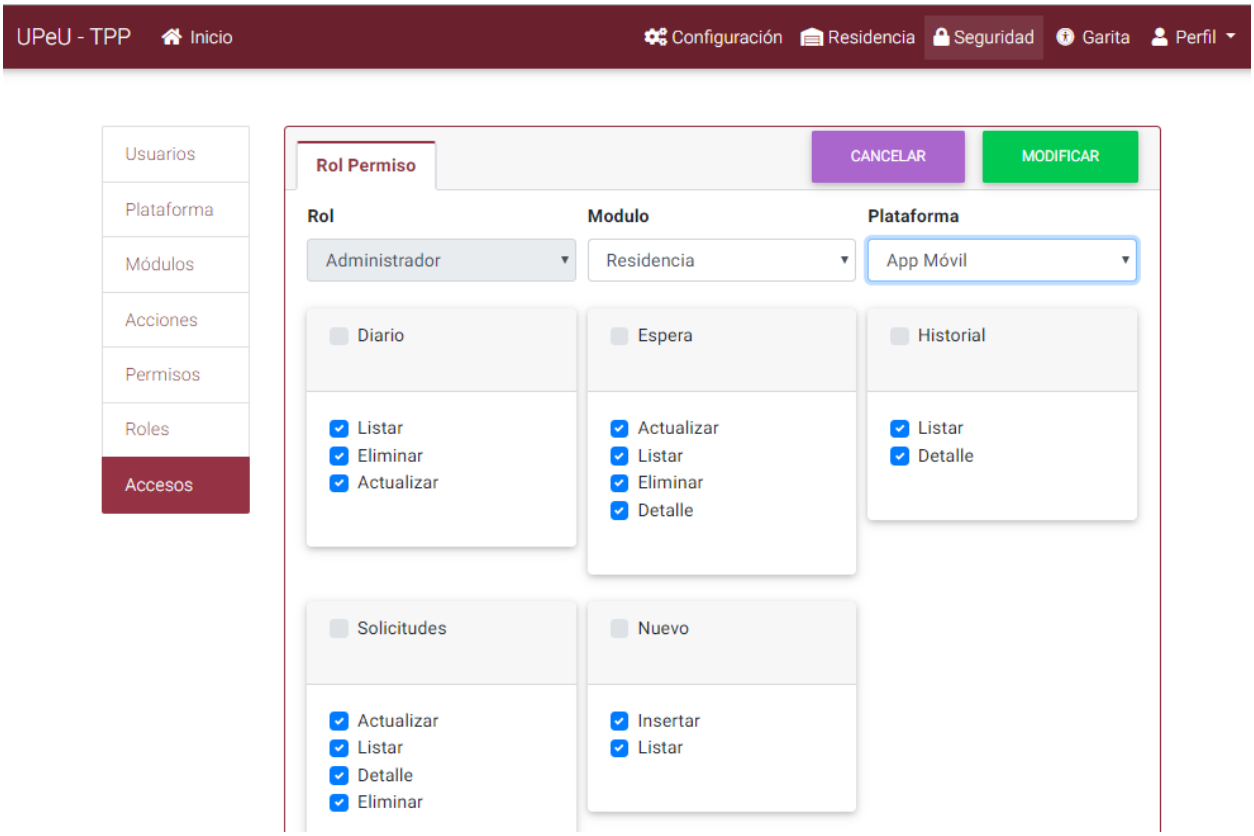


Figura 39. Configuración de accesos al sistema por rol – Elaboración propia.

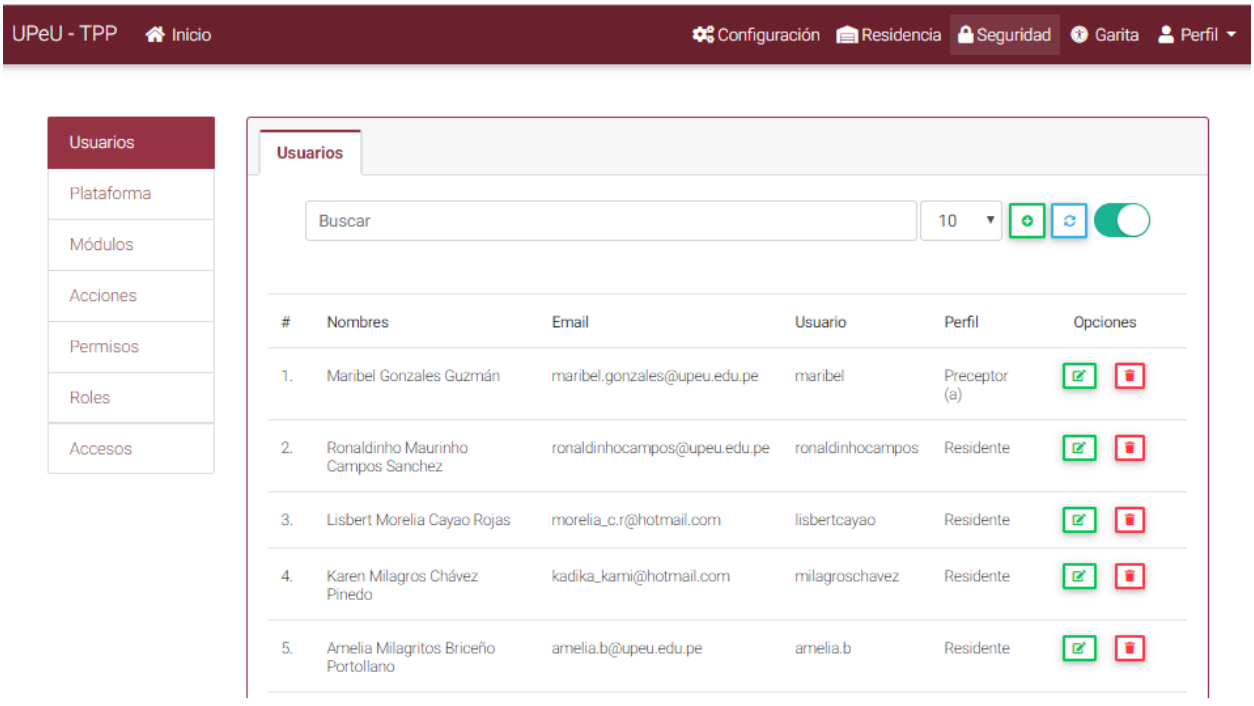


Figura 40. Reporte de usuarios – Elaboración propia.

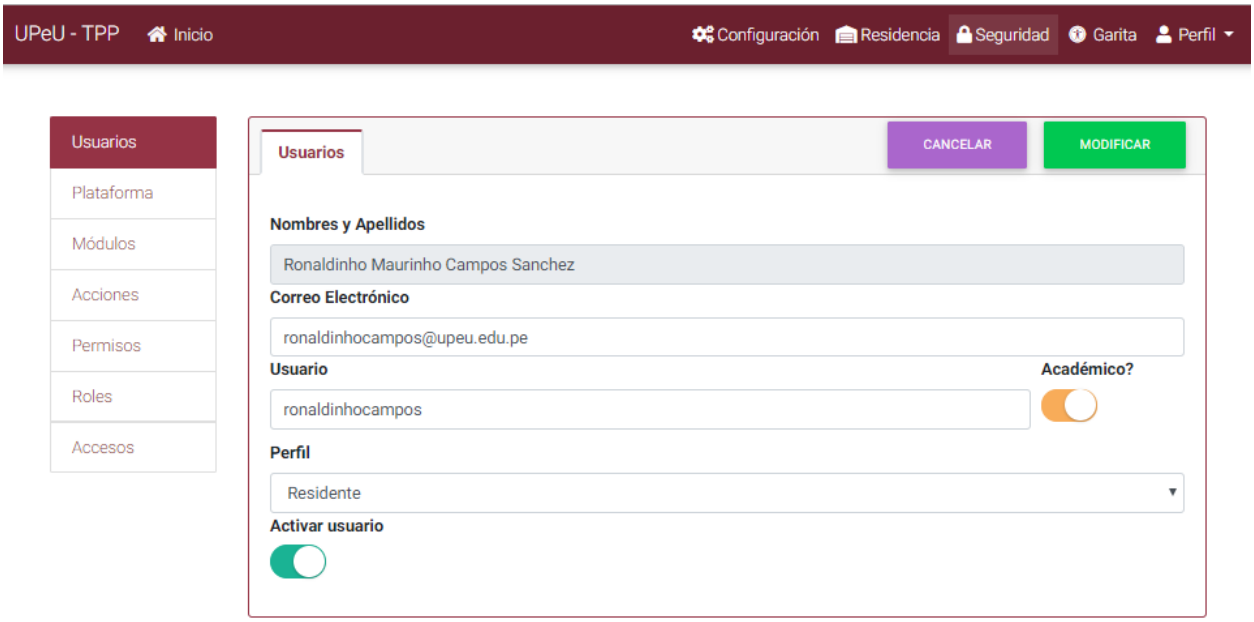


Figura 41. Configuración de usuarios – Elaboración propia.

4.3.2.4.3. Frontend Mobile

En esta sección se aborda a un nivel con mayor detalle sobre cómo utilizamos Ionic Framework para la construcción del aplicativo móvil.

Lo que se creó fue una aplicación hibrida, construida por tecnologías web de código abierto como HTML, CSS y JavaScript. Las aplicaciones híbridas se ejecutan en un navegador de pantalla completa, llamado vista web, que es invisible para el usuario. A través de complementos nativos personalizables, pueden acceder a las funciones nativas de dispositivos móviles específicos (como la cámara o la identificación táctil), sin que el código central esté vinculado a ese dispositivo.

Es importante también mencionar que la comunidad de desarrolladores web es aproximadamente 10 veces más grande que la cantidad de desarrolladores nativos de aplicaciones móviles, según la última encuesta de Stack Overflow.

Ionic permite visualizar cómo van quedando las vistas y funcionalidades a través de un navegador web, simulador o dispositivo móvil. Veamos en un simulador web en la Figura 42.

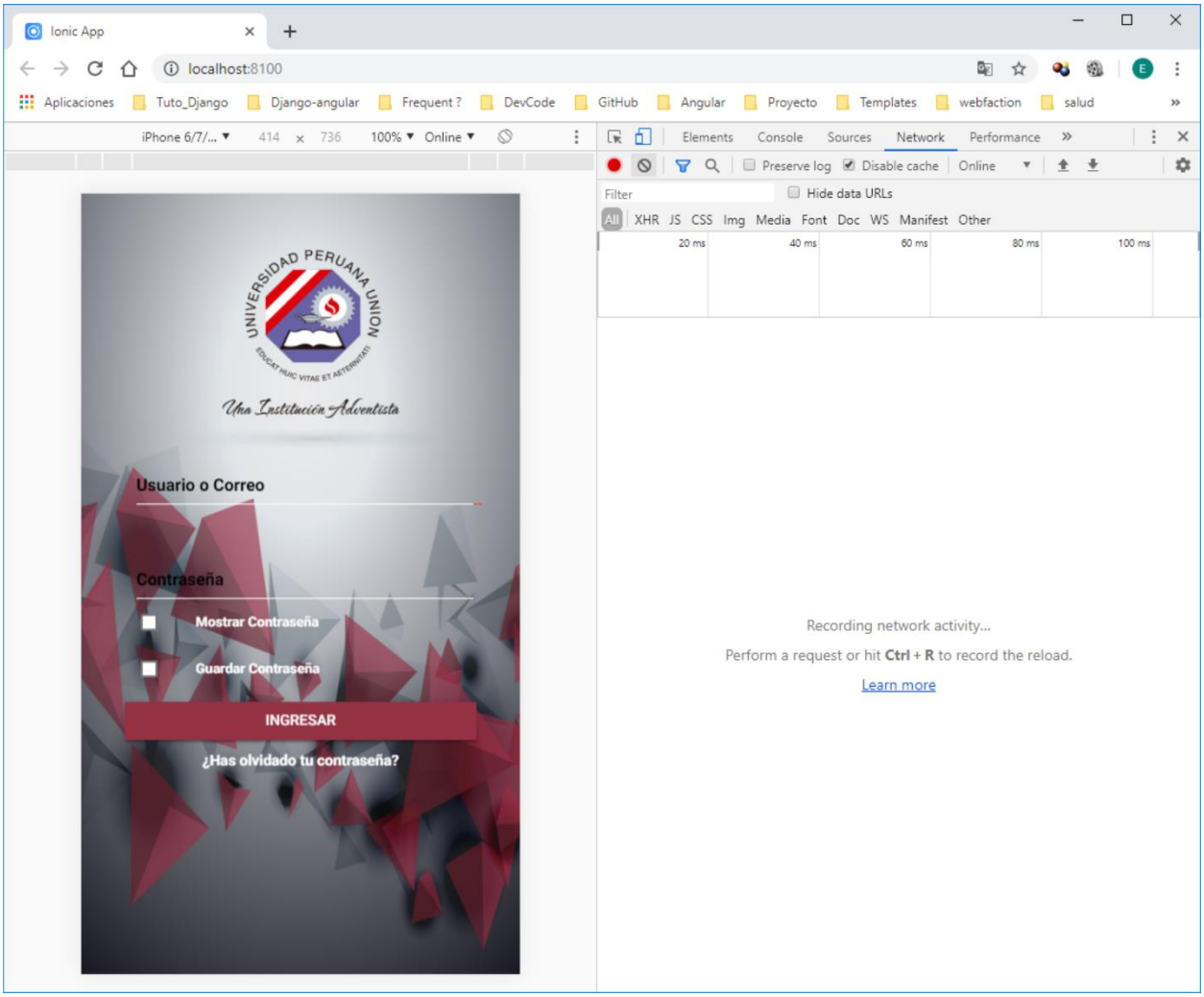


Figura 42. Simulación en tiempo real en un navegador web – Elaboración propia.

Después de compilar y firmar la aplicación para Android y/o IOS se deben subir a las respectivas tiendas de aplicaciones de cada sistema operativo. En este caso, se subió a Google Play Store con la cuenta institucional de la Universidad Peruana Unión con el nombre de “Control Residente”.

Ahora se muestra de forma resumida la secuencia de instrucciones para lograr tener como resultado final la aplicación deseada.

Primeramente, necesitamos configurar el proyecto con la estructura básica con Ionic. En este proyecto utilizamos la versión 3. En la Figura 43 se muestra la estructura del proyecto.

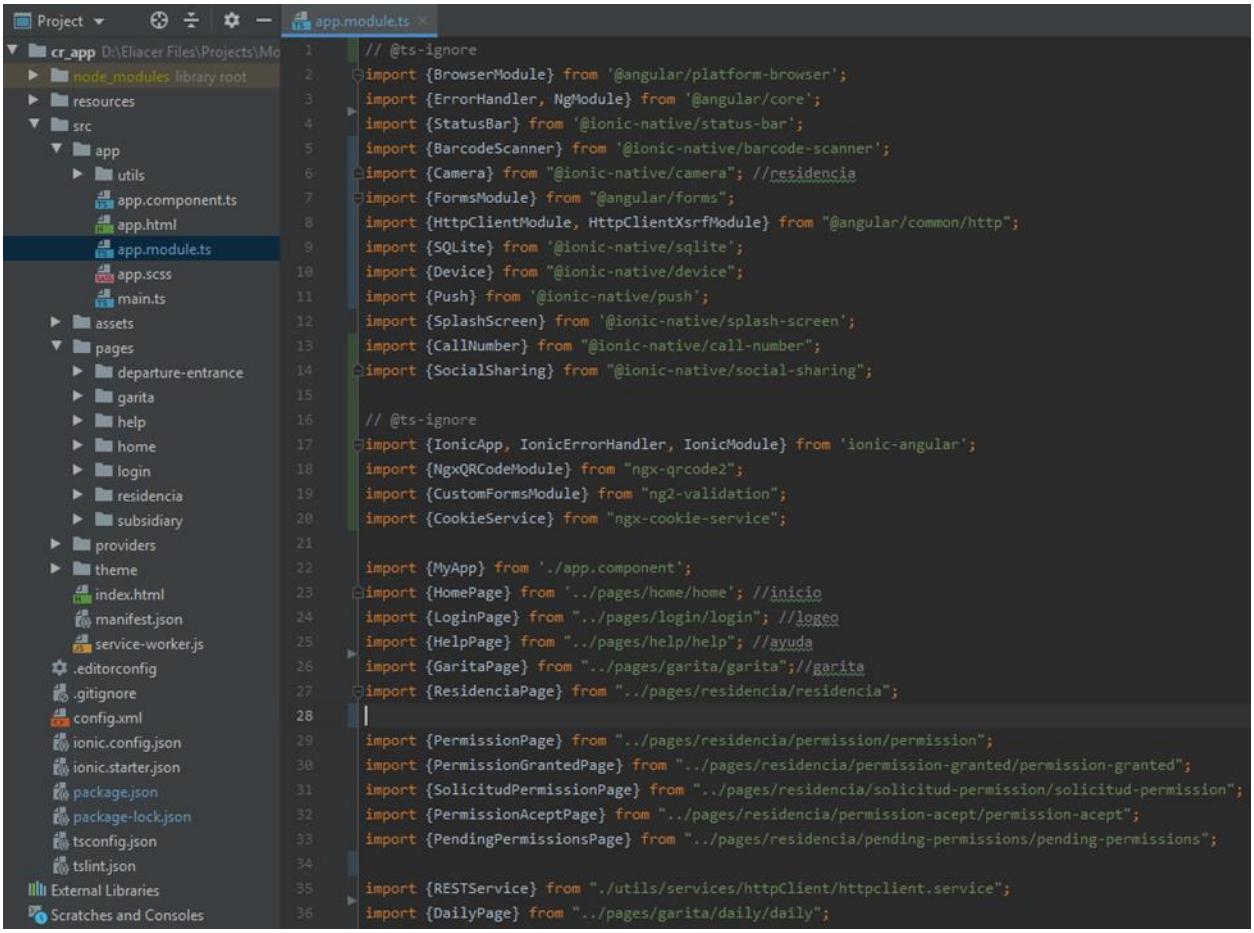


Figura 43. Estructura del proyecto con Ionic 3 – Elaboración propia.

Después de codificar cada uno de los requisitos, por el cliente (por cada sprint), y una vez lista, se subió la aplicación a Play Store con la cuenta de la institución, ver Figura 44.

Figura 44. Aplicación "Control Residente" en Google Play

Finalmente, veremos los resultados de las interfaces de usuario, mostrando cada una de las operaciones que evidencian el cumplimiento de los requisitos solicitados por el cliente.

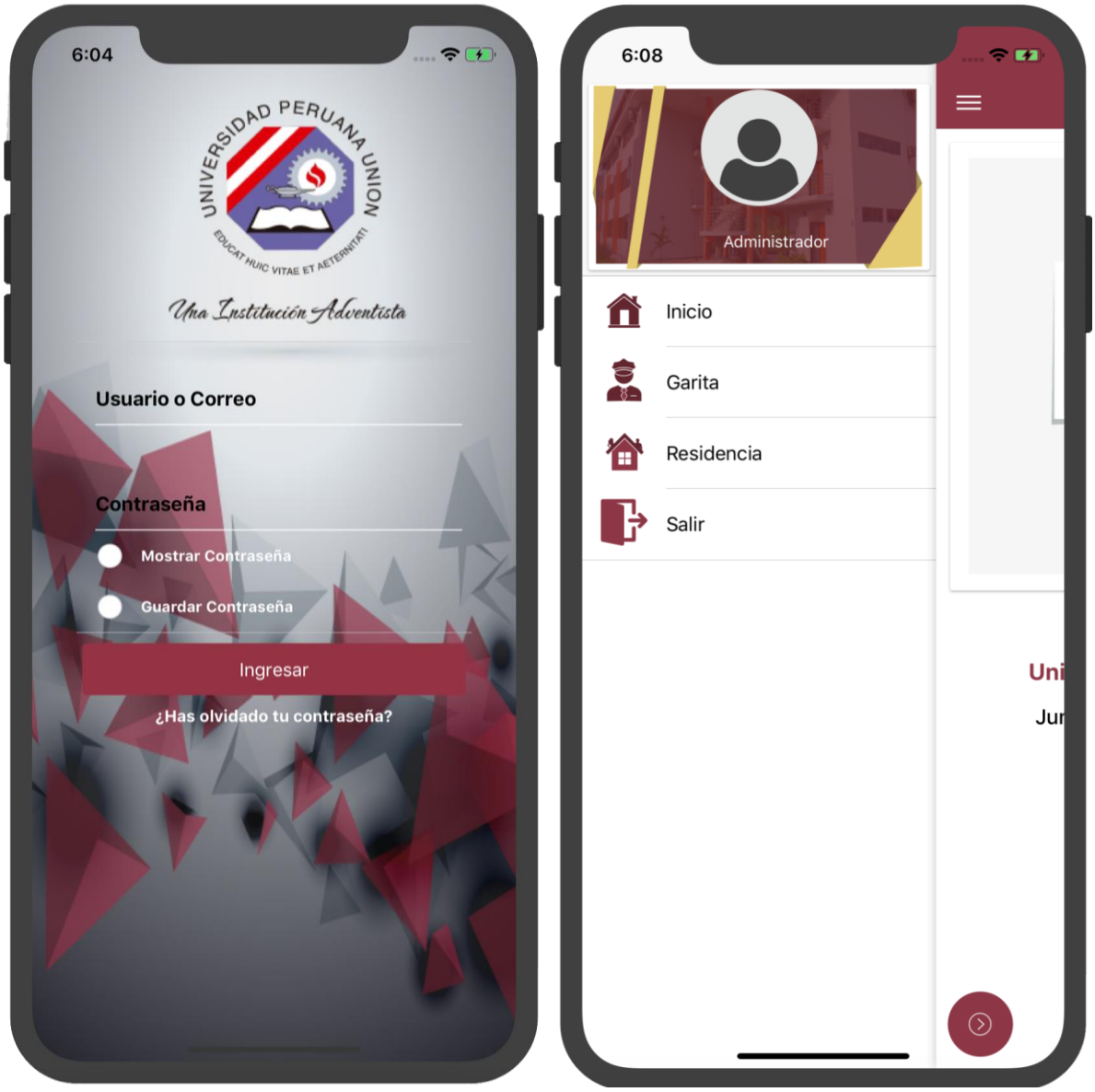


Figura 45. Interfaz de logueo e inicio de la aplicación – Elaboración propia.

En la Figura 45, se muestra que el acceso al sistema se realiza mediante el ingreso de un usuario y contraseña, que, al validar los accesos, muestra el inicio y los accesos de acuerdo al perfil del usuario en sesión.

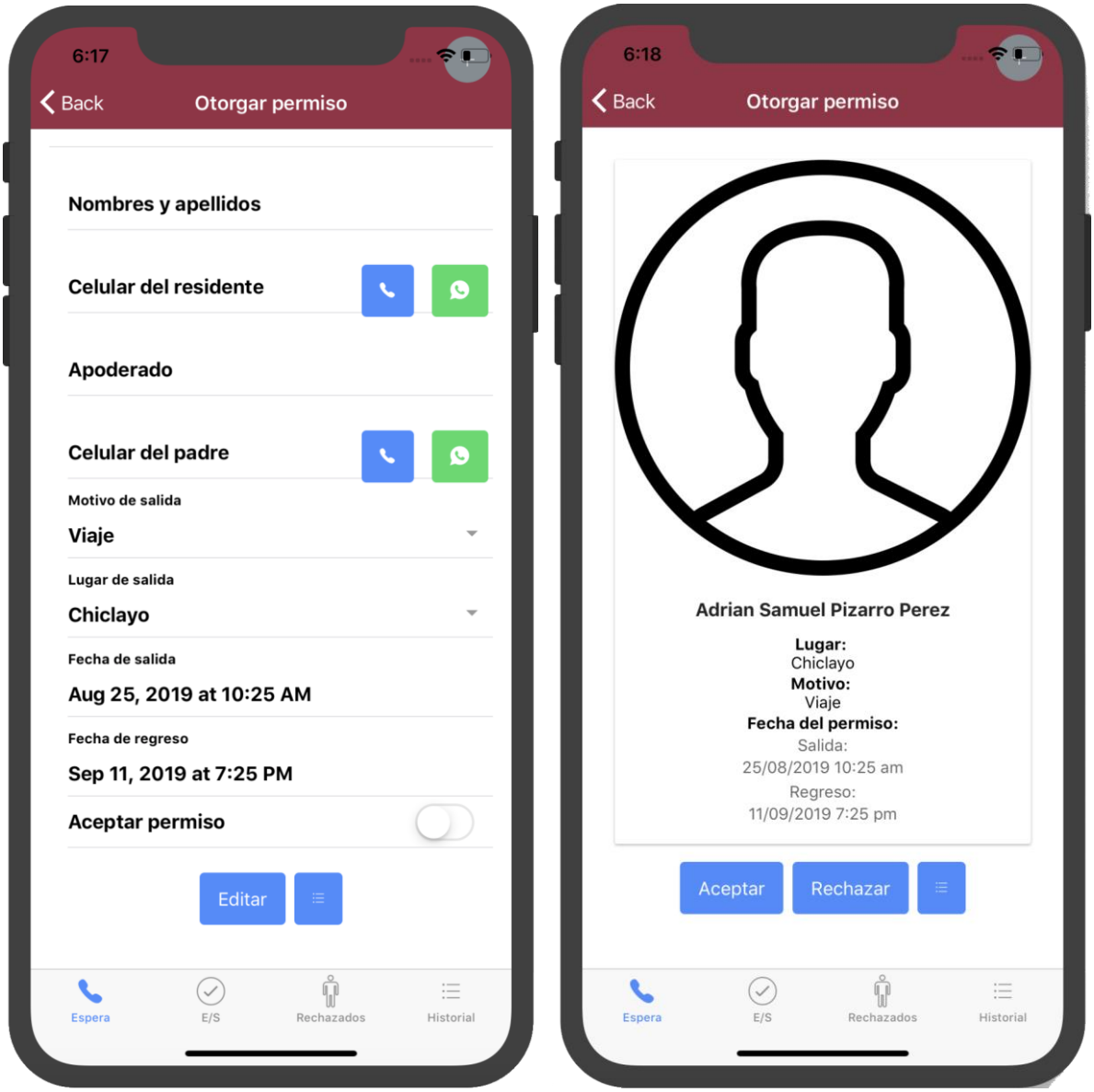


Figura 46. Formulario de registro, y la acción al mismo (aceptar y/o rechazar) – Elaboración propia.

En la Figura 46, se muestra el formulario de registro de permisos donde se ingresa la información básica que se requiere. Luego de ser registrada (o también llamada solicitud), tiene que ser aprobada y/o rechazada de acuerdo a las medidas que consigne el preceptor (a) encargado(a).

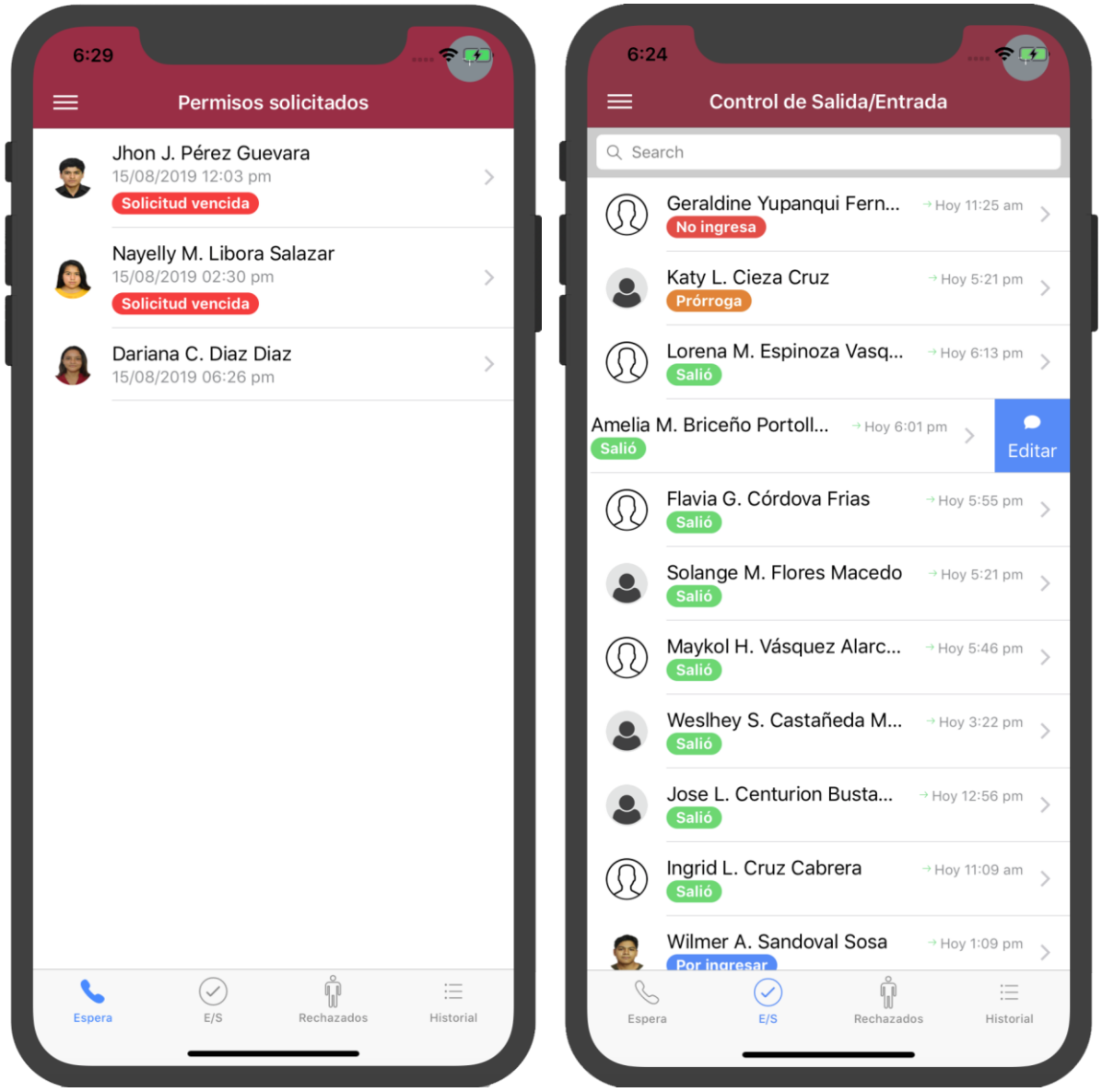


Figura 47. Lista de los permisos solicitados, y estados de los mismos en tiempo real – Elaboración propia.

En la Figura 47, se pueden visualizar todos aquellos permisos solicitados, pendientes de observación de los encargados. Una vez que estos son aprobados, se muestran la opción de E/S, mostrando los siguientes estados: *por salir*, *salió*, *por ingresar*, *prórroga*, *no ingresa*; de acuerdo a los pasos por los cuales pase cada uno.

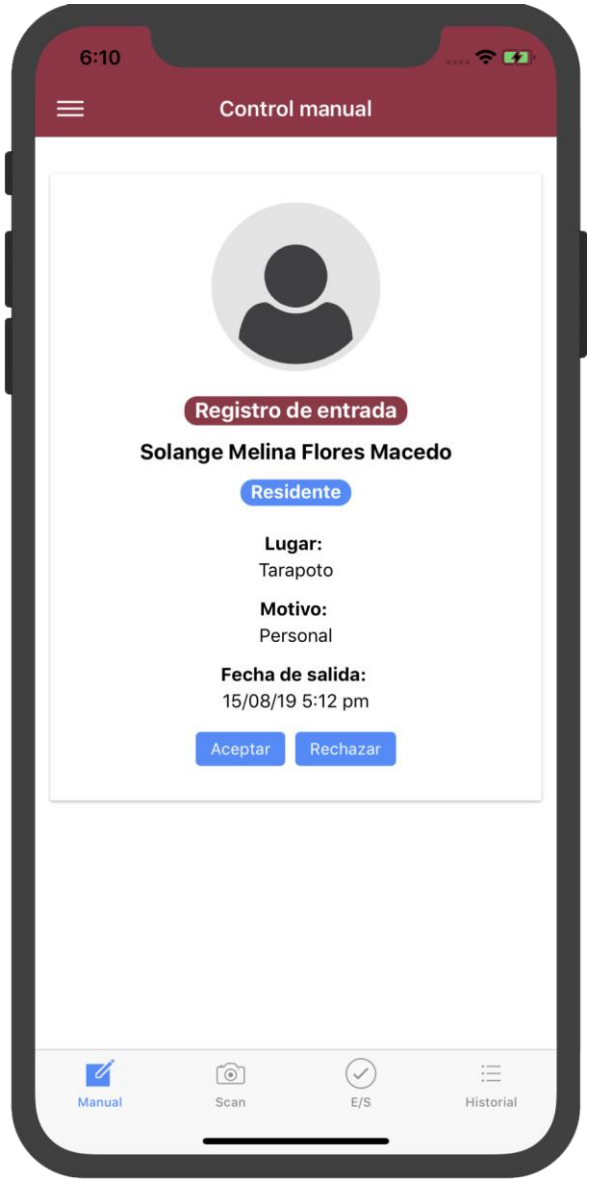


Figura 48. Registro de entrada en garita – Elaboración propia.

En la Figura 48, se muestra la forma de registrar una entrada o una salida respecto de un permiso. Se consideran 2 opciones: la primera es manual, quiere decir que se busca al residente; y la segunda, mediante Scan de código QR o Barras desde la misma cámara del dispositivo.

Figura 49. Historial de permisos – Elaboración propia.

En la Figura 49, se muestra el historial de permisos donde indica el comportamiento histórico de cada residente. En la primera parte de muestra de manera gráfica las cantidades de cada uno: *puntual, prórroga y tardanza.* En la segunda parte se especifica detalladamente los lugares y las fechas que se solicitaron permisos, y el estado de cada uno.

4.3.3. Revisión y Retrospectiva.

El equipo Scrum y las partes interesadas se reúnen al final de cada Sprint para presentar y revisar las historias de usuario completados.

4.3.3.1. Sprint 1.

Tabla 28

Resultados de revisión del primer Sprint – Elaboración propia.

#	Historia de usuario	Validación
1	Registro de residentes	✓
2	Registro y configuración de preceptores	✓
3	Registro de permisos	✓
4	Consultar los estados de los permisos	✓

Revisión de las historias de usuario por parte del dueño del producto y dados por completado como se muestra en la Tabla 28.

4.3.3.2. Retrospectiva del Sprint 1.

Las técnicas utilizadas en la retrospectiva del proyecto son las siguientes:

Plus and Delta: Herramienta para identificar las condiciones a mantener y las condiciones a mejorar para los siguientes Sprints, que permitirán avanzar al equipo.

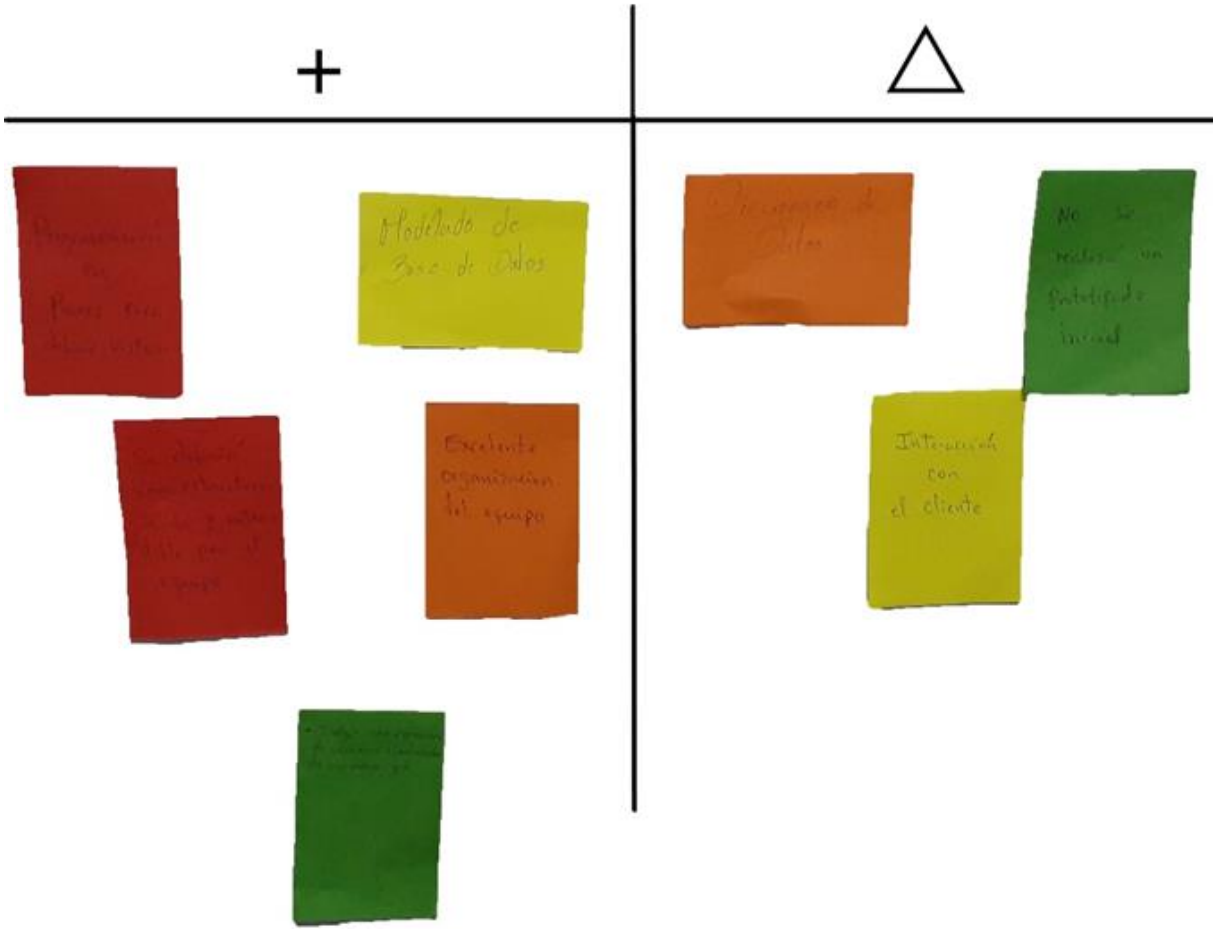


Figura 50. Retrospectiva del primer Sprint – Elaboración Propia.

Tabla 29

Condiciones a mantener del primer Sprint – Elaboración propia.

#	**Condiciones a mantener**
1	Programación en pares.
2	Modelado de base de datos.
3	Se definió una estructura sólida y entendible por el equipo.
4	Excelente organización del equipo.
5	Trabajo con repositorio de versiones o controlador de versiones git.

Las condiciones a mantener mencionados en la Tabla 29 quedarán a cargo de Scrum Master para que se siga aplicando en el equipo de desarrollo.

Tabla 30

Condiciones a mejorar del primer Sprint – Elaboración propia.

#	Condiciones a mejorar
1	Diccionario de datos
2	No se realizó un prototipado
3	Interacción con el cliente

En la Tabla 30, se muestra las condiciones a mejorar identificados del primer Sprint.

Una vez identificados las condiciones a mejorar del primer Sprint aplicamos la regla de Pareto.

Tabla 31

Regla de Pareto del primer Sprint – Elaboración propia.

#	Indicador	Eliacer	Pedro	Ulices	Heber	Jhan
1	Diccionario de datos					
2	No se realizó un prototipado			✓		
3	Interacción con el cliente	✓	✓		✓	✓

Después de obtener el indicador con mayor puntaje acumulado realizamos la técnica de los cinco por qué.

Tabla 32

Tecnica de los cinco por qué del primer Sprint – Elaboración propia.

#	¿Por qué?	Causas
1	¿Por qué no hubo interacción con el cliente?	Porque no disponía de tiempo.
2	¿Por qué no disponía de tiempo el cliente?	Porque tenía actividades cruzadas.

Después de realizar la técnica de los cinco por qué y tener la última causa que originó el problema damos a conocer acciones de contingencia y su responsable para el seguimiento.

- Ayudarle al cliente a planificar sus actividades y fijar una fecha para realizar las interacciones respectivas la cual quedará como responsable el Scrum Master.

4.3.3.3. Sprint 2.

Tabla 33

Resultados de revisión del segundo Sprint – Elaboración propia.

N°	Historia de usuario	Validación
1	Registrar usuarios del sistema	✓
2	Registrar entradas y salidas de los residentes	✓
3	Registrar motivos de salida	✓
4	Registrar lugares	✓

Revisión de las historias de usuario por parte del dueño del producto y dados por completado como se muestra en la Tabla 33.

4.3.3.4. Retrospectiva del Sprint 2.

Las técnicas utilizadas en la retrospectiva del proyecto son las siguientes:

Plus and Delta: Herramienta para identificar las condiciones a mantener y las condiciones a mejorar para los siguientes Sprints, que permitirán avanzar al equipo.

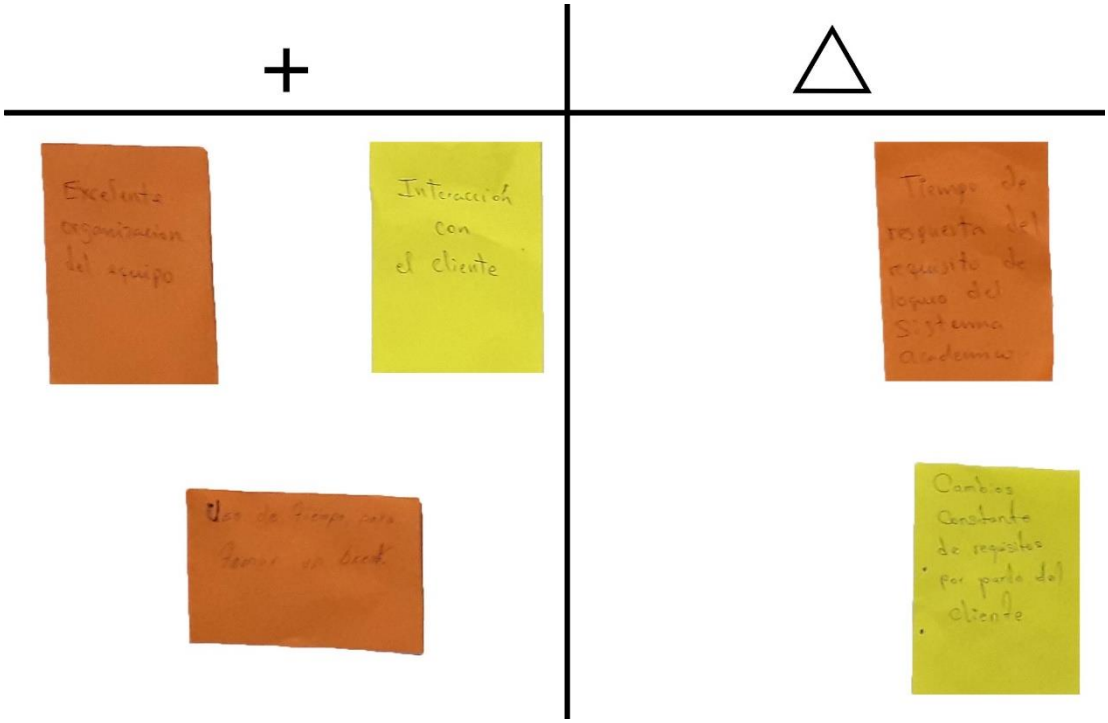


Figura 51. Retrospectiva del segundo Sprint – Elaboración Propia.

Tabla 34

Condiciones a mantener del segundo Sprint – Elaboración propia.

#	Condiciones a mantener
1	Excelente organización del equipo.
2	Interacción con el cliente.
3	Uso de tiempo, para tomar un Break.

Las condiciones a mantener mencionados en la Tabla 34, quedaran a cargo de Scrum Master para que se siga aplicando en el equipo de desarrollo.

Tabla 35

Condiciones a mejorar del segundo Sprint – Elaboración propia.

#	Condiciones a mejorar
1	Tiempo de respuesta del requisito del logueo del sistema académico.
2	Cambios constantes de requisitos por parte del cliente.

En la Tabla 35 se muestra las condiciones a mejorar identificados del primer Sprint.

Una vez identificados las condiciones a mejorar del segundo Sprint aplicamos la regla de Pareto.

Tabla 36

Regla de Pareto del segundo Sprint – Elaboración propia.

#	Indicador	Eliacer	Pedro	Ulices	Heber	Jhan
1	Tiempo de respuesta del requisito del logueo del sistema académico.		✓			✓
2	Cambios constantes de requisitos por parte del cliente.	✓		✓	✓	

Después de obtener el indicador con mayor puntaje acumulado realizamos la técnica de los cinco por qué.

Tabla 37

Tecnica de los cinco por qué del segundo Sprint – Elaboración propia.

#	¿Por qué?	Causas
1	¿Por qué hay cambios constantes de requisitos por parte del cliente?	Porque las necesidades del cliente no fueron claras al inicio del proyecto.
2	¿Por qué las necesidades del cliente no fueron claras?	Mala comunicación del Stakeholder con el dueño del producto.

Después de realizar la técnica de los cinco por qué y tener la última causa que originó el problema damos a conocer acciones de contingencia y su responsable para el seguimiento.

- Ayudar al dueño del producto en la creación de la lista de requisitos del producto la cual quedará bajo la responsabilidad del Scrum Master.

4.3.3.5. Sprint 3.

Tabla 38

Resultados de revisión del tercer Sprint – Elaboración propia.

#	Historia de usuario	Validación
1	Registro de apoderados	✓
2	Consultar permisos de sus apoderandos	✓
3	Notificaciones	✓
4	Enlazar número telefónico a llamadas y WhatsApp	✓
5	Presentar reportes mensuales de permisos	✓
6	Consultar permisos de sus apoderandos	✓

Revisión de las historias de usuario por parte del dueño del producto y dados por completado como se muestra en la Tabla 38.

4.3.3.6. Retrospectiva del Sprint 3.

Las técnicas utilizadas en la retrospectiva del proyecto son las siguientes:

Plus and Delta: Herramienta para identificar las condiciones a mantener y las condiciones a mejorar para los siguientes Sprints, que permitirán avanzar al equipo.

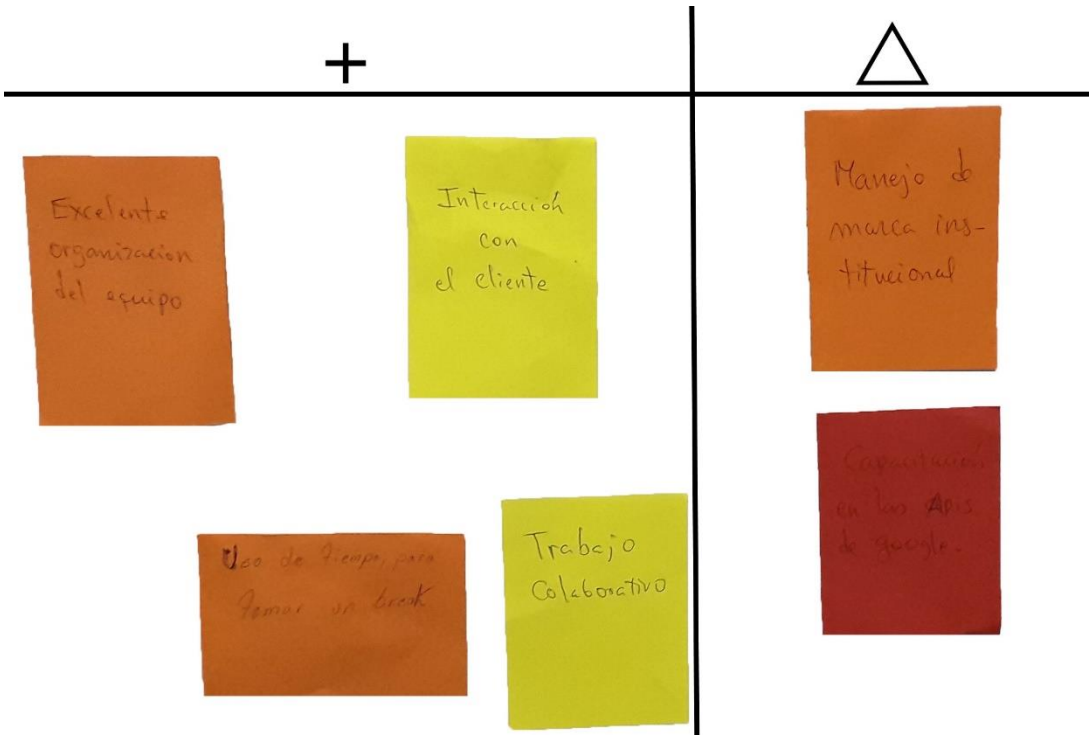


Figura 52. Retrospectiva del tercer Sprint – Elaboración Propia.

Tabla 39

Condiciones a mantener del tercer Sprint – Elaboración propia.

#	Condiciones a mantener
1	Excelente organización del equipo
2	Interacción con el cliente
3	Uso de tiempo, para tomar un Break
4	Trabajo colaborativo

Las condiciones a mantener mencionados en la Tabla 39, quedarán a cargo de Scrum Master para que se siga aplicando en el equipo de desarrollo.

Tabla 40

Condiciones a mejorar del tercer Sprint – Elaboración propia.

#	Condiciones a mejorar
1	Manejo de marca institucional
2	Capacitación en las Apis de Google.

En la Tabla 40, se muestra las condiciones a mejorar identificados del primer Sprint.

Una vez identificados las condiciones a mejorar del tercer Sprint aplicamos la regla de Pareto.

Tabla 41

Regla de Pareto del tercer Sprint – Elaboración propia.

#	Indicador	Eliacer	Pedro	Ulices	Heber	Jhan
1	Manejo de marca institucional					
2	Capacitación en las Apis de Google.	✓	✓	✓	✓	✓

Después de obtener el indicador con mayor puntaje acumulado realizamos la técnica de los cinco por qué.

Tabla 42

Tecnica de los cinco por qué del tercer Sprint – Elaboración propia.

#	¿Por qué?	Causas
1	¿Por qué se necesita capacitación en las Apis de Google?	Para mejorar conocimientos y habilidades del equipo.
2	¿Por qué el equipo necesita mejorar conocimientos y habilidades?	Para simplificar o incluir más contenido en el desarrollo de las aplicaciones.
3	¿Por qué se necesita simplificar o incluir más contenido en el desarrollo de las aplicaciones?	Para mejorar el rendimiento de las aplicaciones con el avance de la tecnología.

Después de realizar la técnica de los cinco por qué y tener la última causa que originó el problema damos a conocer acciones de contingencia y su responsable para el seguimiento.

- Para mejorar el rendimiento de las aplicaciones se necesita asistir a talleres, charlas, eventos, etc. Para obtener conocimientos sobre cosas nuevas o ya conocidas y como responsable será el Scrum Master.

4.3.4. Lanzamiento

Para realizar el respectivo despliegue de las aplicaciones se tuvo que conversar con el área de Redes DIGETI UPeU de la sede principal. Tras varias reuniones se preparó un servidor de “acuerdo a los requerimientos propios del proyecto (lenguaje de programación, base de datos, entre otros).

La dinámica del despliegue de la primera versión involucró los siguientes pasos:

- Acceder al servidor mediante el cliente SSH PuTTY.
- Clonar el proyecto desde el repositorio de GitHub en una carpeta específica.
- Configurar el proyecto con el gestor de base de datos PostgreSQL.
- Configurar el servidor web Gunicorn para correr aplicaciones en Python.
- Configurar Supervisord para que la aplicación pueda ser monitorizada correctamente.
- Y listo.

Respecto a las siguientes adiciones de entregables, fruto de las demás iteraciones, se realizaron los siguientes pasos (ver Figura 53):

- Acceder al servidor mediante el cliente SSH PuTTY.
- Hacer un Pull Request del repositorio de GitHub.
- Reiniciar Supervisord por consola.
- Y Listo.

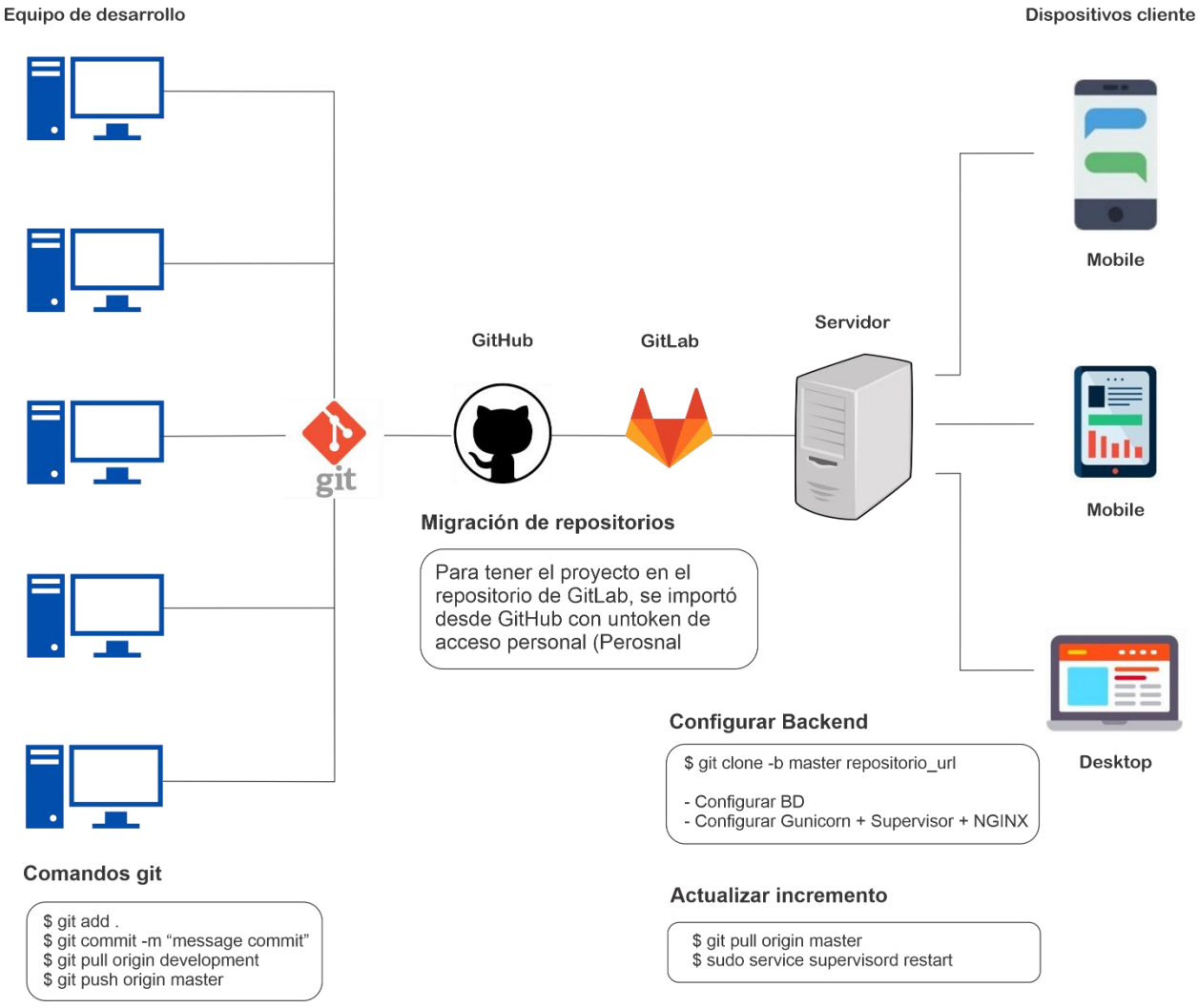


Figura 53. Procedimiento general del proyecto – Elaboración propia.

4.4. Valoración de resultados

4.4.1. Evaluar postest

En este apartado se explica la ejecución del instrumento de postest, realizado con el fin de conocer el impacto de la solución tecnológica en el problema planteado.

Capítulo 5

Resultados y discusión

En esta sección se detalla todos los resultados obtenidos de la presente investigación, de acuerdo a los objetivos definidos en este proyecto.

5.1.1. Análisis descriptivo de la población

En la presente investigación la población estuvo conformado por 4 preceptores (2 de la residencia de mujeres y 2 de varones). En la Tabla 43, se presenta los resultados del Pre y Post respecto a la dimensión tiempo.

Tabla 43

Resultados pre y postest en la dimensión "Tiempo" (segundos) – Elaboración propia.

# Personas	Tiempo para atender una solicitud		Tiempo para realizar un permiso		Tiempo para realizar un reporte	
	Pre	**Post**	**Pre**	**Post**	**Pre**	**Post**
Persona 1	180	60	180	45	1200	35
Persona 2	120	30	120	30	1800	20
Persona 3	120	45	90	30	900	15
Persona 4	180	90	300	60	600	25

En la dimensión gestión de permisos cortos y largos se tomó datos comparativos entre 2018-2 (Pre) y 2019-1 (Post). Los resultados del Pre se obtuvieron por conteo de registros manuales, mientras que los resultados del Post se obtuvieron de los reportes de la aplicación, ver Tabla *44*.

Tabla 44

Resultados pre y postest en la dimensión "Gestion de Permisos cortos y largos" – Elaboración propia.

# Meses	Números de permisos cortos		Números de permisos largos	
	Pre	**Post**	**Pre**	**Post**
Mes 1	455	764	55	159
Mes 2	538	928	121	183
Mes 3	623	853	98	178

5.1.2. Análisis de Hipótesis

Al analizar los resultados obtenidos entre el pre y postest, primero se obtuvo las medias de cada una de las pruebas.

Tabla 45

Medias y desviaciones típicas de las dimensiones en el pre y postest – Elaboración propia.

#	**Dimensiones**	**N**	**Pretest**	**Postest**
			Me ± DE	**Me ± DE**
DM1	Tiempo para atender una solicitud	4	150 ± 34.64	56.25 ± 25.62
DM2	Tiempo para realizar un permiso	4	172.50 ± 92.87	41.25 ± 14.36
DM3	Tiempo para realizar un reporte	4	1125 ± 512.35	23.75 ± 8.54
DM4	Números de permisos cortos	3	538.67 ± 84	848.33 ± 82.10
DM5	Números de permisos largos	3	91.33 ± 33.50	173.33 ± 12.66

Como se aprecia en tabla anterior, las medias de ambas pruebas no son muy cercanas, es decir, son relativamente diferentes. En el caso de las dimensiones DM1, DM2, DM3 (dimensión tiempo), las medias del postest disminuyen. Asimismo, se observa que en las dimensiones DM4, DM5 (dimensión número de permisos) aumentan.

Para establecer si existen diferencias significativas entre las medias de ambas pruebas se estableció:

μ_1: Es la media del pretest

μ_2: Es la media del postest

Planteamos las siguientes hipótesis:

H_0: Las medias son iguales, no hay diferencia significativa entre ambos al revaluar el postest.

H_1: Las medias son diferentes, hay diferencia significativa entre ambos al revaluar el postest.

Es decir:

$H_0: \mu_1 = \mu_2$

$H_1: \mu_1 \neq \mu_2$

Con un ensayo bilateral de nivel de significación de $\alpha = 0.05$ y con 3 grados de libertad, se aplicó la prueba t – Student para muestras independientes.

Tabla 46

Prueba t a las dimensiones específicas de la investigación – Elaboración propia.

Dimensiones		**t**	**gl**	**p**	**IC**
DM1	**Pre - Post**	9.93	3	0.002	[63.72, 123.78]
DM2	**Pre - Post**	3.32	3	0.045	[5.89, 256.61]
DM3	**Pre - Post**	4.30	3	0.023	[285.49, 1917.01]
DM4	**Pre - Post**	-6.70	2	0.022	[-508.40, -110.93]
DM5	**Pre - Post**	-6.74	2	0.021	[-134.34, -29.66]

Respecto al tiempo de atender una solicitud con t = 9.93 y p = 0.002 < α, se acepta la hipótesis alterna, es decir las medias son significativas. Lo cual indica que hay diferencia significativa en el tiempo para atender una solicitud entre el pre y el postest.

Respecto al tiempo para realizar una solicitud con $t = 3.32$ y $p = 0.045 < \alpha$, se acepta la hipótesis alterna, es decir las medias son significativas. Lo cual indica que hay diferencia significativa en el tiempo para realizar una solicitud entre pre y postest.

Respecto al tiempo para realizar un reporte con $t = 4.30$ y $p = 0.023 < \alpha$, se acepta la hipótesis alterna, es decir las medias son significativas. Lo cual indica que hay diferencia significativa en el tiempo para realizar un reporte entre pre y postest.

Respecto al número de permisos cortos con $t = -6.70$ y $p = 0.022 < \alpha$, se acepta la hipótesis alterna, es decir las medias son significativas. Lo cual indica que hay diferencia significativa en el número de permisos cortos entre pre y postest.

Respecto al número de permisos largos con $t = --6.74$ y $p = 0.021 < \alpha$, se acepta la hipótesis alterna, es decir las medias son significativas. Lo cual indica que hay diferencia significativa en el número de permisos largos entre pre y postest.

5.1.3. Análisis estadístico descriptivo

En esta sección se detalla a nivel descriptivo los resultados de los instrumentos pre y postest.

Tabla 47

Pregunta N° 1 - Me costaba / cuesta trabajo organizar la información – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	2
Casi Nunca	0	2
A veces	0	0
Casi siempre	3	0
Siempre	1	0

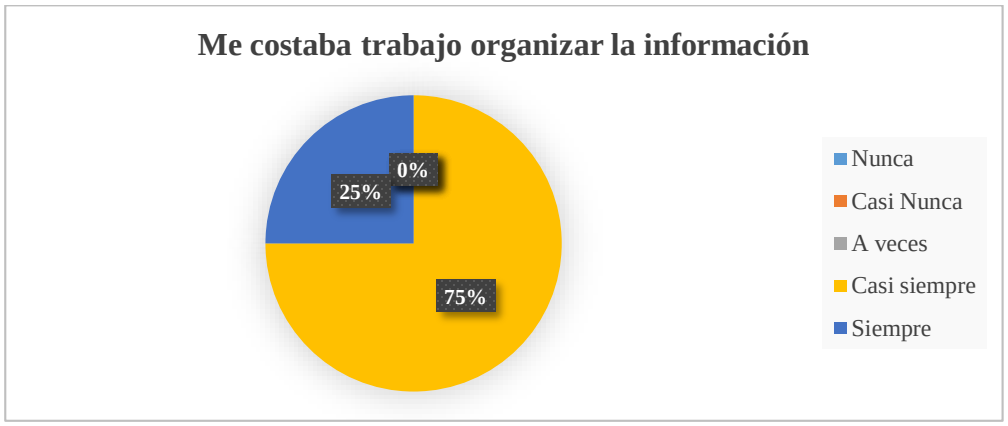


Gráfico 2. Análisis descriptivo - Pregunta 1 (pretest) – Elaboración propia.

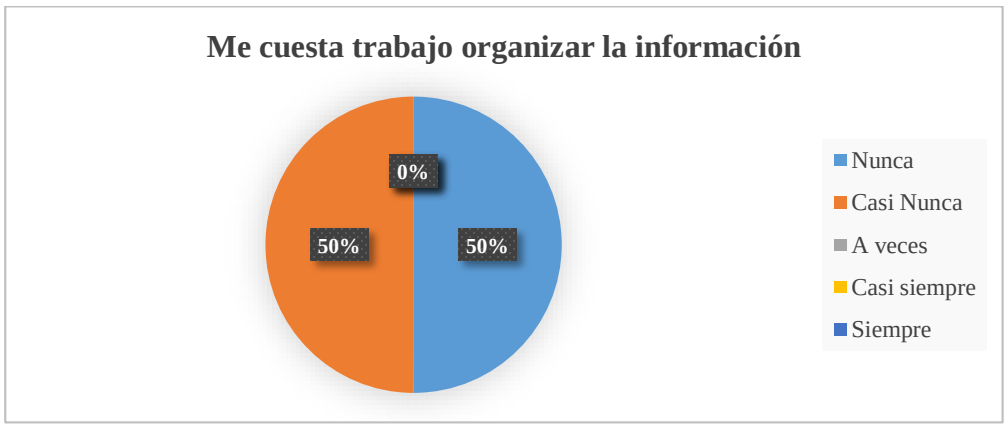


Gráfico 3. Análisis descriptivo - Pregunta 1 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 25% consideró que siempre les costaba trabajo organizar la información, mientras que el 75% lo considera casi siempre.

- **Postest:** Del total de preceptores encuestados, el 50% mencionan que casi nunca le cuesta trabajo organizar la información, y el otro 50% como nunca.

Tabla 48

Pregunta N° 2 - Me era / es fácil realizar el seguimiento de cada permiso – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	0
Casi Nunca	3	0
A veces	1	0
Casi siempre	0	1
Siempre	0	3

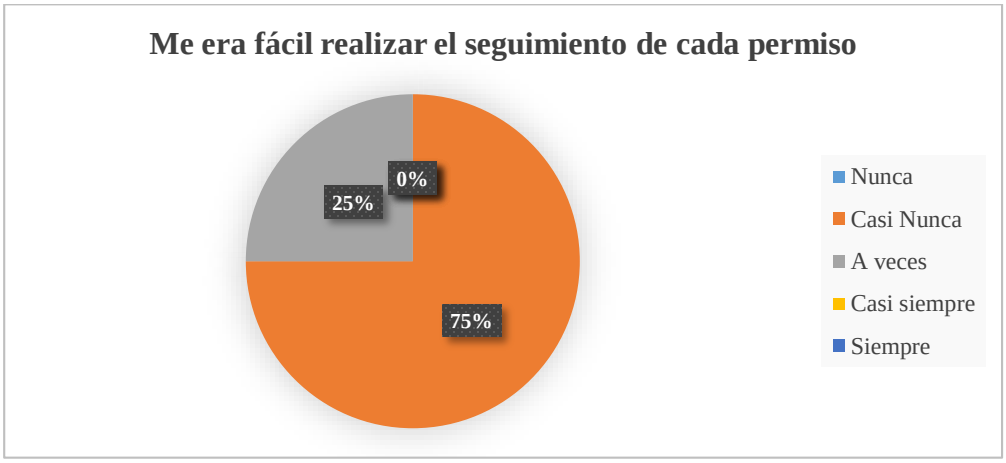


Gráfico 4. Análisis descriptivo - Pregunta 2 (pretest) – Elaboración propia.

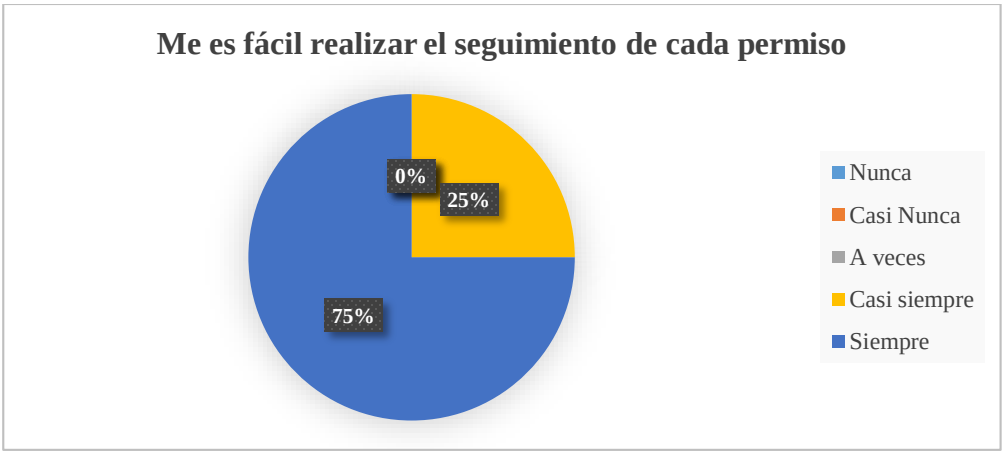


Gráfico 5. Análisis descriptivo - Pregunta 2 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 75% consideró que casi nunca era facil realizar el seguimiento de cada permiso, mientras que el 25% lo consideraba a veces.

- **Postest:** Del total de preceptores encuestados, el 75% mencionan que siempre es facil realizar el seguimiento de cada permiso, mientras que el 25% como casi siempre.

Tabla 49

Pregunta N° 3 - Utilizaba / utilizo material físico para generar los permisos – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	1
Casi Nunca	0	2
A veces	0	1
Casi siempre	0	0
Siempre	4	0

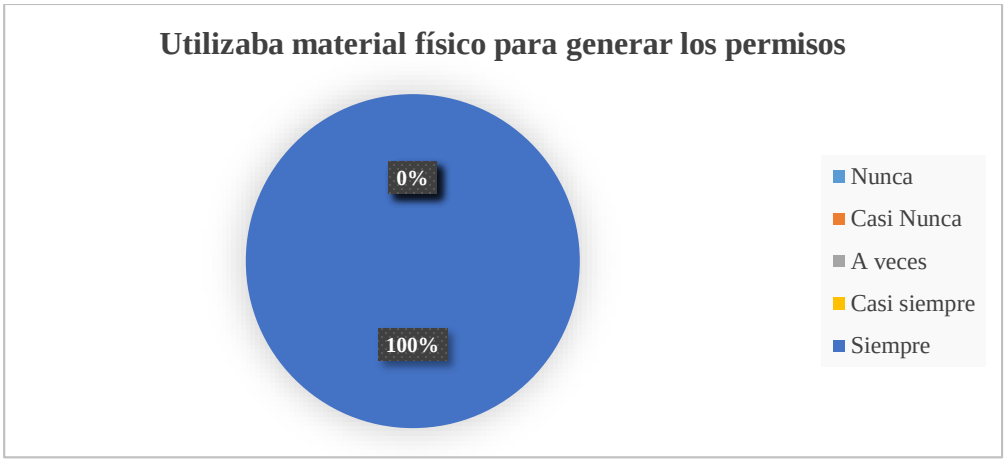


Gráfico 6. Análisis descriptivo - Pregunta 3 (pretest) – Elaboración propia.

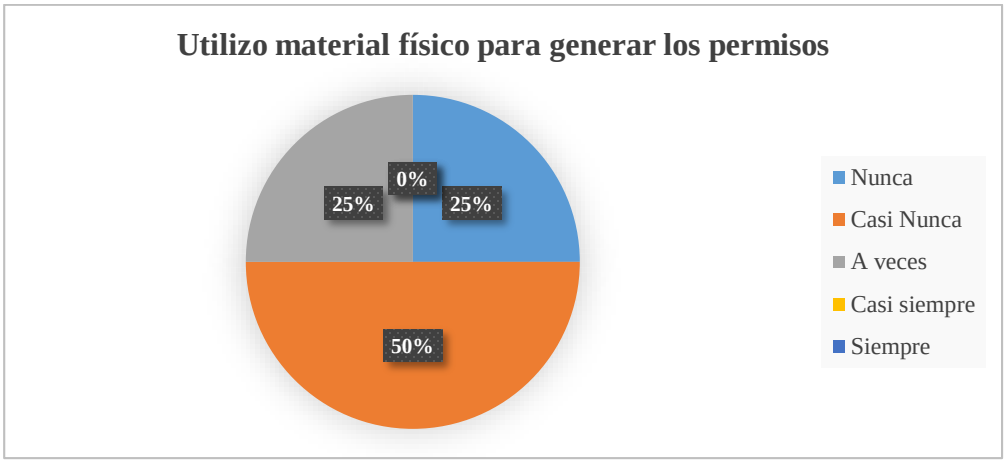


Gráfico 7. Análisis descriptivo - Pregunta 3 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 100% consideró que siempre utilizaba material físico para generar los permisos.

- **Postest:** Del total de preceptores encuestados, el 50% mencionan que casi nunca utiliza material físico para generar los permisos, mientras que un 25% a veces, y un 25% nunca.

Tabla 50

Pregunta N° 4 - Daba / doy tiempo extra para realizar los reportes mensuales y/o semanales – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	1
Casi Nunca	0	2
A veces	0	1
Casi siempre	0	0
Siempre	4	0

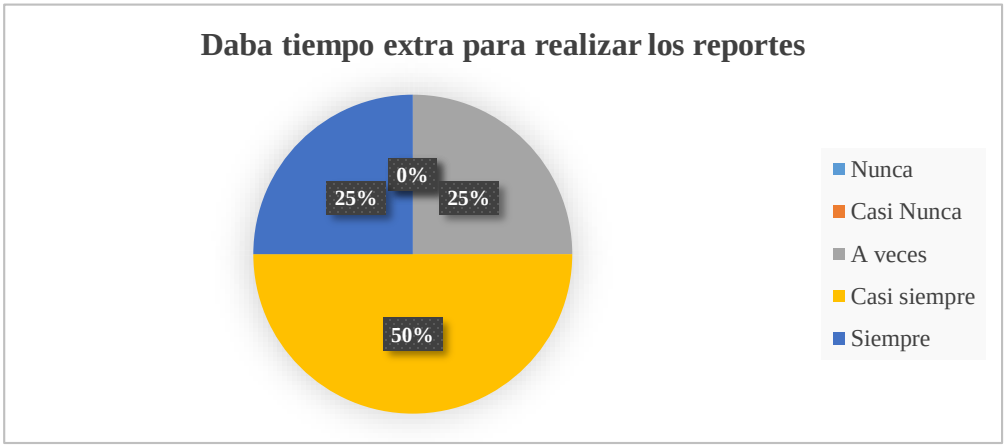


Gráfico 8. Análisis descriptivo - Pregunta 4 (pretest) – Elaboración propia.

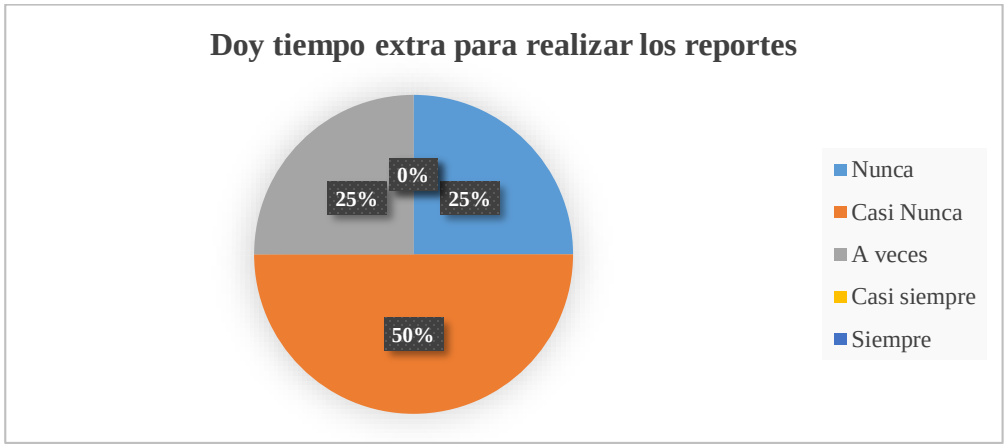


Gráfico 9. Análisis descriptivo - Pregunta 4 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 25% consideró que siempre daba tiempo extra para realizar los permisos, un 50% casi siempre y un 25% a veces.

- **Postest:** Del total de preceptores encuestados, el 50% mencionan que casi nunca da tiempo extra para realizar los permisos, mientras que un 25% a veces, y un 25% nunca.

Tabla 51

Pregunta N° 5 - Existía / existe algún riesgo de que la información sea vulnerada, o sufra algún tipo de pérdida – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	2
Casi Nunca	0	2
A veces	2	0
Casi siempre	2	0
Siempre	0	0

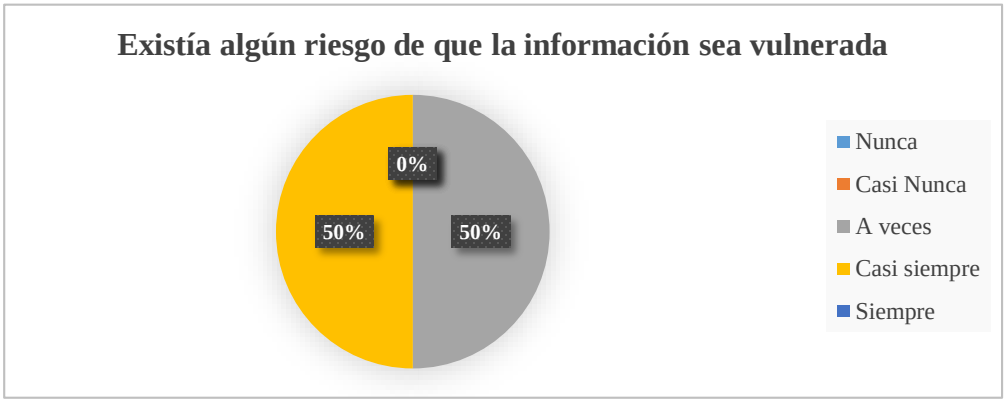


Gráfico 10. Análisis descriptivo - Pregunta 5 (pretest) – Elaboración propia.

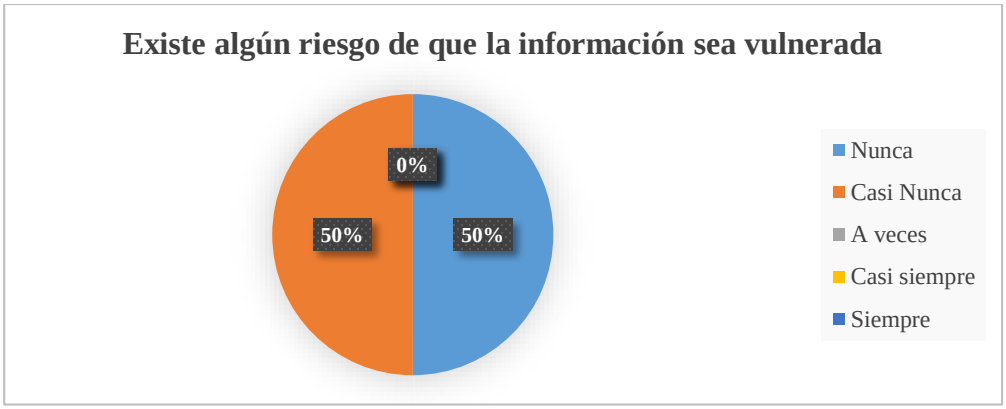


Gráfico 11. Análisis descriptivo - Pregunta 5 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 50% consideró que casi siempre existía algún riesgo de que la información sea vulnerada, mientras que un 50%, a veces.

- **Postest:** Del total de preceptores encuestados, el 50% consideran que nunca existe algún riesgo de que la información sea vulnerada, mientras que un 50%, casi nunca.

Tabla 52

Pregunta N° 6 - Si no me encontraba en la residencia, me costaba / cuesta atender una solicitud de permiso – Elaboración propia.

Calificación	Pretest	Postest
Nunca	0	2
Casi Nunca	0	2
A veces	1	0
Casi siempre	1	0
Siempre	2	0

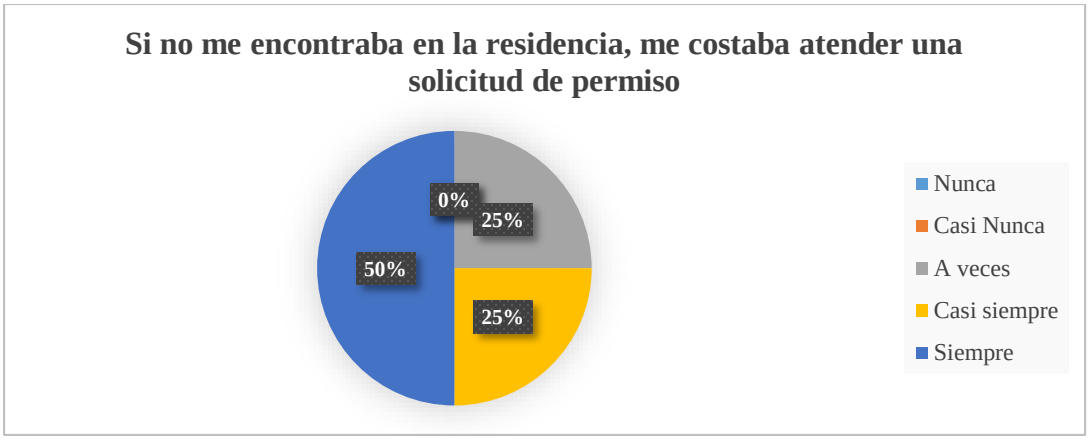


Gráfico 12. Análisis descriptivo – Pregunta 6 (pretest) – Elaboración propia.

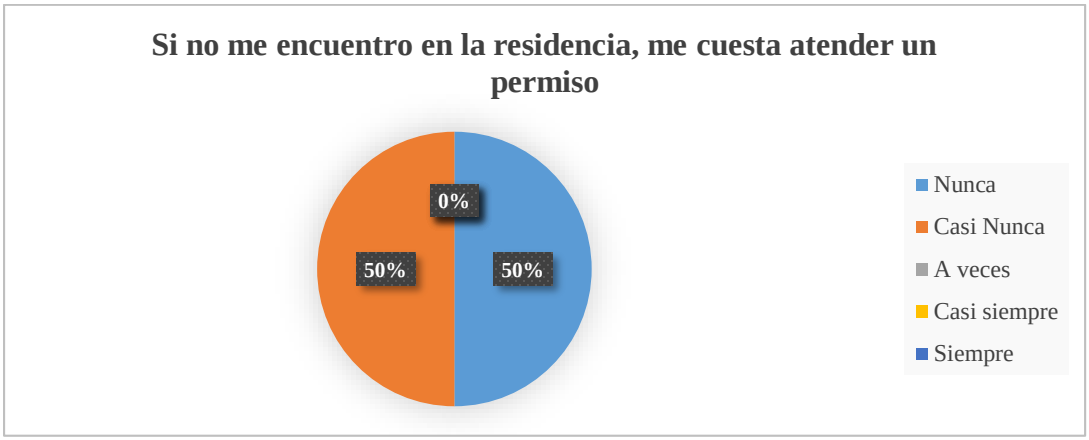


Gráfico 13. Análisis descriptivo - Pregunta 6 (postest) – Elaboración propia.

INTERPRETACIÓN:

- **Pretest:** Del total de preceptores encuestados, el 50% consideró que siempre les costaba atender un permiso en caso no se encontraban en la residencia, mientras que un 25% casi siempre, y otro 25% a veces.

- **Postest:** Del total de preceptores encuestados, el 50% consideran que nunca les cuesta atender un permiso en caso no se encuentran en la residencia, mientras que un 50%, casi nunca.

5.1.4. Análisis de satisfacción de uso del sistema

En esta sección se detalla la percepción de los usuarios finales (preceptores) respecto al uso del aplicativo web y móvil.

Tabla 53

Pregunta N° 1 - Presenta una interfaz amigable y fácil de usar – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	3
Excelente	1

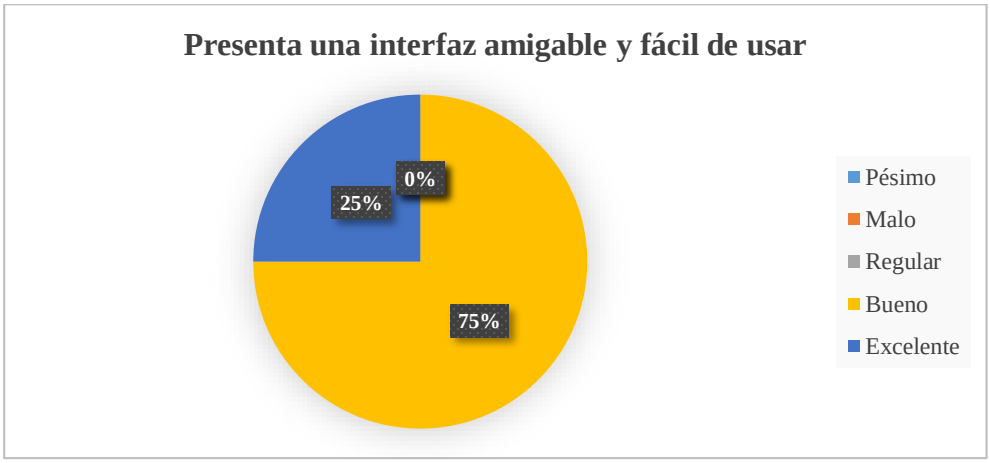


Gráfico 14. Análisis de satisfacción - Pregunta 1, Interfaz amigable – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 25% califican como excelente, y el 75% restante como bueno a la interfaz de la aplicación “Control Residente”.

Tabla 54

Pregunta N° 2 - El tiempo de respuesta es óptimo – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	3
Excelente	1

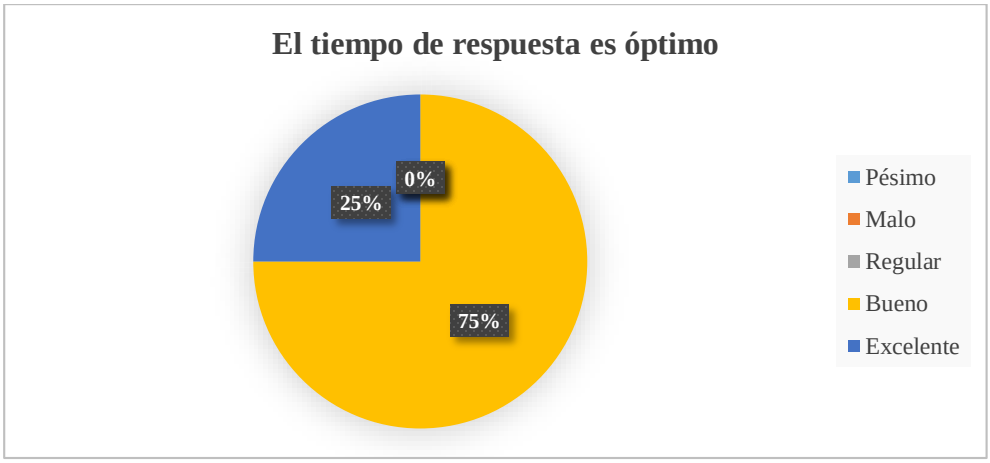


Gráfico 15. Análisis de satisfacción - Pregunta 2, tiempo de respuesta es óptimo – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 25% califican como excelente, y el 75% restante como bueno al tiempo de respuesta de la aplicación “Control Residente”.

Tabla 55

Pregunta N° 3 - Me brinda información actualizada en cualquier momento que requiera – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	2
Excelente	2

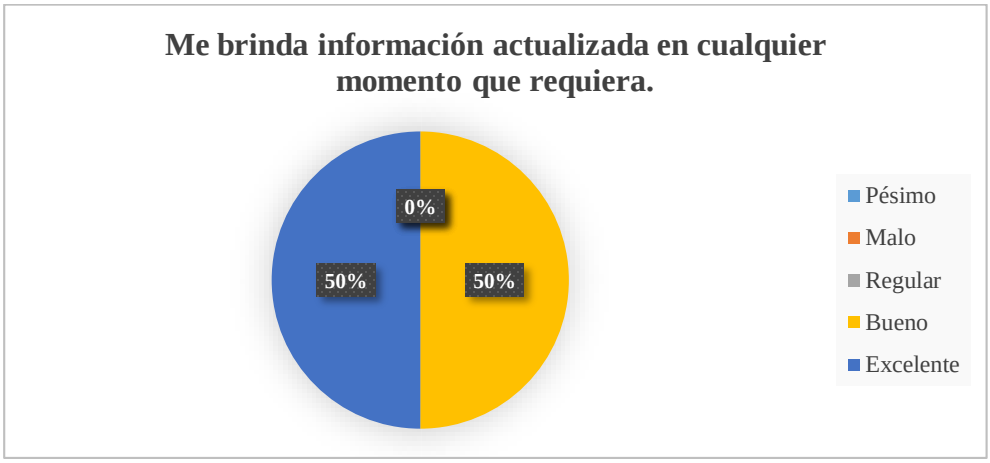


Gráfico 16. Análisis de satisfacción - Pregunta 3, información actualizada – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 50% califican como excelente, y el 50% restante como bueno respecto a la información actualizada que brinda la aplicación “Control Residente”.

Tabla 56

Pregunta N° 4 - Me ayuda a reducir el material físico – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	2
Excelente	2

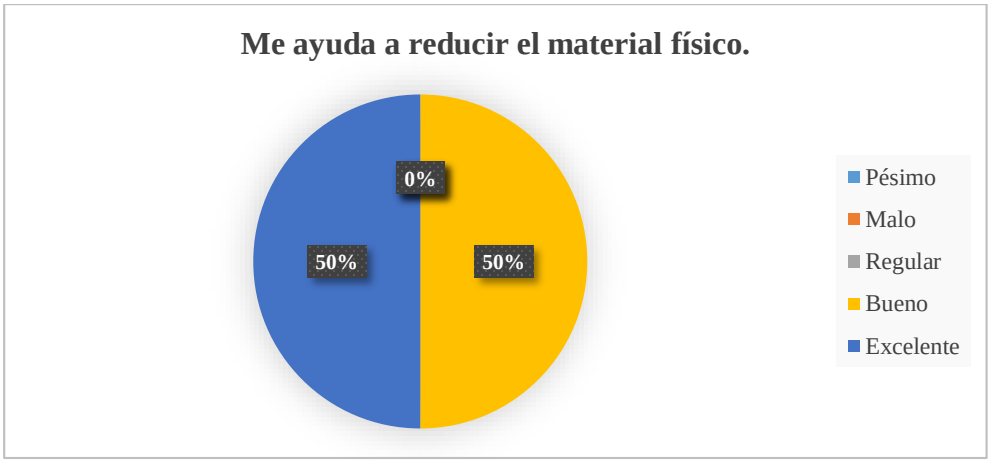


Gráfico 17. Análisis de satisfacción - Pregunta 4, ayuda a reducir el material fisico – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 50% califican como excelente, y el 50% restante como bueno respecto a la reducción de material físico gracias a la aplicación "Control Residente".

Tabla 57

Pregunta N° 5 - Me ayuda a gestionar la solicitud de los permisos con facilidad – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	1
Excelente	3

Gráfico 18. Análisis de satisfacción - Pregunta 5, ayuda a gestionar la solicitud de los permisos – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 75% califican como excelente, y el 25% restante como bueno a la facilidad de gestionar los permisos con la ayuda de la aplicación "Control Residente".

Tabla 58

Pregunta N° 6 - Me es fácil realizar el seguimiento de los permisos – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	0
Excelente	4

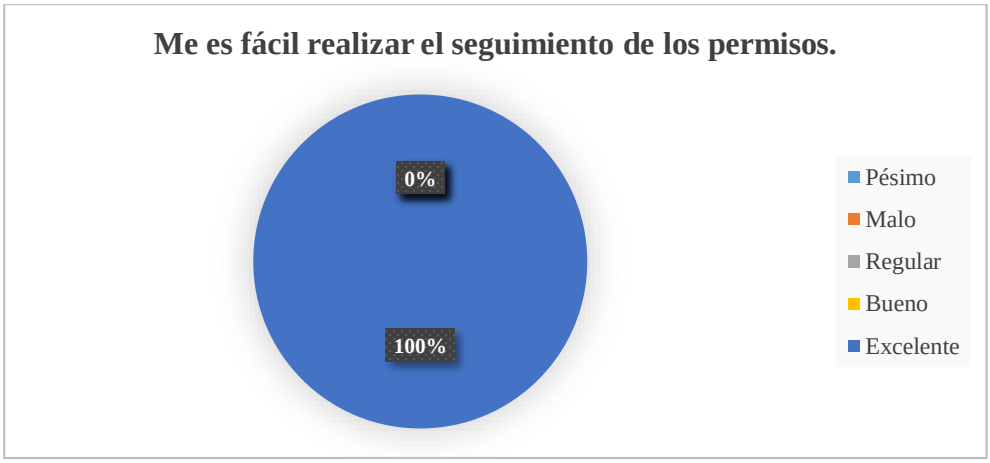


Gráfico 19. Análisis de satisfacción - Pregunta 6, facilidad para realizar seguimientos – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 100% califican como excelente la facilidad de realizar los seguimientos de los permisos con la aplicación “Control Residente”.

Tabla 59

Pregunta N° 7 - Calidad de los beneficios que brinda la aplicación "Control Residente" – Elaboración propia.

Calificación	Cantidad
Pésimo	0
Malo	0
Regular	0
Bueno	1
Excelente	3

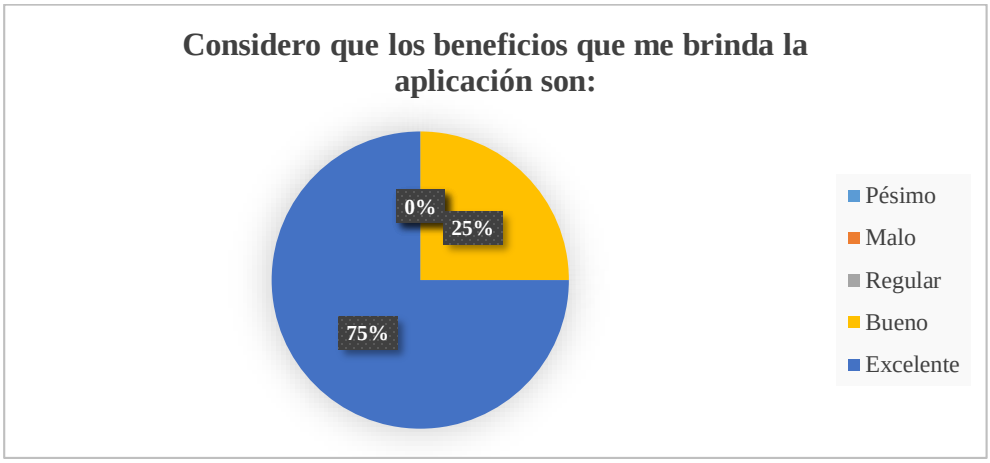


Gráfico 20. Análisis de satisfacción - Pregunta 7, beneficios que brinda la aplicación – Elaboración propia.

INTERPRETACIÓN: Del total preceptores encuestados, el 75% califican como excelente, y el 25% restante como bueno a la calidad de los beneficios que brinda la aplicación “Control Residente”.

Capítulo 6

Conclusiones y Recomendaciones

6.1. Conclusiones

Los resultados de la investigación presentada, fueron expuestos en el capítulo anterior, sin embargo, mencionaremos los resultados más relevantes en respuesta a los objetivos generales y específicos planteados al inicio de la investigación.

- Se puede sostener que la aplicación "Control Residente", es eficaz en la mejora del control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, campus Tarapoto.
- Se concluye, que con el uso de la aplicación "Control Residente", el número de permisos cortos y largos brindados a los estudiantes ha aumentado significativamente.
- Se concluye, que con el uso de la aplicación "Control Residente", se redujo en un 75% el uso de material fungible en el procedimiento de registro y seguimiento de los permisos.
- Se concluye, que con el uso de la aplicación "Control Residente", se ha reducido significativamente el tiempo de atención, registro y reporte de los permisos; mejorando la atención a la comunidad de residentes.

En lo referente al grupo de estudio podemos concluir lo siguiente:

- Del total de preceptores encuestados, el 25% calificó como: *excelente*, y el 75% restante como: *bueno*; al tiempo de respuesta de la aplicación: "Control Residente", es decir, se optimizó el tiempo de ejecución de las operaciones. Respecto del tiempo para realizar un permiso, de 172.50 ± 92.87 segundos a 41.25 ± 14.36 segundos; tiempo para realizar un reporte, de 1125 ± 512.35 segundos a 23.75 ± 8.54 segundos.

- Del total de preceptores encuestados, más del 75% mencionaron que con el uso de la aplicación: “Control Residente, les resulta fácil realizar el seguimiento de cada permiso. Este resultado se refleja gracias a que la aplicación brinda información actualizada y en tiempo real, favoreciéndoles tomar decisiones en el momento adecuado.
- Se concluye, que la aplicación “Control Residente” tiene una excelente percepción por parte de los encuestados, y como herramienta tecnológica, una ayuda que hace que sus funciones sean más eficientes y eficaces; aprobado por más del 75% de encuestados.

6.2. Recomendaciones

En función a lo investigado hasta el momento, recomendamos lo siguiente:

- Validar constantemente los requerimientos y funcionalidades con la ayuda del cliente, para evitar crear funcionalidades no solicitadas o invertir recursos innecesariamente en razones demasiado cambiantes por parte del mismo (cliente).
- Capacitar a los usuarios finales en el uso del aplicativo con el fin de sacar el mejor provecho a las funcionalidades del mismo.
- Es indispensable tener al cliente inmerso en todo el proceso de implementación de la aplicación. Esto ayudará a lograr el objetivo de los equipos Scrum.
- Al desarrollar aplicaciones móviles es importante recordar que este tenga una interfaz amigable, con colores que no cansen la vista del usuario, y mas que todo sea fácil de usar, con las opciones necesarias y en los lugares indicados, es decir, donde comúnmente los usuarios suelen encontrar en las aplicaciones más utilizadas.
- Aplicar Scrum como metodología ágil teniendo en cuenta los lineamientos que plantea, debido a que garantiza el éxito de los proyectos. En este caso en particular, ayudó a mantener un equipo bien organizado, unido y focalizado en el logro del objetivo final.

Referencias

Alaimo, D. M. (2013). *Proyectos Ágiles con Scrum: Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos.* Retrieved from https://www.amazon.es/gp/product/9874576340/ref=as_li_ss_tl?ie=UTF8&linkCode=sl1&tag=laoficdeproyd-21&linkId=0cd380637f52c53cd41a4ac796d91208

Arévalo, J. A. (2007). *Gestión de la Información, gestión de contenidos y conocimiento*. 193–206. https://doi.org/10.1076/epri.10.10.36.6816

Arjonilla Domínguez, S. J., & Medina Garrido, J. A. (2007). *La gestión de los sistemas de información en la empresa*. Retrieved from https://www.edicionespiramide.es/libro.php?id=2360732#

Asana. (2019). Portal de Asana. Retrieved January 20, 2019, from https://asana.com/es

Bahit, E. (2012). *Scrum y eXtreme Programming para Programadores*. 162. Retrieved from https://openlibra.com/es/book/scrum-y-extreme-programming-para-programadores

Beck, K. (2000). *Extreme programming eXplained : embrace change*. Retrieved from https://books.google.com.pe/books?hl=es&lr=&id=G8EL4H4vf7UC&oi=fnd&pg=PR13&dq=roles+de+extreme+programming&ots=jawJzunUvs&sig=OffxYmcn7El6mocg71Bri9-ZgG0#v=onepage&q=roles de extreme programming&f=false

Beck, K., Beedle, M., Van Bennekum, A., & Cockburn, A. (2001). Manifesto for Agile Software Development. Retrieved January 28, 2019, from https://agilemanifesto.org/

Buitrago, B. (2010). El Lenguaje de Programación Comunicación Programador y Computadora. *Journal Boliviano de Ciencias*, *7*, 60–62.

Chacon, S., & Straub, B. (2014). *Pro Git*. https://doi.org/10.1007/978-1-4842-0076-6

Chatterjee, A. (2017). Information Systems and Networks. In *Elements of Information Organization and Dissemination* (pp. 353–426). https://doi.org/10.1016/B978-0-08-102025-8.00022-3

Chiluisa, A., & Loarte, B. (2014). *Desarrollo e Implantación del Sistema de Control de Inventarios y Gestión de Laboratorios para la Facultad de Ciencias de la Escuela Politécnica Nacional*. 116. Retrieved from http://bibdigital.epn.edu.ec/bitstream/15000/7732/1/CD-5638.pdf

Cockburn, A., & Williams, L. (2001). *The Costs and Benefits of Pair Programming*. Retrieved from https://collaboration.csc.ncsu.edu/laurie/Papers/XPSardinia.PDF

DB-Engines. (2019). DB-Engines Ranking - Trend Popularity. Retrieved January 13, 2019, from https://db-engines.com/en/ranking_trend

Firebase. (2019). Firebase. Retrieved January 20, 2019, from https://firebase.google.com/products/?hl=es-419

Garcia, S. (2015). *La guía definitiva de Django: Desarrolla aplicaciones web de forma rápida y sencilla*. Retrieved from http://github.com/saulgm/djangobook.com

Garro, A. (2011). *HTM5*. Retrieved from https://openlibra.com/es/book/download/html5

GitLab. (2019). GitLab Documentation. Retrieved January 20, 2019, from https://docs.gitlab.com/ee/README.html

Grant, A. (2014). The Basics of AngularJS. In *Beginning AngularJS* (pp. 35–45). https://doi.org/10.1007/978-1-4842-0160-2_2

Guía SBOK™. (2017). *Una guía para el CUERPO DE CONOCIMIENTO DE SCRUM* (3rd ed.). SCRUMstudy™.

Guill Fuster, H., Hormigo, I. G., José María Joana, & Rodríguez, J. R. (2011). *Fundamentos de Sistemas de Información*. Retrieved from https://openlibra.com/es/book/fundamentos-de-sistemas-de-informacion

Gunicorn. (2019). Gunicorn - Servidor HTTP WSGI de Python para UNIX. Retrieved January 28, 2019, from https://gunicorn.org/

HotFrameworks. (2019). Web framework rankings. Retrieved January 13, 2019, from https://hotframeworks.com/

Iruela, J. (2016). Los gestores de bases de datos más usados. Retrieved January 13, 2019, from Revista Digital INESEM website: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/

JetBrains. (2019). All Developer Tools and Products. Retrieved January 20, 2019, from https://www.jetbrains.com/products.html?fromMenu#type=ide

Joskowicz, J. (2008). *Reglas y Prácticas en eXtreme Programming*. Retrieved from https://iie.fing.edu.uy/~josej/docs/XP - Jose Joskowicz.pdf

Kendall, K. E., & Kendall, J. E. (2005). *Análisis y diseño de sistemas*. Retrieved from http://gen.lib.rus.ec/book/index.php?md5=CAC6302B7745C92EBE99C5AF324FE873

Kyocera. (2017). Los 6 principales tipos de sistemas de información. Retrieved January 13, 2019, from Kyocera website: https://smarterworkspaces.kyocera.es/blog/los-6-principales-tipos-sistemas-informacion/

Lapiedra, R., Devece, C., & Guiral, J. (2011). *Introducción a la gestión de sistemas de información en la empresa* (C. de la Plana, Ed.).

Laravel. (2019). Laravel - The PHP Framework For Web Artisans. Retrieved January 13, 2019, from https://laravel.com/

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, *5*(26), 17. https://doi.org/1666-1680

Letelier, Patricio, & Penadés, M. C. (2002). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, *05*(26), 17. Retrieved from http://www.cyta.com.ar/ta0502/v5n2a1.htm

Liang, D. (2012). Introduction to Java programming: brief version. In *Hpc-Education.Org*. https://doi.org/10.1136/bmj.1.5135.1458

Matthew Norman. (2004). *Database Design Manual: using MySQL for Windows*. https://doi.org/10.1007/b97536

Menzinsky, A., López, G., & Palacio, J. (2019). *Scrum Master*. Retrieved from http://www.streetsofdublin.com/

Millet, P. B. (2009). *Laboratorio de PHP y MySQL*. Retrieved from https://openlibra.com/es/book/laboratorio-de-php-y-mysql#details

Mountain Goat Software. (1998a). Scrum Methodology and Project Management. Retrieved February 6, 2019, from https://www.mountaingoatsoftware.com/agile/scrum

Mountain Goat Software. (1998b). Scrum Methodology and Project Management.

Navicat. (2018). *Navicat Premium named "Best DBA Solution" by DBTA Readers*. Retrieved from https://www.navicat.com/es/company/press/793-navicat-premium-named-best-dba-solution-by-dbta-readers.html

NGINX. (2019). Bienvenido a NGINX Wiki. Retrieved January 28, 2019, from https://www.nginx.com/resources/wiki/

Palacia, J. (2015). *Gestión de proyectos Scrum Manager*. https://doi.org/10.1021/ma0001032

Postman Inc. (2019). Postman | API Development Environment. Retrieved January 20, 2019, from https://www.getpostman.com/

Pries, K. H., & Quigley, J. M. (2011). *Scrum project management*. Retrieved from http://gen.lib.rus.ec/book/index.php?md5=EFE9208A61A50A249B4F8F50AE03B46C

Priolo, S. (2009). *Métodos Ágiles* (MP Edicion). Retrieved from https://www.iberlibro.com/METODOS-AGILES-Spanish-Edition-SEBASTIAN-PRIOLO/10433583113/bd

PuTTY. (2019). PuTTY. Retrieved January 21, 2019, from https://www.putty.org/

Python. (2017). Beginners Guide/Overview. Retrieved January 7, 2019, from https://wiki.python.org/moin/BeginnersGuide/Overview

Rafael, E., Morales, M., Guzmán, R., & Marcial, I. (2017). Sistema integral web para la gestión, control y seguimiento de residencias profesionales, servicio social y visitas a empresas. *Revista de Sistemas y Gestión Educativa*, *4*, 44–54. Retrieved from www.ecorfan.org/bolivia

Ruiz de la Peña, J., & Cuba Céspedes, I. (2010). Sistema de gestión de información para la Residencia Universitaria de la Universidad de Holguín "Oscar Lucero Moya." *Ciencias Holguín*, *XVI*(1), 1–8. Retrieved from http://www.redalyc.org/articulo.oa?id=181517919017

Schultz, D., & Cook, C. (2007). *Beginning HTML with CSS and XHTML*. https://doi.org/10.1007/978-1-4302-0350-6

Schwaber, K., & Sutherland, J. (2010). *SCRUM GUIDE*. 21. Retrieved from http://gen.lib.rus.ec/book/index.php?md5=248D7F0CBF5E367ECBA911656CF06438

Software Freedom Conservancy. (2019). Git. Retrieved January 20, 2019, from https://git-scm.com/

Sturgeon, P., & Lockhart, J. (2015). PHP : The " Right " Way. *Leanpub*. Retrieved from https://openlibra.com/es/book/download/php-the-right-way

Suehring, S., & Valade, J. (2013). PHP, MySQL, JavaScript & HTML5 All-in-One For Dummies. In *John Wiley & Sons, Inc.*

SUNEDU. (2014). *Ley Universitaria N.° 30220*. Retrieved from http://www.minedu.gob.pe/reforma-universitaria/pdf/ley_universitaria.pdf

Supervisor. (2019). Supervisor: A Process Control System. Retrieved January 28, 2019, from http://supervisord.org/

The CentOS Project. (2019). About CentOS. Retrieved January 28, 2019, from https://centos.org/about/

TIOBE. (2019). TIOBE Index for January 2019. Retrieved January 7, 2019, from https://www.tiobe.com/tiobe-index/

Vega, J. F., & Van Der Henst, C. (2011). *Guía HTML5: El presente de la web*. 1–47. Retrieved from https://openlibra.com/es/book/download/guia-html5-el-presente-de-la-web

W3Techs Web Technology Surveys. (2019). Usage Statistics and Market Share of Linux for Websites. Retrieved January 28, 2019, from https://w3techs.com/technologies/details/os-linux/all/all

Anexos

Anexo 1. Solicitud de aceptacion del proyecto.

Morales, 07 de noviembre del 2018

Magister
Joel Ricardo Turpo Chaparro
Director de Bienestar Universitario de la Universidad Peruana Unión Filial Tarapoto

Solicita: Autorización de ejecución de proyecto de tesis en el área de Bienestar Universitario

De nuestra especial consideración.

Los que suscriben, Eliacer Fernandez Guevara, identificado con DNI N° 77573082 y Heber Quelion Flores Chura, identificado con DNI N° 47390945, ante usted con el debido respeto exponemos lo siguiente:

Que con el propósito de terminar el desarrollo nuestro proyecto de tesis para la obtención del grado de Ingeniero de Sistemas, solicitamos permiso para la Implementación del Sistema de Información multiplataforma (Móvil, Web) "Control Residente" en las residencias universitarias, con el objetivo de optimizar el flujo de información entre los actores que intervienen, ayudando a optimizar los recursos y mejorando la calidad de servicio a los estudiantes.

En el proyecto presentado se pretende abordar la gestión de permisos, control de asistencias al comedor universitario, consulta de temas financieros y académicos de los estudiantes.

Esperando su pronta respuesta, agradecemos de antemano su atención.

Atentamente,

UNIVERSIDAD PERUANA UNIÓN BIENESTAR UNIVERSITARIO DIRECTOR

Eliacer Fernandez Guevara
DNI: 77573082

Heber Quelion Flores Chura
DNI: 47390945

Anexo 2. Constancia de autorización del proyecto.

Una Institución Adventista

CONSTANCIA

La Universidad Peruana Unión Filial Tarapoto, hace constar:

Que los Bachilleres. ***Eliacer Fernandez Guevara y Heber Quelion Flores Chura***, Identificados con DNI N° ***77573082*** y N° ***47390945***, están realizando en nuestra Institución el proyecto de investigación titulado "***Implementación de la aplicación "Control Residente", bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, para mejorar el control de servicios brindados a los estudiantes residentes de la Universidad Peruana Unión, Filial Tarapoto***".

Se expide la presente constancia a solicitud escrita del Interesado, para los usos y fines que crea conveniente.

Morales, Tarapoto, 09 de enero de 2019.

Mg. Joel Ricardo Turpo Chaparro
Director de Bienestar Universitario UPeU - FT

Anexo 3. Juicio de expertos.

Ficha de Juicio de Expertos

1.- Datos informativos

Apellidos y nombres: Pérez Rivera Jessica
Cargo dentro de la institución donde labora:
Titulo:
Grado: Lic. matemática / magister
Nombre del instrumento de evaluación:
Autores del instrumento:

2- Aspectos de validación:

Nº	INDICADORES	CRITERIOS	Deficiente 0-20 %	Regular 21-40%	Bueno 41-60%	Muy Bueno 61-80%	Excelente 81-100%
1	Claridad	Esta formulado con el lenguaje apropiado					✓
2	Objetividad	Esta expresado en conductas observables					✓
3	Actualidad	Adecuado al avance de la tecnología y ciencia					✓
4	Organización	Existe una organización lógica					✓
5	Intencionalidad	Adecuado para valorar aspectos referencias al estado actual de estudiante en relación a la investigación					✓
6	Consistencia	Basado en aspectos teóricos científicos					✓

3.- Opinión de aplicación y promedio de validación

Luego de valorar la hoja en el promedio de respuesta, el instrumento alcanza un porcentaje de .....%, lo cual opino de su validez y su confiabilidad para ser aplicado.

Moyobamba, 20/05/19	42581319	
Lugar y Fecha	DNI	Firma de Expertos

Anexo 4. Instrumento pretest.

Universidad Peruana Unión

Proyecto de Investigación

"Automatización del control de servicios brindados a los estudiantes residentes, bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la Universidad Peruana Unión, Filial Tarapoto"

Encuesta sobre la gestión de permisos antes de la implementación del aplicativo "Control Residente"

Agradecemos de antemano su colaboración en el llenado de esta encuesta que se realiza con fines investigativos.

1. Duración al procesar una solicitud

Escriba el valor en minutos en el siguiente recuadro.

#	Descripción	Valor (min)
1	¿Aproximadamente cuánto tiempo demoraba en atender una solicitud de permiso?	
2	La respuesta a una solicitud ¿En cuánto tiempo aproximado lo atendía?	
3	¿Cuánto tiempo invertía en realizar un reporte mensual de permisos?	

2. Percepción de la gestión de permisos

Califique las siguientes situaciones de acuerdo a como las percibía.
Marque con una X según estime conveniente.

#	Característica	Calificación				
		1 Nunca	2 Casi nunca	3 A veces	4 Casi siempre	5 Siempre
1	Me costaba trabajo organizar la información.					
2	Me era fácil realizar el seguimiento de cada permiso.					
3	Utilizaba material físico para generar los permisos.					
4	Daba tiempo extra para realizar los reportes mensuales y/o semanales.					
5	Existía algún riesgo de que la información sea vulnerada, o sufra algún tipo de pérdida.					
6	Si no me encontraba en la residencia, me costaba atender una solicitud de permiso.					

Anexo 5. Instrumento postest.

Universidad Peruana Unión

Proyecto de Investigación

"Automatización del control de servicios brindados a los estudiantes residentes, bajo el marco de trabajo SCRUM y la metodología de desarrollo ágil Extreme Programming, en la Universidad Peruana Unión, Filial Tarapoto"

Encuesta sobre la gestión de permisos después de la implementación del aplicativo "Control Residente"

Agradecemos de antemano su colaboración en el llenado de esta encuesta que se realiza con fines investigativos.

1. Duración al procesar una solicitud

Escriba el valor en segundos en el siguiente recuadro.

#	Descripción	Valor (seg.)
1	¿Aproximadamente cuánto tiempo demora en atender una solicitud de permiso?	
2	La respuesta a una solicitud ¿En cuánto tiempo aproximado lo atiende?	
3	¿Cuánto tiempo invierte en realizar un reporte mensual de permisos?	

2. Percepción de la gestión de permisos

Califique las siguientes situaciones de acuerdo a como las percibía.

Marque con una X según corresponda.

#	Característica	Calificación				
		1 Nunca	2 Casi nunca	3 A veces	4 Casi siempre	5 Siempre
1	Me cuesta trabajo organizar la información.					
2	Me es fácil realizar el seguimiento de cada permiso.					
3	Utilizo material físico para generar los permisos.					
4	Doy tiempo extra para realizar los reportes mensuales y/o semanales.					
5	Existe algún riesgo de que la información sea vulnerada, o sufra algún tipo de pérdida.					
6	Si no me encuentro en la residencia, me cuesta atender una solicitud de permiso.					

3. Sobre uso del aplicativo "Control Residente"

Marque con una X según estime conveniente.

#	Característica	Calificación				
		1 Pésimo	2 Malo	3 Regular	4 Bueno	5 Excelente
1	Presenta una interfaz amigable y fácil de usar.					
2	El tiempo de respuesta es óptimo.					
3	Me brinda información actualizada en cualquier momento que requiera.					
4	Me ayuda a reducir el material físico.					
5	Me ayuda a gestionar la solicitud de los permisos con facilidad.					
6	Me es fácil realizar el seguimiento de los permisos.					
7	Considero que los beneficios que me brinda la aplicación son:					

Anexo 6. Metodología Scrum

Se realizó historias de usuario, estimaciones, checklist de historias aprobadas, entre otros.

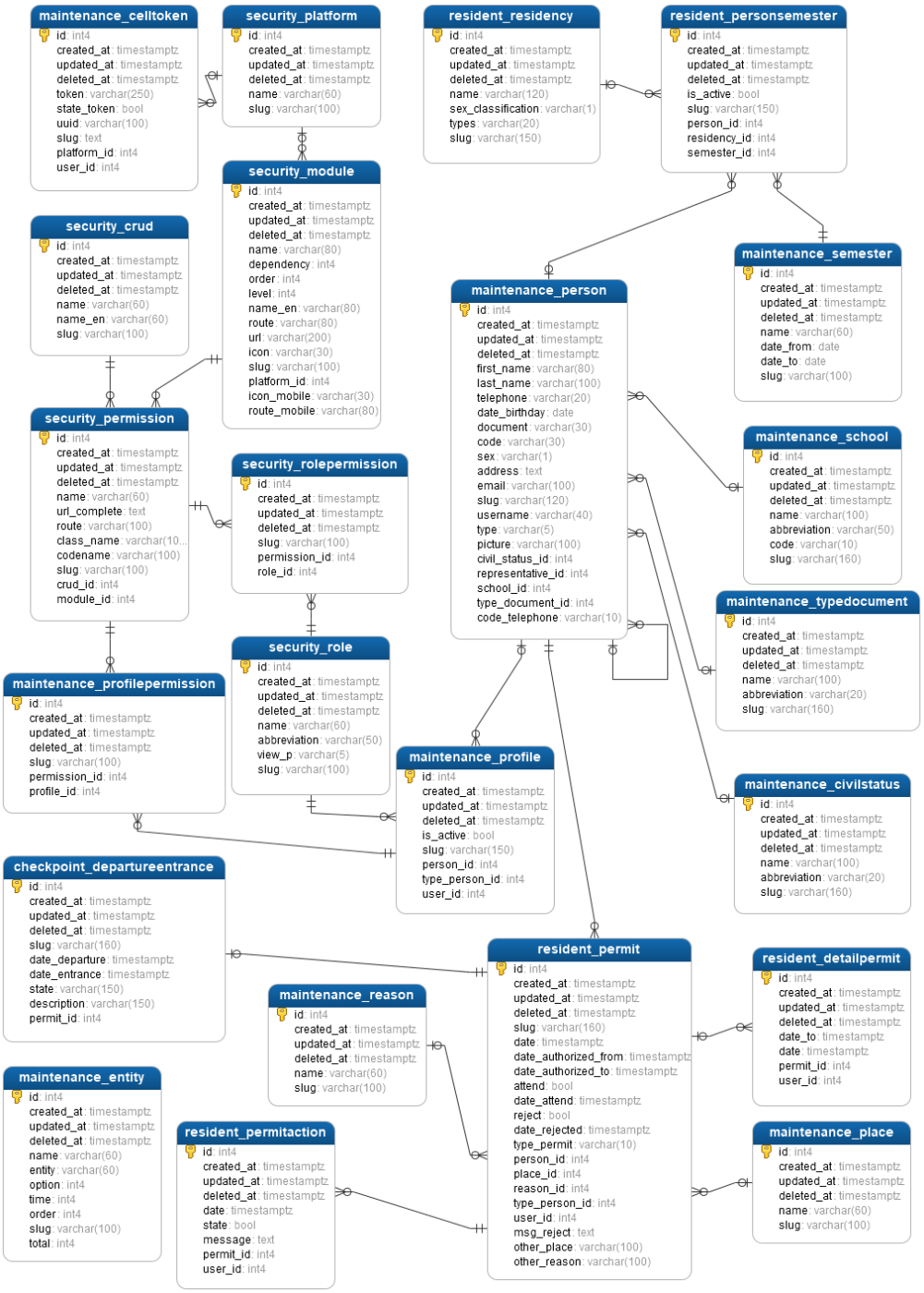
N°	Historia de Usuario
1	Registro de residentes
2	Registro de apoderados
3	Registro y configuración de preceptores
4	Registro de permisos
5	Consultar los estados de los permisos
6	Consultar permisos de sus apoderandos
7	Registrar entradas y salidas de los residentes
8	Registrar motivos de salida
9	Registrar lugares
10	Registrar usuarios del sistema
11	Notificaciones
12	Enlazar número telefónico a llamadas y WhatsApp
13	Presentar reportes mensuales de permisos

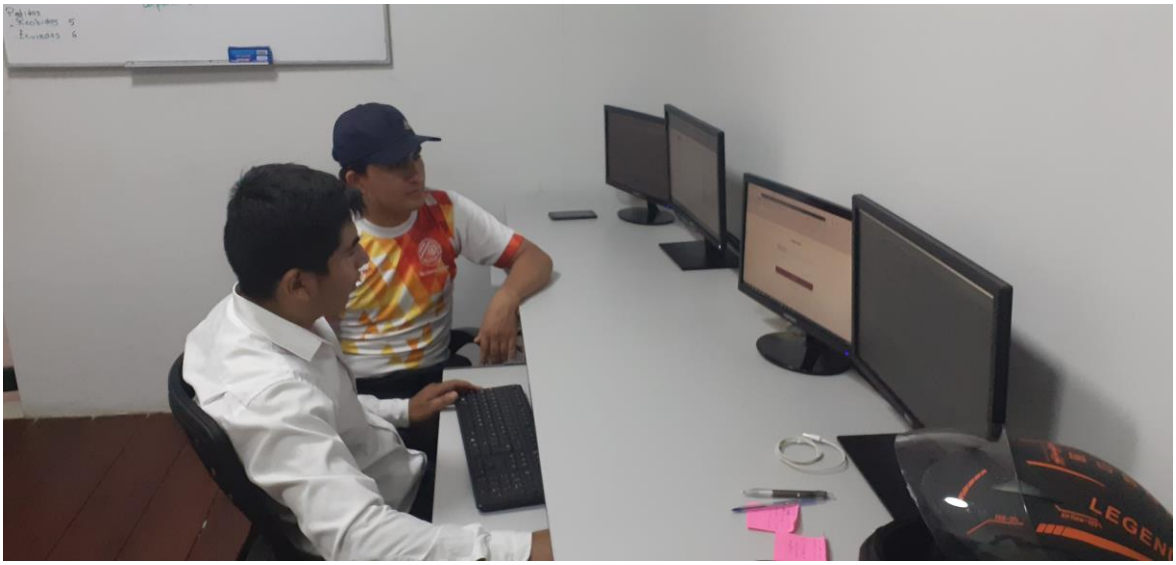
Nº	Historia	Ulices	Pedro	Jhan	Eliacer	Heber	Estimación
01	Registro de residentes	8	10	10	8	10	**10**
02	Registro de apoderados	5	6	6	5	7	**6**
03	Registro y configuración de preceptores	5	7	7	5	7	**7**
04	Registro de permisos	10	12	13	11	12	**12**
05	Consultar los estados de los permisos	7	9	8	8	9	**9**
06	Consultar permisos de sus apoderandos	5	6	6	5	6	**6**
07	Registrar entradas y salidas de los residentes	7	9	10	8	9	**9**
08	Registrar motivos de salida	4	4	4	4	4	**4**
09	Registrar lugares	4	4	4	4	4	**4**
10	Registrar usuarios del sistema	14	15	16	14	17	**16**
11	Notificaciones	6	8	8	6	9	**8**
12	Enlazar número telefónico a llamadas y WhatsApp	3	4	3	3	3	**4**
13	Presentar reportes mensuales de permisos	6	6	7	5	8	7
	Sumatoria						**102 puntos**

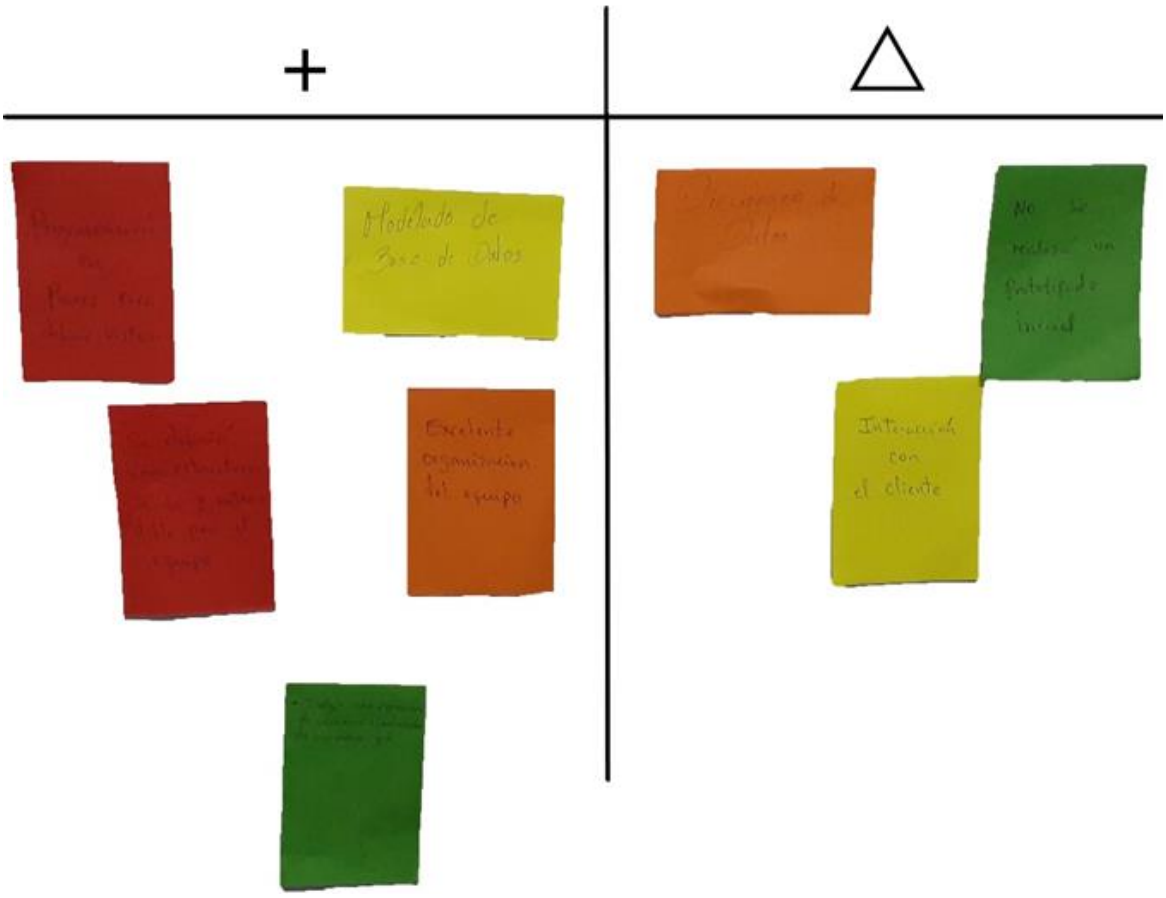
#	Historia de usuario	Validación
1	Registro de residentes	✓
2	Registro de apoderados	✓
3	Registro y configuración de preceptores	✓
4	Registro de permisos	✓
5	Consultar los estados de los permisos	✓
6	Consultar permisos de sus apoderandos	✓
7	Registrar entradas y salidas de los residentes	✓
8	Registrar motivos de salida	✓
9	Registrar lugares	✓
10	Registrar usuarios del sistema	✓
11	Notificaciones	✓
12	Enlazar número telefónico a llamadas y WhatsApp	✓
13	Presentar reportes mensuales de permisos	✓

Anexo 7. Metodología XP

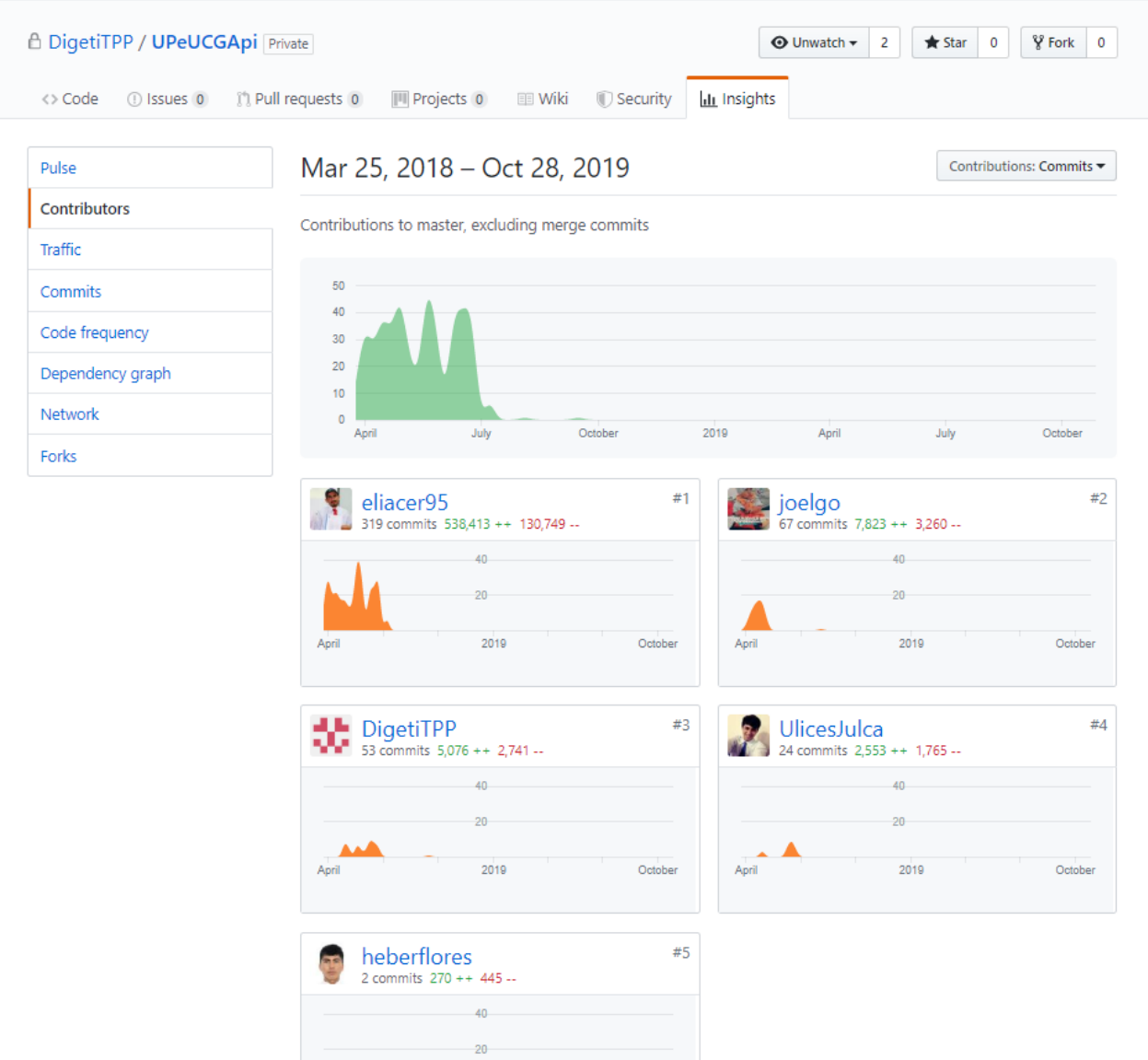
Se realizó modelo Entidad Relación, Programación en parejas, Retrospectivas, 40 horas semanales, propiedad colectiva del código, entre otros.

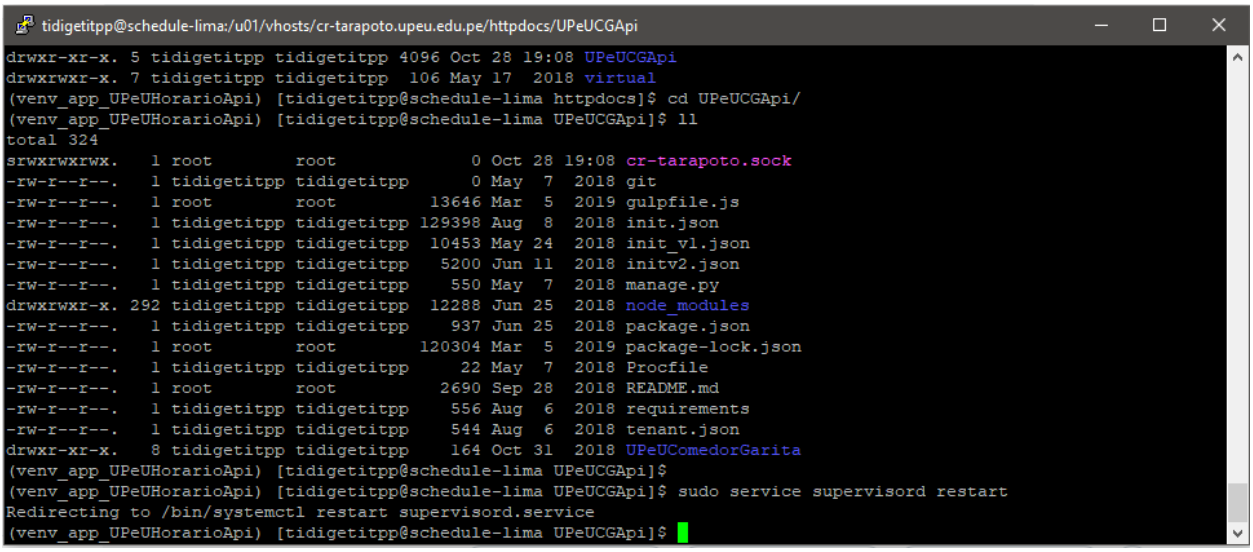


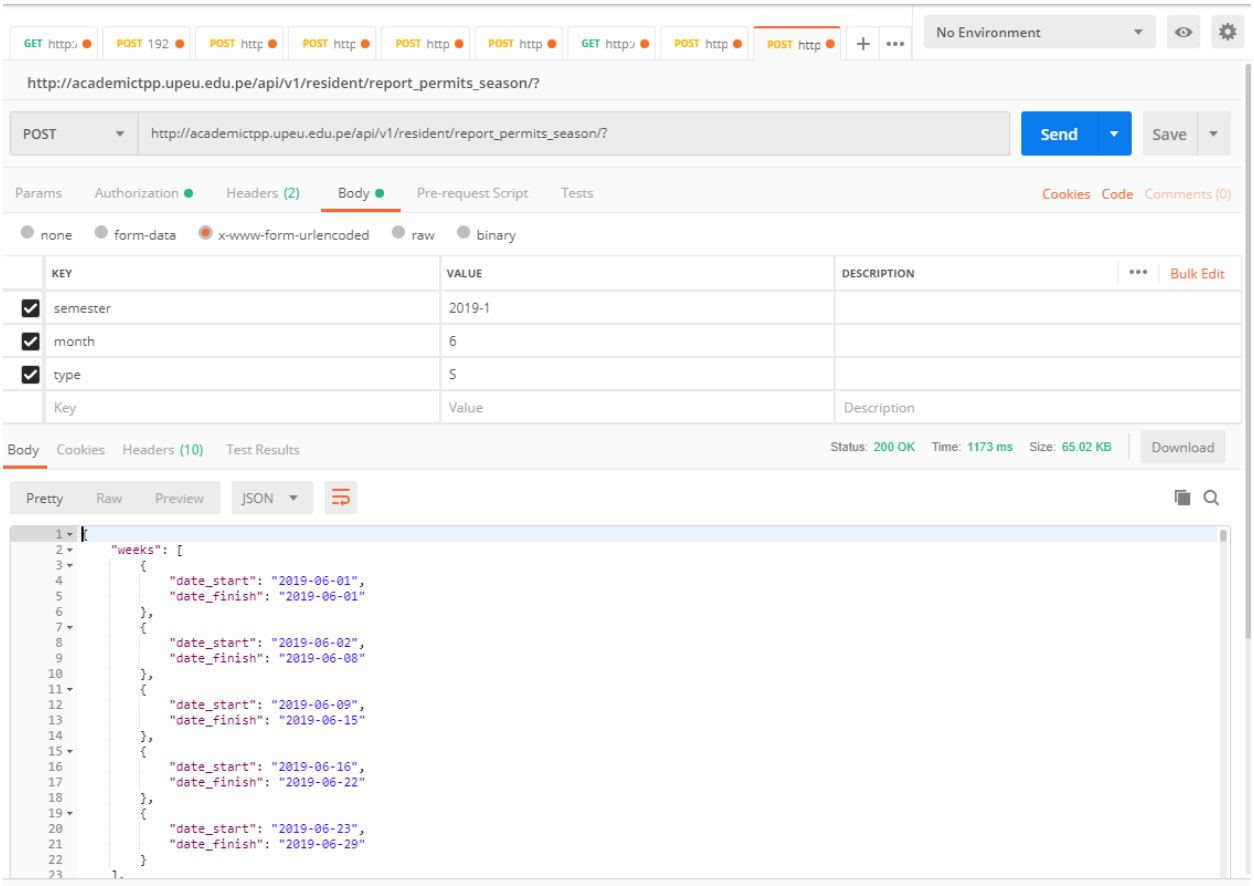



+
△
Modelado de Base de datos
Excelente organización del equipo
Interacción con el cliente

Nº	Historia de Usuario
1	Registro de residentes
2	Registro de apoderados
3	Registro y configuración de preceptores
4	Registro de permisos
5	Consultar los estados de los permisos
6	Consultar permisos de sus apoderandos
7	Registrar entradas y salidas de los residentes
8	Registrar motivos de salida
9	Registrar lugares
10	Registrar usuarios del sistema
11	Notificaciones
12	Enlazar número telefónico a llamadas y WhatsApp
13	Presentar reportes mensuales de permisos


DigetiTPP / UPeUCGApi Private
Unwatch 2
Star 0
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Wiki
Security
Insights
Pulse
Contributors
Traffic
Commits
Code frequency
Dependency graph
Network
Forks
Mar 25, 2018 – Oct 28, 2019
Contributions: Commits
Contributions to master, excluding merge commits
50
40
30
20
10
0
April
July
October
2019
April
July
October
eliacer95
#1
319 commits 538,413 ++ 130,749 --
joelgo
#2
67 commits 7,823 ++ 3,260 --
DigetiTPP
#3
53 commits 5,076 ++ 2,741 --
UlicesJulca
#4
24 commits 2,553 ++ 1,765 --
heberflores
#5
2 commits 270 ++ 445 --


tidigetitpp@schedule-lima:/u01/vhosts/cr-tarapoto.upeu.edu.pe/httpdocs/UPeUCGApi
drwxr-xr-x. 5 tidigetitpp tidigetitpp 4096 Oct 28 19:08 UPeUCGApi
drwxrwxr-x. 7 tidigetitpp tidigetitpp 106 May 17 2018 virtual
(venv_app_UPeUHorarioApi) [tidigetitpp@schedule-lima httpdocs]$ cd UPeUCGApi/
(venv_app_UPeUHorarioApi) [tidigetitpp@schedule-lima UPeUCGApi]$ ll
total 324
srwxrwxrwx. 1 root root 0 Oct 28 19:08 cr-tarapoto.sock
-rw-r--r--. 1 tidigetitpp tidigetitpp 0 May 7 2018 git
-rw-r--r--. 1 root root 13646 Mar 5 2019 gulpfile.js
-rw-r--r--. 1 tidigetitpp tidigetitpp 129398 Aug 8 2018 init.json
-rw-r--r--. 1 tidigetitpp tidigetitpp 10453 May 24 2018 init_v1.json
-rw-r--r--. 1 tidigetitpp tidigetitpp 5200 Jun 11 2018 initv2.json
-rw-r--r--. 1 tidigetitpp tidigetitpp 550 May 7 2018 manage.py
drwxrwxr-x. 292 tidigetitpp tidigetitpp 12288 Jun 25 2018 node_modules
-rw-r--r--. 1 tidigetitpp tidigetitpp 937 Jun 25 2018 package.json
-rw-r--r--. 1 root root 120304 Mar 5 2019 package-lock.json
-rw-r--r--. 1 tidigetitpp tidigetitpp 22 May 7 2018 Procfile
-rw-r--r--. 1 root root 2690 Sep 28 2018 README.md
-rw-r--r--. 1 tidigetitpp tidigetitpp 556 Aug 6 2018 requirements
-rw-r--r--. 1 tidigetitpp tidigetitpp 544 Aug 6 2018 tenant.json
drwxr-xr-x. 8 tidigetitpp tidigetitpp 164 Oct 31 2018 UPeUComedorGarita
(venv_app_UPeUHorarioApi) [tidigetitpp@schedule-lima UPeUCGApi]$
(venv_app_UPeUHorarioApi) [tidigetitpp@schedule-lima UPeUCGApi]$ sudo service supervisord restart
Redirecting to /bin/systemctl restart supervisord.service
(venv_app_UPeUHorarioApi) [tidigetitpp@schedule-lima UPeUCGApi]$


No Environment
http://academictpp.upeu.edu.pe/api/v1/resident/report_permits_season/?
POST
http://academictpp.upeu.edu.pe/api/v1/resident/report_permits_season/?
Send
Save
Params
Authorization
Headers (2)
Body
Pre-request Script
Tests
Cookies
Code
Comments (0)
none
form-data
x-www-form-urlencoded
raw
binary
KEY
VALUE
DESCRIPTION
Bulk Edit
semester
2019-1
month
6
type
5
Key
Value
Description
Body
Cookies
Headers (10)
Test Results
Status: 200 OK
Time: 1173 ms
Size: 65.02 KB
Download
Pretty
Raw
Preview
JSON
{
"weeks": [
{
"date_start": "2019-06-01",
"date_finish": "2019-06-01"
},
{
"date_start": "2019-06-02",
"date_finish": "2019-06-08"
},
{
"date_start": "2019-06-09",
"date_finish": "2019-06-15"
},
{
"date_start": "2019-06-16",
"date_finish": "2019-06-22"
},
{
"date_start": "2019-06-23",
"date_finish": "2019-06-29"
}