Imagen que contiene Texto

Descripción generada automáticamente

**PROFESOR PATROCINANTE:**

**GERMÁN BARRIENTOS**

**ESCUELA DE INFORMÁTICA Y TELECOMUNICACIONES**

**SOFTWARE DELACRUZ**

Proyecto Asignatura Proceso de Portafolio de Título

Para optar

al título de **INGENIERÍA EN INFORMÁTICA**

**PILAR BONNAULT MANCILLA – NICOLÁS GONZÁLEZ ESPINOZA - CHRISTOFER RUIZ ALMONACID**

**PUERTO MONTT – CHILE**

**2024**

# DEDICATORIA

“Dedico este proyecto a todas las personas que estuvieron a lo largo de esta etapa formativa dando su apoyo incondicional.” Pilar Bonnault Mancilla (2024)

“Este trabajo está dedicado principalmente a mis padres, quienes, a pesar de los problemas surgidos a lo largo de mi carrera, siempre estuvieron apoyándome de manera incondicional y desinteresada. Por su esfuerzo, cariño y constante apoyo, este trabajo es dedicado a ellos” Nicolás González Espinoza (2024).

“Este trabajo se lo dedico a todas las personas que me apoyaron en este proceso, profesores, amigos y familiares” Christofer Ruiz (2024)

# AGRADECIMIENTOS

**Como equipo queremos agradecer a nuestro profesor guía Germán Gabriel Barrientos Tereucan por su apoyo a lo largo de todo el proyecto y su buena disposición, además agradecer al grupo de profesores que nos enseñaron y guiaron durante todo el proceso académico.**

**Finalmente, a Carolina Martínez por la motivación constante y muestra de interés hacia sus estudiantes con el objetivo de ayudarlos como fuera necesario.**

**Pilar Bonnault**

**Quiero agradecer al profesor Francisco Calfún Gutiérrez por su confianza en mí y darme la oportunidad de participar en el proyecto inicial De La Cruz lavandería, su apoyo y paciencia en enseñar nuevas tecnologías.**

**Agradezco también al Centro de Innovación y Transferencia Tecnológica que siempre facilitaron sus instalaciones para poder hacer realidad muchas de las cosas necesarias para el desarrollo exitoso del proyecto.**

**Personalmente quiero agradecer a Liz Castillo García, mi familia y amigos específicamente a Nicolás González Espinoza y Christofer Ruiz Almonacid por el apoyo incondicional a lo largo de estos cuatro años que me alentaron, comprendieron y confiaron en mis habilidades lo que me dio una gran confianza para llegar a este momento tan importante en mi vida.**

**Finalmente, a mi abuela Lidia Vargas Águila que fue quien me dio la oportunidad de ingresar a la universidad y fue mi principal fuente de motivación a lo largo del tiempo.**

**Nicolás González**

**En esta oportunidad quisiera agradecerle a mi familia y amigos, quienes me apoyaron a lo largo de este proceso, agradeciendo a mis compañeros de carrera con quienes formé un vínculo bastante sólido, personas las cuáles puedo decir que fuera de lo académico, es decir, en la vida cotidiana son bastante valiosas.**

**Agradecer a mis amigos de la infancia Fernando España, Sergio Martínez y Vicente Andrade, quienes durante estos cuatro años fueron un pilar fundamental por su gran apoyo e inspiración que generaron en mí, personas que no dejaron que me rindiera y me sacaron de momentos difíciles, sacando lo mejor de mí, alentándome y haciendo que viera de las cosas que soy capaz cada vez que podían.**

**Finalmente, quiero agradecer a mi familia y abuelos, quienes han estado apoyándome incondicionalmente en cada momento, quienes no permitieron que me faltara algo, agradecer la educación que me brindaron desde pequeño e hicieron que me formara como el hombre que soy hoy en día, siendo una persona de bien y con valores.**

**Christofer Ruiz**

**Quiero agradecer a mi familia que me ayudó anímicamente en todo este proceso universitario, a mi mama por darme consejos, a mi papá por ayudar financieramente y darme la opción de ingresar al DUOC, a mi hermana, con ella hablo todos los problemas y cosas que estoy haciendo y a mis abuelos que me siempre me apoyaron en todo mi progreso y a mi primo Manuel Almonacid que me ayudó a encontrar practica cuando no sabía dónde buscar.**

**Agradecer al profesor German Gabriel Barrientos Tereucan que fue el hizo que me interesara en las bases de datos aportando todo el conocimiento que apliqué en todos estos años y por su amabilidad al momento de prestar ayuda en este proyecto y en general a todos los profesores que me ayudaron a resolver dudas.**

**Agradecer a mis amigos Pilar Bonnault y Nicolás González, ellos fueron los que me acompañaron en todo este camino académico, ayudando a solucionar problemas, ayudando anímicamente, además con ellos realice la mayoría de los trabajos y también este proyecto.**

**Y finalmente agradecer a mis amigos del colegio que me ayudaron a distraerme y relajarme cuando quería hacerlo.**

# SUMARIO

**Este documento presenta el proyecto “Software DeLaCruz”, su desarrollo y resultados obtenidos**

1. **Dedicatoria y agradecimientos: En este apartado dejamos los agradecimientos y dedicatorias a las personas que fueron relevantes en nuestro desarrollo académico y en la vida.**
2. **Antecedentes generales de proyecto: Se presenta el proyecto con su razón, las áreas en las que se relaciona con nuestra formación académica, la relevancia del proyecto en la lavandería y la relación que tiene con nuestros intereses profesionales.**
3. **Planteamiento del problema y solución: Este apartado está dedicado a el problema y la solución que se presentó con sus objetivos específicos y general.**
4. **Diseño metodológico: Se describe la metodología utilizada, la estructura y cómo la aplicamos en el proyecto.**
5. **Resultados: Se muestran diferentes imágenes con los resultados del sistema.**
6. **Conclusiones y recomendaciones: Experiencia personal de cada integrante y las limitaciones que presentaron.**

# ÍNDICE

[DEDICATORIA 1](#_heading=h.gjdgxs)

[AGRADECIMIENTOS 2](#_heading=h.30j0zll)

[SUMARIO 4](#_heading=h.1fob9te)

[ÍNDICE 5](#_heading=h.2et92p0)

[ÍNDICE DE TABLAS 6](#_heading=h.tyjcwt)

[ÍNDICE DE FIGURAS 7](#_heading=h.3dy6vkm)

[ÍNDICE DE ANEXOS 8](#_heading=h.1t3h5sf)

[1. ANTECEDENTES GENERALES 9](#_heading=h.4d34og8)

[1.1. Introducción 9](#_heading=h.2s8eyo1)

[1.2. Descripción del Proyecto 9](#_heading=h.17dp8vu)

[1.2.1. Tema 9](#_heading=h.3rdcrjn)

[1.2.2. Áreas de Desempeño 9](#_heading=h.26in1rg)

[1.2.3. Competencias o Unidades de Competencias 10](#_heading=h.lnxbz9)

[1.3. Fundamentación Proyecto APT 10](#_heading=h.35nkun2)

[1.3.1. Relevancia del proyecto APT 10](#_heading=h.1ksv4uv)

[1.3.2. Descripción de Proyecto APT 11](#_heading=h.44sinio)

[1.3.3. Pertinencia del proyecto con el perfil de egreso 11](#_heading=h.2jxsxqh)

[1.3.4. Relación con los intereses profesionales 11](#_heading=h.z337ya)

[1.3.5. Fortalezas y debilidades para desarrollar el proyecto APT 13](#_heading=h.3j2qqm3)

[1.4. Planteamiento del problema 13](#_heading=h.1y810tw)

[1.5. Objetivos 13](#_heading=h.4i7ojhp)

[1.5.1. Objetivo General 13](#_heading=h.2xcytpi)

[1.5.2. Objetivos Específicos 13](#_heading=h.1ci93xb)

[2. DISEÑO METODOLÓGICO 15](#_heading=h.3whwml4)

[2.1. ETAPA N°1 18](#_heading=h.qsh70q)

[2.1.1. Requerimientos del cliente 18](#_heading=h.3as4poj)

[2.2. ETAPA N°2 23](#_heading=h.yua28iv0ph3y)

[2.3. ETAPA N°3 38](#_heading=h.c60t2x12px1)

[2.4. ETAPA N°4 42](#_heading=h.delmcbszekn9)

[2.5. ETAPA N°5 48](#_heading=h.8dual6dec59p)

[2.6. ETAPA N°6 74](#_heading=h.h7ltu6m7zi6p)

[2.7. ETAPA N°7 75](#_heading=h.ul9nml95l8uj)

[3. RESULTADOS 76](#_heading=h.1pxezwc)

[4. CONCLUSIONES Y RECOMENDACIONES 85](#_heading=h.49x2ik5)

[4.1. Conclusiones 85](#_heading=h.2p2csry)

[4.2. Limitaciones y Recomendaciones. 86](#_heading=h.147n2zr)

[BIBLIOGRAFÍA 87](#_heading=h.3o7alnk)

[LINKOGRAFÍA 88](#_heading=h.23ckvvd)

[ANEXO A: 90](#_heading=h.ihv636)

[ANEXO B: 90](#_heading=h.32hioqz)

# ÍNDICE DE TABLAS

[Identificación de Scrum Master y Stakeholders 16](#_heading=h.z1lyalk2hzfo)

[DISEÑO METODOLÓGICO 16](#_heading=h.si8sm9z9y0n5)

[Requerimientos del cliente 19](#_heading=h.7aaudig0r7s9)

[Desarrollo ágil: Historias de usuario y criterios de aceptación 21](#_heading=h.4842wcy7qzxy)

# ÍNDICE DE FIGURAS

[Figura 1: Elaboración propia 16](#_heading=h.v5qw0bmqfk4d)

[Figura 2: Conexión cables (kit, 2018) 29](#_heading=h.rz47wm8goklx)

[Figura 3: Elaboración propia](#_heading=h.yojfaob84gl1)

[Figura 4: Elaboración propia](#_heading=h.yojfaob84gl1)

[Figura 5: Elaboración propia 29](#_heading=h.yojfaob84gl1)

[Figura 6: Elaboración propia 30](#_heading=h.ec71rwgqgk1)

[Figura 7: Elaboración propia 31](#_heading=h.hojfj0n2u3pi)

[Figura 8: Elaboración propia 33](#_heading=h.tx122xpjzuwk)

[Figura 9: Elaboración propia 33](#_heading=h.ymauacw1haf4)

[Figura 10: Elaboración propia 34](#_heading=h.kmprm44w64yg)

[Figura 11: Software DeLaCruz 56](#_heading=h.nhgl36boldvt)

[Figura 12: Software DeLaCruz 56](#_heading=h.cqi0eqj1dafn)

[Figura 13: Software DeLaCruz 56](#_heading=h.sabetzeq7jez)

[Figura 14: Software DeLaCruz 57](#_heading=h.1yzm3lchhpy)

[Figura 15: Software DeLaCruz 58](#_heading=h.2ii415fio8ob)

[Figura 16: Software DeLaCruz 58](#_heading=h.7v81ylclnsvi)

[Figura 17: Software DeLaCruz 59](#_heading=h.dwdy1q4l3kiv)

[Figura 18: Software DeLaCruz 59](#_heading=h.i08sue29gui4)

[Figura 19: Software DeLaCruz 60](#_heading=h.4d4b0r16opau)

[Figura 20: Software DeLaCruz 60](#_heading=h.etwe7vnfv0uo)

[Figura 21: Software DeLaCruz 61](#_heading=h.11ehlbt6xnh)

[Figura 22: Software DeLaCruz 61](#_heading=h.up40iy6mshyv)

[Figura 23: Software DeLaCruz 62](#_heading=h.jmp8uucjfw2)

[Figura 25: Software DeLaCruz 63](#_heading=h.ep0sn2w21u7y)

[Figura 26: Software DeLaCruz 63](#_heading=h.aaih63k6tmrb)

# ÍNDICE DE ANEXOS

[**ANEXO A: 72**](#_heading=h.ihv636)

[**ANEXO B: 72**](#_heading=h.32hioqz)

# ANTECEDENTES GENERALES

## Introducción

El presente documento busca mostrar el trabajo realizado y conclusiones obtenidas a lo largo de todo el desarrollo del proyecto “Software DeLaCruz”. El proyecto fue realizado con el propósito de apoyar en un proyecto de la empresa DeLaCruz Lavandería el cuál apuesta a la reducción de plásticos contaminantes utilizando pecheras reutilizables.

Durante el desarrollo de este proyecto se utilizaron distintas herramientas que permitieron la creación, organización y análisis de los procesos a los cuales se buscó mejorar constantemente. Además, se enfrentaron distintos desafíos relacionados con el desarrollo de hardware y con los tiempos de entrega y desarrollo de software, los cuales fueron abordados de la manera más rápida y adecuada posible para entregar el producto final en el mejor estado posible.

En este informe se detallan las etapas clave del proyecto, desde la planificación hasta la obtención del producto final, dando una conclusión de lo que se vivió.

## Descripción del Proyecto

### Tema

Este proyecto consiste en el desarrollo de un software de trazabilidad destinado a usarse en el seguimiento de pecheras reutilizables para tener en conocimiento la duración de estas, su fecha de fabricación, poder tener registro de la cantidad de lavado, este proyecto nace por la necesidad de disminuir la contaminación por residuos plásticos, específicamente lo que genera utilizar pecheras desechables, en la escala que lo hacen las industrias.

### Áreas de Desempeño

Para el desarrollo de este proyecto se incluyeron múltiples áreas de desempeño para poder cumplir con los requerimientos del cliente, entre estas se encuentran:

1. Desarrollo de software: Que consiste en el diseño y programación del sistema que incluye tanto el frontend (React) y backend (Node.js).
2. Bases de datos: Se crean y manejan las bases de datos en MySQL donde se almacenarán todos los datos que sean registrados en el sistema.
3. Tecnología RFID y Arduino: Se utiliza para el desarrollo del hardware que cumple con la función de lectura y escritura de chips y lectura continua para un registro eficiente y rápido.
4. Gestión de proyectos: Se realiza la planificación, distribución de tareas, asignación de tiempos y control de avances.

### Competencias o Unidades de Competencias

Para desarrollar este proyecto, se aplicaron múltiples competencias que corresponden a la carrera Ingeniería en Informática, siempre siguiendo la línea de los requerimientos necesarios para poder llevar a cabo este proyecto, algunas de estas competencias son:

1. “Administrar la configuración de ambientes, servicios de aplicaciones y bases de datos en un entorno empresarial a fin de habilitar operatividad o asegurar la continuidad de los sistemas que apoyan los procesos de negocio de acuerdo con los estándares definidos por la industria.
2. Ofrecer propuestas de solución informática analizando de forma integral los procesos de acuerdo con los requerimientos de la organización.
3. Desarrollar una solución de software utilizando técnicas que permitan sistematizar el proceso de desarrollo y mantenimiento, asegurando el logro de los objetivos.
4. Construir modelos de datos para soportar los requerimientos de la organización de acuerdo con un diseño definido y escalable en el tiempo.
5. Programar consultas o rutinas para manipular información de una base de datos de acuerdo con los requerimientos de la organización.
6. Implementar soluciones sistémicas integrales para automatizar y optimizar procesos de negocio de acuerdo con las necesidades de la organización.
7. Gestionar proyectos informáticos, ofreciendo alternativas para la toma de decisiones de acuerdo con los requerimientos de la organización.” (Ingeniería En Informática – Duoc UC, 2024)

## Fundamentación Proyecto APT

### Relevancia del proyecto APT

DeLaCruz Lavandería presenta un proyecto que tiene como objetivo reducir el desecho de pecheras plásticas que se utilizan en las diferentes empresas de rubro salmonero u otras relacionadas a los alimentos en las que se usan unas pecheras que son desechables, por esto la lavandería crea una pechera que se puede reutilizar.

Para lograr un buen manejo de sus pecheras se desarrolló un software que les permite registrar cada una de estas, tener un seguimiento adecuado de la duración, de los centros de trabajos vinculados a las pecheras y una vista ecológica que les permite ir visualizando como la reducción de plástico ayuda al medio ambiente o perjudicar dependiendo el punto de vista y los parámetros que se le den.

Gracias a este sistema llamado “Software DeLaCruz” facilitará inmensamente la tarea de trazabilidad y manejo de la información.

### Descripción de Proyecto APT

Software DeLaCruz consiste en un sistema que permite el registro de los datos de las pecheras por medio de la lectura RFID que se puede realizar gracias a un hardware desarrollado con Arduino el objetivo es poder almacenar datos como el UID (identificador único), fecha de fabricación, la talla, último lavado, cantidad lavados, observaciones (cualquier dato que pueda ser relevantes al seguimiento de duración), centro de trabajo, parámetros e índice microbiológico. Por otro lado, se registran los lavados que tiene cada una con el objetivo de saber la duración de estas mismas, el registro de empresas (centros de trabajo) y usuarios.

Este sistema fue desarrollado con Node.js para el backend, React en el frontend, MySQL para las bases de datos y el hardware se creó con Arduino MKR WIFI 1010 y un lector RC522 programado en Arduino IDE en lenguaje C++.

### Pertinencia del proyecto con el perfil de egreso

El desarrollo del software se alinea perfectamente con el perfil de egreso de la carrera “Ingeniería en Informática” dado que nos permite aplicar los conocimientos en:

1. Levantamiento de requerimientos: Se hicieron distintas reuniones en las cuales se recogieron los requerimientos del cliente para poder realizar el trabajo en base a ellos.
2. Ofrecer soluciones informáticas: Se realizaron reuniones en las que se plantearon los problemas y se buscaron las mejores soluciones informáticas para mejorar sus procesos actuales que les retrasaban bastante el avance.
3. Desarrollo de software: Se presenta en el desarrollo del sistema en general, codificación, planificación y realización de pruebas para cumplir con los requerimientos del cliente.
4. Construcción de modelos de datos: Se construyó el modelo de datos para mantener un correcto almacenamiento y manejo de estos.
5. Programar consultas o rutinas: La programación de consultas y rutinas fue necesaria para el correcto funcionamiento de la base de datos en conjunto con las solicitudes del sistema.
6. Integración de tecnologías: Se integraron variadas tecnologías con el fin de lograr un producto que pueda satisfacer las necesidades del cliente.

### Relación con los intereses profesionales

Pilar Bonnault

Este proyecto se alinea bastante bien con mis intereses profesionales dado que mi enfoque va en el desarrollo de software fullstack, la creación de este software me permite aplicar mis habilidades técnicas en un entorno real aplicando mis conocimientos en el diseño, funcionalidades y gestión de un sistema, por otro lado, pude aplicar lo aprendido en Arduino y llevarlo a un entorno profesional.

El liderar este proyecto fue una experiencia muy importante dado que uno de mis grandes objetivos es poder ser Scrum Máster y esto me permite fortalecer mis habilidades y tener conocimientos de otras, una de las más importantes es la toma de decisiones y el manejo del trabajo en equipo, saber apoyar y entender cómo funciona y trabaja cada integrante.

Software DeLaCruz me dio la oportunidad de obtener experiencia y poder ofrecer soluciones informáticas innovadoras con base y conocimientos.

Nicolás González

Dentro de los intereses personales más fuertes están el desarrollo de sistemas web y la ciberseguridad, ya que dentro de lo visto en la carrera fueron las áreas que más se acercan a mi visión de trabajo, la cual busca entregar un producto de buena calidad y, sobre todo, que sea seguro al cliente.

En este proyecto pude desarrollar de mejor manera además descubrir nuevas tecnologías sobre el desarrollo de software, lo que me ayudó a confirmar con mayor fuerza que estos son mis intereses dentro del ámbito laboral, teniendo una responsabilidad sobre todo con el desarrollo frontend del sistema creado, me permitió de igual manera tener un acercamiento más fiel al mundo laboral y como es el trato con gente al momento de tener que realizar un proyecto cosa que siento, favoreció mis habilidades interpersonales.

Este trabajo fue de gran utilidad para tener un crecimiento como persona y como profesional, ya que como fue mencionado anteriormente, me adapté al formato de trabajo con un cliente, en este caso una empresa por lo que fueron varios usuarios. Además del ámbito profesional, este proyecto permitió desarrollar de mejor manera teniendo que crear un software en equipo y con tecnologías que antes no conocía por lo que estoy bastante satisfecho con la experiencia.

Christofer Ruiz

Los intereses que yo tengo es ser un programador completo, pero con un mayor interés en las bases de datos y lo relacionado a estas, como el backend de algún proyecto.

Dentro del proyecto pude realizar todo el apartado de la base de datos, algo que me ayudó mucho a mejorar la forma de administrarlas sumado a que tuve la oportunidad de ocupar otro tipo de base de datos que no había usado todavía, qué es MySql, aquí pude aprender otro tipo de lenguaje de consulta, logrando aumentar mis conocimientos sobre los diferentes tipos de lenguaje.

Además de crear y administrar la base de datos, trabajé en el backend, algo que nunca había hecho y me ayudó bastante en entender cómo manejar los datos en una base de datos, ya que tuve que gestionar el cómo mostrar los datos dentro del software mediante consultas en la base de datos.

El proyecto en general me gusto bastante ya que pude desarrollar y mejorar más mis conocimientos de los intereses laborales que tengo.

### Fortalezas y debilidades para desarrollar el proyecto APT

Fortalezas: El equipo cuenta con integrantes los cuáles se complementan bastante bien, siendo capaces de cubrir las necesidades del otro en caso de ser necesario. Los integrantes del equipo dentro del área de desarrollo se unifican teniendo cubiertas las áreas de bases de datos, frontend y backend, teniendo todos ellos la capacidad de adaptación ante circunstancias de último momento aprendiendo nuevas tecnologías o siendo capaz de tratar situaciones.

Debilidades: Dentro de las debilidades del equipo podemos ver la clara falta de experiencia en el ámbito del desarrollo, a pesar de lograr un buen producto. Junto a esto, podemos encontrar la organización que es buena, pero tiene falencias, como la constancia en horarios para trabajar.

## Planteamiento del problema

El problema principal es que De La Cruz Lavandería desea tener el seguimiento de la vida útil de las pecheras de manera que se lleven los datos de las cantidades de lavados que tiene cada una de estas, además de ser categorizadas dependiendo el daño que ha recibido esta misma.

## Objetivos

### Objetivo General

El sistema "Software DeLaCruz" está diseñado para la empresa De La Cruz lavandería que se encuentra en la ciudad de Puerto Montt, Región de Los Lagos.

Tiene como objetivo gestionar la información de las pecheras que se asignan a distintos centros de costos, permitiendo un seguimiento detallado de las mismas, ya sea que se encuentren en taller, en proceso de lavado o en las plantas correspondientes.

Estas son registradas por medio de lectura RFID, lo cual busca que sea eficiente y rápido a la hora de realizar este proceso.

### Objetivos Específicos

1. Desarrollar una base de datos para el almacenamiento de información y realizar consultas de datos utilizando MySQL
2. Desarrollar interfaz gráfica de usuarios intuitiva y fácil de usar para facilitar el entendimiento del sistema y rapidez en el uso en la que se puedan generar informes históricos de las pecheras registradas
3. Integrar la tecnología RFID en el software para realizar los registros de las pecheras de forma continua y poder identificar fácilmente a qué empresa están destinadas, por otro lado, tener conocimiento de la cantidad de lavados que tiene cada una y poder hacer seguimiento de la duración de estas.

# DISEÑO METODOLÓGICO

El enfoque metodológico que se utilizó para desarrollar este proyecto fue la metodología ágil, específicamente scrum, “ya que se utiliza en proyectos que deben ser desarrollados a corto plazo o los requerimientos no están bien definidos, además se dividen en pequeñas partes denominadas sprint y se enfoca en el trabajo en equipo.” (https://www.facebook.com/APDasociacion, 2024)

| Identificación de Scrum Master y Stakeholders | |
| --- | --- |
| Product Owner | De La Cruz Lavandería |
| Scrum Master | Pilar Bonnault |
| Stakeholders | Angela González  Johanna Lizana  Bárbara Morales  Felipe Tapia |
| Scrum Team | Pilar Bonnault  Nicolás González  Christofer Ruiz |

Fuente: Elaboración propia.

Diagrama del diseño metodológico.

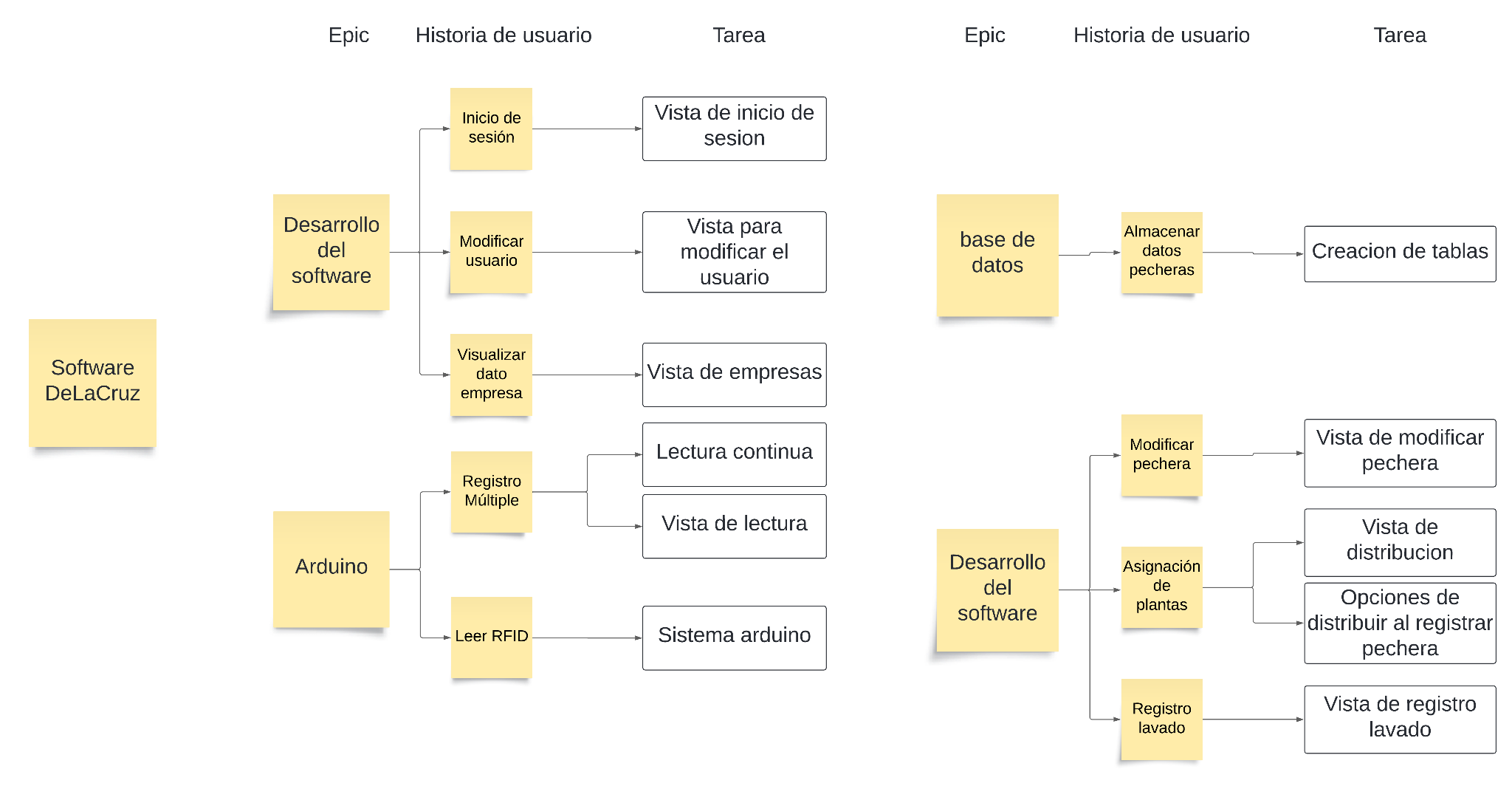
|  | DISEÑO METODOLÓGICO | |  |
| --- | --- | --- | --- |
| Objetivos específicos | Etapas | | Actividades |
| 1.- Realizar la toma de requerimientos del cliente.  2.- Diseño previo (mockup). | N°1 | Captura de requerimientos | Reuniones, Maquetado |
| 1.- Desarrollar interfaz gráfica de usuarios intuitiva y fácil de usar para facilitar el entendimiento del sistema y rapidez en el uso.  2.- Generar informes tanto históricos como filtrados por fechas. | N°2 | Desarrollo de software | Programación de los módulos del software, conexión con API’s |
| 1.- Integrar la tecnología RFID en el software para realizar los registros de las pecheras de forma continua.  2.- Crear un hardware personalizado para poder realizar la lectura de los dispositivos RFID. | N°3 | Arduino | Elección de componentes a utilizar, conexión de componentes, programación de dispositivo, modelado e impresión 3D |
| 1.- Desarrollar una base de datos para el almacenamiento de información y realizar consultas de datos. | N°4 | Creación de Base de datos | Programación de las tablas, datos y disparadores necesarios |
| 1.- Realizar testeo de los módulos del sistema al igual que del hardware. 2.- Corrección de errores. | N°5 | Fase de pruebas | Pruebas de rendimiento, solución de errores |
| 1.- Entrega preliminar para la marcha blanca.  2.- Soluciones de errores en base a feedback.  3.- Entrega final. | N°6 | Despliegue | Entrega y marcha blanca, solución de errores, entrega final. |

Fuente: Elaboración propia.

Para poder definir cuánto tiempo se le dedicará a cada tarea, se decidió hacer una carta Gantt, en la cual se definieron las tareas, los responsables y la cantidad de tiempo para cada tarea del proyecto. [Anexo Carta Gantt](#_heading=h.ihv636)

Para poder llevar un mejor orden y control de las tareas del proyecto que deben ser realizadas se realizó un Sprint Backlog, en el cuál podemos ver las horas estimadas y de avance por cada tarea y responsable, además de esto, una descripción en palabras del cliente por cada tarea. [Anexo Sprint Backlog](#_heading=h.32hioqz)

Alcance de proyecto



##### Figura 1: Elaboración propia

## ETAPA N°1

### Requerimientos del cliente

Se realizó una reunión con los stakeholders donde nos dieron a conocer los requerimientos del proyecto y que es lo que querían conseguir con esto.

**Descripción General**

El software, se podrá usar de forma local, lo que solo permitirá el uso en el dispositivo que quede habilitado para esto, lo que evitará problemas de seguridad por uso en múltiples aparatos. Este sistema aborda todos los problemas actuales de la empresa como la lectura continua para agilizar el proceso de registro, asignación de centros de costos e informes detallados.

Perspectiva del Producto

La plataforma está diseñada para ayudar al cliente a mejorar la organización, facilitando el manejo de la información de las pecheras y el seguimiento de su estado. Los usuarios tendrán acceso a diferentes vistas que les permitirá consultar la disponibilidad de pecheras, así como realizar asignaciones o registrar lavados, por otro lado, poder registrar centros de costos. Además, el sistema contará con una interfaz para que las empresas clientes puedan visualizar la información de las pecheras que se les han asignado previamente.

**Funciones del Producto**

El sistema permitirá una administración intuitiva y sencilla de las pecheras, facilitando tareas como el registro, asignación, eliminación y modificación del estado de las pecheras según sea necesario. También se podrá registrar los lavados y emitir informes detallados sobre el uso de cada pechera. Para las empresas clientes, el sistema proporcionará una interfaz que permitirá iniciar sesión, solicitar la asignación de pecheras y anular solicitudes de manera ágil y eficiente.

**Restricciones**

La empresa no impone ninguna restricción.

**Requisitos Futuros**

El sistema se podrá ir optimizando según el flujo que tenga la página, además de una revisión de seguridad constante. Dependiendo del flujo que tenga la página se podrá evaluar el cambio de hosting a uno que vaya estando acorde a sus necesidades. Además, se analizará la seguridad del sitio y se irá mejorando en el caso de alguna vulnerabilidad.

| Requerimientos del cliente | | | | | |
| --- | --- | --- | --- | --- | --- |
| [R-N°] | [Nombre del Requerimiento] | Tipo Requerimiento  [Funcional, No Funcional] | Actores Relacionados | [Descripción corta del requerimiento] | Estado |
| R.1 | Lectura Continua | Funcional | Hardware, Personal de lavandería, Jefa de operaciones | Lectura constante, sin necesidad de tener alguna interacción con el software | Activo |
| R.2 | Registro de pecheras | Funcional | Sistema, Hardware, Personal de lavandería, Jefa de operaciones | Registrar nuevas pecheras en el sistema | Activo |
| R.3 | Modificación de datos de pecheras | Funcional | Sistema, Personal de lavandería, Jefa de operaciones | Permitir modificar ciertos campos de las pecheras ya registradas | Activo |
| R.4 | Eliminación de pecheras | Funcional | Sistema, Personal de lavandería, Jefa de operaciones | Eliminar las pecheras que ya no se usarán en la empresas | Activo |
| R.5 | Asignación por plantas | Funcional | Sistema, Personal de lavandería, Jefa de operaciones | Asignar plantas a las pecheras nuevas | Activo |
| R.6 | Lectura de los chips realizada de forma individual | Funcional | Sistema, Hardware, Personal de lavandería, Jefa de operaciones | Lectura de los chips de forma manual, en caso de ser necesario encontrar información de una pechera en particular | Activo |
| R.7 | Registro de lavados | Funcional | Sistema, Hardware, Personal de lavandería, Jefa de operaciones | Registrar los lavados de las pecheras | Activo |
| R.8 | Registrar empresas | Funcional | Sistema, Personal de lavandería, Jefa de operaciones | Formulario de registro para nuevos centros de trabajo que se integren a la lavandería | Activo |
| R.9 | Visualización de la información | Funcional | Personal de lavandería, Jefa de operaciones | Vistas que muestran la información almacenada sobre pecheras, empresas y usuarios. | Activo |
| R.10 | Parámetros | No Funcional |  | Sin información | Pendiente |
| R.11 | Migración de base de datos | No Funcional |  | Proceso integrar la base de datos a una nueva versión sin pérdida de datos. | Activo |
| R.12 | Vistas para las empresas que tienen en uso las pecheras | Funcional | Sistema, Personal de desarrollo | Crear vistas para que las empresas puedan ver el estado de sus pecheras asignadas. | Pendiente |
| R.13 | Informes mensuales y anuales (descargar) | Funcional | Sistema, Personal de lavandería, Jefa de operaciones | Permitir la descarga de informes mensuales y anuales | Activo |
| R.14 | Filtros de información por empresa | Funcional | Sistema, Personal de desarrollo | Agregar filtros para visualizar la información según la empresa seleccionada | Activo |
| R.15 | Nuevo diseño (Organización de la información) | No Funcional |  | Implementar un nuevo diseño para mejorar el uso y organización de la información | Activo |
| R.16 | Reutilización de chips | Funcional | Sistema, Hardware, Personal de lavandería, Jefa de operaciones, Equipo de desarrollo | Poder utilizar los chips que ya han sido usado anteriormente sin perjudicar la visualización de la información | Activo |
| R.17 | Conservar datos históricos (Pecheras eliminadas) | No Funcional |  | Guardar registros históricos de pecheras que han sido eliminadas para futuras consultas | Activo |

| Desarrollo ágil: Historias de usuario y criterios de aceptación | | | | | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Enunciado de la historia | | | | Criterios de aceptación | | | |
| Identificador (ID) de la historia | Rol | Característica / Funcionalidad | Razón / Resultado | Número (#) de escenario | Criterio de aceptación (Título) | Contexto | Evento | Resultado / Comportamiento esperado |
| H1 | Como trabajador de la empresa | Quiero tener lectura continua | Para poder registrar múltiples pecheras a la misma vez | 1 | Que se registren múltiples pecheras a la vez | Al momento de ingresar a la vista registro pechera | Se lee y registra pechera | Ver las pecheras agregadas en la vista "Pecheras" |
| H2 | Como trabajador de la empresa | Necesito modificar datos pecheras u eliminar | Para gestionar la información de las pecheras | 1 | Almacenar una pechera | En la vista "Ingreso pechera" | El usuario ingresa los datos y selecciona agregar pechera | La pechera es registrada y aparece en la vista "Datos pecheras" |
| H3 | Como trabajador | Necesito asignar plantas a las pecheras sin asignar | obtener los datos de las pecheras sin asignar y definir una planta | 1 | Asignar plantas | En la vista de distribución | El usuario asigna una cantidad y una planta | Las pecheras se asignan correctamente y los valores se actualizan |
| H4 | Como trabajador de la empresa | Necesito poder visualizar la las empresas registrada previamente para poder designar la planta de destino | Para designar la planta de destino | 1 | Mostrar empresas | En la vista de distribución | El usuario selecciona una categoría | La categoría se muestra correctamente en la vista |
| H5 | Como administración | Almacenar la información de las pecheras | Almacenar la información de las pecheras | 1 | Registro de datos | En el sistema de almacenamiento (Bases de datos MySql) | Se registran todos los datos de pecheras | La información se guarda correctamente para futuras consultas |
| H6 | Como usuario del sistema | Iniciar sesión en la aplicación | Para acceder a las funcionalidades del sistema | 1 | Iniciar sesión | En la pantalla de login | El usuario introduce credenciales | El usuario es autenticado y redirigido a la pantalla principal |
| H7 | Como trabajador de la empresa | Necesito una vista para modificar usuario | Para gestionar la información de los usuarios | 1 | Modificar datos de usuario | En la vista "Modificar usuario" | El usuario edita los datos de un usuario | Los cambios se guardan y se actualizan en la vista de datos de usuarios |
| H8 | Como trabajador de la empresa | Necesito un dispositivo en el que se pueda leer el chip RFID de las pecheras | Para poder visualizar, modificar y registrar pecheras | 1 | Leer el chip RFID de la pechera | Al acercar una pechera al dispositivo | El dispositivo lee el chip RFID | La información de la pechera se guarda o muestra de acuerdo al contexto. |
| H9 | Como trabajador de la empresa | Necesito poder registrar los lavados de cada pechera y poder visualizarlo más tarde | Registrar la cantidad de lavados | 1 | Se lee el chip en la vista lavados y se le suma uno a la pechera | Al acercar una pechera al dispositivo | El dispositivo lee el chip RFID y muestra el valor incrementado | La información del lavado se guarda exitosamente |

## ETAPA N°2

**Herramientas y librerías para el desarrollo de software**

Para el desarrollo de este sistema se buscaron las mejores herramientas y librerías a criterio del equipo para poder entregar al cliente el mejor producto posible, apostando por tecnologías modernas y que nos permiten dar un resultado atractivo, sencillo y funcional.

**Herramientas utilizadas**

1. Visual Studio Code: Es un editor de código fuente ligero pero eficaz que se ejecuta en el escritorio. Incluye compatibilidad integrada con JavaScript, TypeScript y Node.js, y cuenta con un amplio ecosistema de extensiones para otros lenguajes y entornos de ejecución (como C++, C#, Java, Python, Go, .NET). (Visual Studio: IDE Y Editor de Código Para Desarrolladores de Software Y Teams, 2020).
2. NodeJS: Es un entorno de ejecución usado para ejecutar código JavaScript en los servidores. Permite ejecutar JavaScript sin necesidad de un navegador web. (Andrus, 2024).
3. Git: Es un sistema de control de versiones distribuido, lo que significa que un clon local del proyecto es un repositorio de control de versiones completo. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. (mijacobs, 2023).

**Librerías instaladas**

1. ReactJS: Es una de las librerías más populares de JavaScript para el desarrollo de aplicaciones móviles y web. Creada por Facebook, React contiene una colección de fragmentos de código JavaScript reutilizables utilizados para crear interfaces de usuario (UI). (Deyimar A, 2020).
2. Fortawesome/free-solid-svg-icons: Es una librería que proporciona iconos sólidos gratuitos de Font Awesome. (Esteban, 2024)
3. Fortaweosme/react-fontawesome: Es una librería que permite la integración de Font Awesome con la aplicación. (Esteban, 2024)
4. SerialPort/parser-redline: Esta librería permite leer datos desde un puerto serie, usando un protocolo readline. (dotnet-bot, 2024)
5. Axios: Esta se usa para poder realizar las peticiones HTTP, específicamente se usó para llamar a la API’s(Get, Post, Put), esto simplifica la comunicación entre el frontend y el backend. (Gonzales, 2023)
6. Bootstrap: Es un framework css que se utiliza para desarrollar interfaces responsivas y estilizadas. (Santander Universidades, 2023)
7. MySQL2: Se utiliza para interactuar con la base de datos MySQL desde Node.js. (Llamas, 2024)
8. React-bootstrap: Sirve para poder utilizar componentes React en Bootstrap. (https://www.facebook.com/grokkeepcoding, 2023)
9. React-router-dom: Esta librería permite implementar el enrutamiento en la aplicación. (Cabrera, 2024)
10. SerialPort: Se utiliza para leer y escribir datos desde un dispositivo externo, en este caso arduino. (Serial \ Libraries \ Wiring, 2024)
11. Xlsx-style: Se utiliza para trabajar archivos en formato Excel. (Xlsx-Style, 2016)

**Desarrollo del software**

Para la creación del proyecto se utilizaron ciertos comandos, comenzando con “npx create-react-app nombre-del-proyecto” en la consola (CMD), para crear el proyecto como tal y darle el nombre, entramos al proyecto con “cd nombre-del-proyecto” y podemos iniciarlo con “npm start” para poder dar paso a la programación del sistema.

**Estructura Front-End**

Esta es la parte visible de la aplicación para el usuario, en la que se podrá interactuar y hacer todo lo que el cliente desee. (“Frontend Y Backend: Qué Son, En Qué Se Diferencian Y Ejemplos,” 2022)

La parte de la interfaz de usuario se divide en:

Componentes

Los componentes son principalmente elementos que le agregan funcionalidad al sistema y que se utilizan en todo el proyecto, sean visibles o no para el usuario.

1. CustomModal: Nos brinda la opción de tener un pop-up en unas páginas específicas, para acceder a unas funciones de manera más amigable.

2. NavBar: Este nos permite darle identidad al sistema, mostrando el logo y nombre de la empresa en este caso, haciendo más amigable el software.

3. PrivateRoute: A pesar de no ser visible, este componente es bastante importante ya que nos permite darle la función al sistema de NO permitir a un usuario navegar sin haber iniciado sesión con credenciales correctas.

4. sideBar: Este componente nos da la herramienta de la barra lateral con los accesos directos deseados, para darle facilidad al usar el sistema al usuario.

Pages

En esta carpeta se utiliza para almacenar las vistas principales que se irán desarrollando a lo largo del todo el proyecto, cada vista debe ir dentro de una carpeta con un nombre que las identifique y dentro de ellas un archivo index.js en el que se programara la vista. (Estructura de Archivos – React, 2021)

A continuación, se dará una breve explicación de cada una de las vistas que fueron creadas.

**Dashboard**

El objetivo de esta vista es mostrar un resumen de las pecheras registradas históricamente, el total de lavados, las pecheras en uso y las que se encuentran en taller, además de un gráfico anual con un resumen de registros realizados por mes, todo esto distribuidos dentro de cards. Por otro lado, hay un resumen de las últimas 10 pecheras agregadas con una opción para descargar un archivo Excel y tener el conocimiento de los lavados totales realizados por prenda.

¿Cómo funciona?

Esta vista, solicita traer datos de la base de datos mediante las API’S para poder mostrar la información, dichas API’S utilizan sentencias de selección de datos uniendo distintas tablas para la obtención de estos mismos. La vista Dashboard, utiliza las siguientes API’S para el llamado de los datos:

cantidadpecheras: Realiza una consulta a la base de datos para obtener la suma total de registros de dos tablas: pechera e historial.

pecherashistorial: Realiza una consulta a la base de datos para contar los registros en la tabla historial relacionados con la tabla pechera mediante un LEFT JOIN.

pecheras: Obtiene una lista de pecheras con información detallada, como último lavado, cantidad de lavados, planta asociada y observaciones. Permite filtrar por talla si se especifica.

cantidadpecherasmes: Calcula la suma total de registros en las tablas pechera e historiales correspondientes al mes y año actuales.

pecheraenuso: Cuenta las pecheras en uso asociadas a plantas con nombre válido, es decir, aquellas que están registradas y tienen un nombre asignado.

pecherasdisponibles: Cuenta las pecheras que no están asociadas a ninguna planta (es decir, tienen id\_planta como NULL).

fechapecheras: Cuenta el número de pecheras registradas por mes, agrupadas por el nombre del mes de su fecha de registro.

pecherasxmes: Obtiene la cantidad de pecheras registradas por mes, combinando datos de las tablas historial y pechera, filtradas por el año proporcionado. Los resultados se agrupan por mes y año.  
cantidadlavados: Cuenta el total de registros en la tabla lavado, que representan los lavados de pecheras registrados en la base de datos.

**DatoPechera**

Aquí se muestran todos los datos de las prendas registradas, tales como, UID, fecha fabricación, talla, fecha de último lavado, cantidad de lavados, observaciones, centro de trabajo, parámetros, índice microbiológico y dos botones que permiten la modificación de algunos datos o eliminación de la pechera. Además, hay un filtro que permite buscar la información por empresa o fecha lavado lo que permite ver más claramente la información deseada y poder descargar esta misma, por último, dos botones que nos permiten la modificación de los datos de las pecheras, pero por medio de la lectura RFID y evitar la búsqueda manual y otro botón para distribuir la prenda en caso de que no esté asignada a ningún centro de trabajo.

Flujo de trabajo:

1. Primero se deben solicitar los datos a la API para poder visualizar la información deseada
2. El usuario puede aplicar filtros si lo desea para poder personalizar los datos que se están mostrando
3. Se puede visualizar información específica de una pechera, editar información, distribuir y eliminar, todo por medio del uso del dispositivo
4. Cambiar entre varias vistas según lo que el personal requiera hacer.

¿Cómo funciona?

Primeramente, se deben realizar unas importaciones que como “useState” y “useEffect” que se utilizan para manejar estados y otros efectos, por otro lado, se importa XLSX y jsPDF que se utiliza para poder descargar los informes de las pecheras tanto en PDF como en Excel.

Filtrado de información: esta vista cuenta con dos filtros que nos permite poder obtener información de acuerdo con las empresas o fecha de lavado.

Paginación: La información se organiza de tal forma que solo se muestran 10 datos por página para mantener un orden y claridad. Cuenta con los botones necesarios para la navegación entre las páginas.

Botones de navegación: para poder navegar entre vistas se utiliza “useNavigate” que no permite el poder cambiar de vista de forma simple y rápida, a continuación, se presentan los botones que nos permiten realizar eso:

Información pechera: Esta nos redirige a una vista que nos permite ver la información de una pechera específica por medio de la lectura RFID

Distribución pecheras: Nos redirige a una vista donde se pueden distribuir las pecheras que se encuentran en stock.

Modificar pecheras: Nos envía a una página que nos permitirá modificar la información de las pecheras.

Eliminar pechera: Elimina las pecheras que fueron leídas con el dispositivo.

Esta vista incluye un manejo de errores en caso de problemas para mostrar información o usar los filtros.

**DistribucionPecheras**

Esta vista está diseñada para poder distribuir las pecheras que se encuentran en stock en la lavandería, esto se puede realizar utilizando el dispositivo para realizar la lectura de los chips e identificar las que quieren ser distribuidas a los centros de trabajo y así poder actualizar su información correctamente.

¿Cómo funciona?

Cuando ingresamos a la vista se realiza una petición a la API para poder compilar los datos de los centros de trabajo y se guarda en el estado de las empresas.

Lectura:

Para llevar a cabo la lectura se debe utilizar un hook llamado “useEffect” que permite definir un intervalo que realiza una consulta a la API, específicamente al endpoint “latest-uid” cada segundo para poder obtener el ultimo UID que se leído, una vez hecho esto se comprueba que no esté duplicado y se muestra en la vista en el apartado “UID’s Leídas”

Actualización de datos:

A continuación, el usuario debe escoger el centro de trabajo en el menú desplegable y utilizar el botón “Actualizar pecheras” esto validará los UID y la empresa para que sean enviados el servidor a través de una petición “”Put” que se encuentra en la API (“modificarplantapechera”).

Manejo de errores:

Si el proceso se realiza exitosamente se despliega un pop up que nos muestra un mensaje de confirmación y redirige a la vista “DatosPecheras”, en caso contrario nos muestra un mensaje de error para dar a conocer al usuario que hubo un fallo.

Y por último existe un botón que permite limpiar la información de la pantalla.

**EcoVista:**

Esta vista busca mostrar el impacto que tiene el usar pecheras reutilizables al largo de los años utilizando datos simulados que nos permiten tener una proyección amplia.

¿Cómo funciona?

El usuario puede seleccionar entre pecheras reutilizables y desechables, de acuerdo con lo elegido se deben introducir los kilos de plástico y los años de proyección, por otro lado, pueden agregados datos más específicos como las tasas de crecimiento del CO₂ y electricidad. Los datos calculados se muestran en 3 tarjetas y un gráfico dinámico.

Para poder calcular esto se utilizaron las siguientes fórmulas:

Cálculo inicial:

1. numUsos =
2. Emisiones totales de CO₂: consumoElectricoTotal = emisionesCO2 x numUsos
3. Consumo eléctrico total: consumoElectricoTotal = cosumoElectrico x NumUsos

Para mostrar los datos se aplican tasas de crecimiento aleatorias para actualizar las tarjetas y el gráfico.

Es importante tener énfasis en que los datos son simulados.

**EliminarPechera**

Esta vista nos da la posibilidad de leer varias pecheras a la vez y poder eliminarlas o darlas de baja, permitiendo un ahorro de tiempo significativo si es que en dado caso se tienen varias pecheras dañadas sin posibilidad de reparación.

Las librerías usadas fueron “axios” para poder realizar las peticiones http, “swal” para poder mostrar pop up visualmente atractivos y por último “useNavigate” que nos permite navegar entre vistas.

Los principales estados son:

pechera: Guarda la lista de datos de las pecheras que se han registrado en el sistema

loading: Indica que los datos se están cargando o esperando ser obtenidos

uid: Almacena el último UID leído

errorMessage: Alacena los mensajes de error.

¿Cómo funciona?

Las principales funciones utilizadas en esta vista para poder cumplir con las tareas que debe realizar son:

fetchUIDLatest: Que permite solicitar el ultimo uid escaneado desde el backend del sistema, en el caso de un error se puede gestionar usando “console.error”

fetchDetailsPecheras(uid): Utiliza el UID para obtener la información exacta de la pechera, en caso de encontrar la pechera verifica que no se encuentre duplicada en la lista que se está mostrando y la agrega en la lista, si no la encuentra abre un pop up con un mensaje de pechera no registrada.

eliminarPechera(id\_pecherar\_registro): Aquí se envía una solicitud DELETE al backend para poder eliminar las pecheras especificadas, si se logra realizar correctamente:

Muestra un pop up que solicita la confirmación del usuario y elimina la pechera y actualiza el id de esta para que se almacene en una tabla de datos históricos, visualmente se muestra un mensaje de confirmación a través del uso de “swal” y si se produce un error muestra otro mensaje.

Una vez realizado esto se limpia la lista del frontend.

**Empresa**

Aquí se visualizan los datos de los centros de trabajo que tienen contrato con DeLaCruz Lavandería y a las que idealmente en un futuro se les entregarán pecheras reutilizables, también se permite modificar o inhabilitar la empresa en caso de dejar de trabajar con el cliente, por último, un botón con acceso directo al registro de nuevas empresas.

¿Cómo funciona?

El funcionamiento de la vista Empresa, principalmente es con el llamado de datos para poder mostrarlos en la página principal de empresa, teniendo ésta distintas subpáginas, las cuales son modificar empresa, donde como su nombre lo indica se pueden modificar los datos de la empresa, teniendo esta página la necesidad de llamar a dos API’s, primeramente a la encargada de mostrar los datos actuales de la empresa o centro de trabajo mediante la id de este centrodetrabajoid, seguidamente de la segunda API actualizarempresa la cual se encarga de actualizar la información en la base de datos, de la empresa seleccionada mediante sentencia SQL.

**Lavado**

En esta vista tenemos la opción de leer varias pecheras para poder ingresar un lavado a cada una, ahorrando tiempo al personal, además, si es que se leyó una pechera que no debía leerse, esta vista da la opción de limpiarla del registro para que no se mezcle o haya algún error de datos.

¿Cómo funciona?

fetchLatestUID: Pide el UID más reciente desde el backend, este UID corresponde al escaneado de una pechera para poder obtener la información.

fetchDetailsPecheras(uid): Utiliza el UID para obtener la información exacta de la pechera, en caso de encontrar la pechera verifica que no se encuentre duplicada en la lista que se está mostrando y la agrega en la lista, si no la encuentra abre un pop up con un mensaje de pechera no registrada.

handleRegistrarLavado: Realiza una petición al backend con la lista de las pecheras, que incluye datos como observaciones, cantidad de lavado, entre otras.

Si el registro tiene éxito, se muestra una ventana con un mensaje de confirmación y se recarga la página.

**Login**

Cómo muchos otros, esta vista solamente nos muestra un inicio de sesión sencillo, en el cual encontramos el campo de “Correo” y “Contraseña” los cuales deben ser rellenados con datos correctos que estén ingresados en la base de datos y que tengan el formato necesario como lo es el correo que solicita que la cadena de texto tenga la siguiente estructura “[texto@texto.cl](mailto:texto@texto.cl)” o “[texto@texto.c](mailto:texto@texto.cl)om” así dando paso al uso del sistema.

¿Cómo funciona?

Este apartado funciona principalmente mediante una validación de los datos insertados en los inputs indicados, mediante las características de React, se añade la funcionalidad de la obligatoriedad en el campo de “Correo” para que este si o si tenga el formato de una dirección de correo electrónico. La contraseña, de igual manera que el correo, es de carácter obligatorio y esta última al ser ingresada al sistema desde la pestaña de “Usuarios” es almacenada con un formato de encriptado SHA2, para aumentar la seguridad del sistema.

**ModificarEmpresa**

En esta vista se permite modificar datos como el nombre, cantidad de pecheras, kilos (plástico ahorrado) y el estado (Activa o inactiva).

¿Cómo funciona?

Cómo se mencionó anteriormente, esta vista funciona mediante el llamado de una API (actualizarempresa) la cual se encarga de realizar la actualización de los datos basándose en la id de la empresa que se desean realizar cambios, de manera que los datos anteriores sean reemplazados por los datos nuevos.

**ModificarPecheras**

Acá se permite la modificación de información como la talla, cantidad de lavados, observaciones, centro de trabajo, parámetros, y el índice microbiológico, pero hay datos como la fecha de fabricación y el UID que no se permiten modificar. Todo esto se puede realizar leyendo el dispositivo RFID con el aparato de lectura.

¿Cómo funciona?

El apartado de ModificarPecheras funciona principalmente de manera que al ser leída la pechera en el dispositivo Arduino, se detecte el ID y se obtengan todos los datos presentados en la vista (ID, Fecha de fabricación, Talla, Observaciones, Centro de trabajo y por último el Índice microbiológico) de los cuáles no puede ser modificado el indicador único y la fecha de fabricación, el usuario debe ingresar los nuevos datos que deben ser actualizados en la base de datos mediante la API modificarpecheras.

**ModificarPecherasSinLeer:**

Esta vista modifica los mismos datos que en la vista anterior a diferencia que se selecciona las pecheras desde la tabla de datos pecheras;

¿Cómo funciona?

En la vista ModificarPecherasSinLeer se encuentran los datos de la pechera seleccionada por medio del panel de la vista Pecheras, para realizar la modificación de los datos de la pechera seleccionada, mediante la selección del id de la pechera, se realiza un llamado a la API (modificarpecheras), para poder realizar la actualización de los datos.

**ModificarUsuario**

En esta vista encontramos un campo menos a diferencia de la vista de RegistroUsuarios, ya que se nos permite cambiar el correo, nombre y empresa, sin posibilidad de cambiar la contraseña de ese usuario por motivos de seguridad del sistema.

¿Cómo funciona?

Al igual que las otras vistas de modificación, se debe primeramente obtener el id asociado al usuario, el cual se busca inmediatamente al presionar el botón modificar de un usuario, dentro de la vista encontramos distintos datos los cuales pueden ser modificables (Nombre, Correo y Centro de trabajo) por temas de seguridad, no se puede modificar la contraseña.

**RegistroEmpresa**

Acá se deben registrar todas las nuevas empresas que comiencen a trabajar con la lavandería, se solicitan datos como; Nombre empresa, Cantidad pecheras solicitadas y los kilos (plástico ahorrado).

¿Cómo funciona?

RegistroEmpresa cumple su función requerida acudiendo a la solicitud de la API registroempresa la cual se encarga de ingresar los datos a la BD mediante el método POST y su consulta SQL permitiendo poder registrar la empresa deseada junto con sus datos.

**RegistroPecheras:**

Esta es una de las vistas más importantes, ya que es aquí donde se registran las pecheras reutilizables por primera vez, aquí se piden datos como la talla que es obligatoria y opcional el centro de trabajo a las que se enviaran. Este registro se realiza por medio de la lectura de un chip RFID que se acerca a un dispositivo preparado para eso y se pueden registrar una cantidad de pecheras ilimitadas siempre y cuando sean de la misma talla.

¿Cómo funciona?

Se están importando librerías para manejar estados y cualquier efecto secundario de la vista entre estas están useState y useEffect, además se importa useNavigate que nos permitirá navegar entre vistas.

Para diseño utilizamos FontAwesomeIcon que nos permite usar una alta variedad de iconos y swal para poder generar alertas personalizadas.

Estados:

En la fracción de código siguiente se crea un objeto pechera que busca almacenar información como talla, que es obligatorio y id\_planta que es opcional,

const [pechera, setPechera] = useState({

talla: '',

id\_planta: ''

});

luego se llama a un endpoint “http://localhost:3000/api/latest-uid” que nos devuelve los UID’s leiodos por el dispositivo Arduino van listando las UID’s leídas en el caso de ser un nuevo UID y que no haya sido leído previamente los agrega a las “uidsLeidas” y actualiza “lastuid”, después se comparan los UID para poder evitar los duplicados en la lista y evitar que se agreguen.

const fetchUIDFromArduino = async () => {

try {

const response = await fetch('http://localhost:3000/api/latest-uid');

if (!response.ok) throw new Error(`HTTP error! status: ${response.status}`);

const data = await response.json();

if (data.uid && data.uid !== lastUid) {

setLastUid(data.uid);

if (!uidsLeidas.includes(data.uid)) {

setUidsLeidas(prevUids => [...prevUids, data.uid]);

}

}

} catch (error) {

console.error('Error al obtener el UID:', error.message);

}

};

También hay manejo de validaciones que nos permiten mostrar mensajes en caso de que los campos obligatorios no hayan sido seleccionados y evitar registros erróneos.

Una vez realizado todo lo necesario se realiza el registro y muestra un mensaje de éxito y nos redirige a “DatosPechera” de caso contrario se realiza en control de errores y no despliega un mensaje de error acorde a la situación.

**RegistroUsuarios**

Como antes fue mencionado, en este apartado podemos ver los campos para ingresar un usuario al sistema, siendo estos campos (Nombre, Correo, Contraseña y Centro de trabajo), para darle acceso al sistema a este usuario. Los usuarios tendrán mayor repercusión en el futuro del sistema ya que se buscará tener un sistema de permisos para que no todos los usuarios tengan acceso a toda la información.

¿Cómo funciona?

El registro de los usuarios funciona de manera que De La Cruz Lavandería ingrese los datos solicitados para que ningún usuario pueda registrarse, esto busca mantener los datos de la empresa lo más seguros y privados posibles. El registro funciona mediante el llamado de la API modificarusuario para realizar la actualización en base al ID.

**Usuarios**

En esta vista se busca mostrar a todos los usuarios que se les permitirá usar el sistema.

¿Cómo funciona?

La vista Usuarios se encarga de mostrar los datos mediante la llamada de estos con una API dedicado (Usuarios) permitiendo esto, la organización visual de los datos de los usuarios, además podemos dentro de esta misma vista, eliminar usuarios, al igual que las otras páginas, la eliminación de datos se debe hacer mediante el llamado de las funciones de la API (eliminarusuarios).

**App.js**

En este archivo, podemos encontrar el enrutado del sistema, definiendo cada una de las rutas del sistema y asignando de igual manera cuáles de estas serán privadas o no, significando esto que no pueden ser accesibles por un usuario que no tenga su sesión iniciada.

¿Cómo funciona?

El archivo App.js gestiona el enrutado de la aplicación utilizando la biblioteca **react-router-dom** para definir las rutas y el componente **PrivateRoute** para proteger ciertas rutas, permitiendo el acceso solo a los usuarios autenticados.

**Index.js**

En este archivo llamamos al uso completo de la aplicación y hacemos que el enrutado sea funcional.

¿Cómo funciona?

El archivo index.js monta <App /> en el DOM dentro del elemento con id="root", habilita el enrutado con <BrowserRouter> y aplica estilos globales como Bootstrap.

**Resources**

En esta sección se almacenan todos los archivos visuales que son usados en el frontend del software, tales como; imágenes, videos, entre otros.

**Styles**

La carpeta Styles, guarda todos los archivos que le dan estilo a cada ‘page’ del proyecto, permitiéndonos darle una personalidad única y hacer que sea mucho más amigable al usuario evitando así una saturación visual y que no sea algo desagradable de usar. Se buscó darle identidad al sistema utilizando los colores corporativos de la empresa, teniendo una gama de colores bastante completa siendo esta utilizada para darle color a gráficos, tarjetas de información, botones, etc.

**Estructura Back-End**

Esta es la parte de la aplicación que corresponde al funcionamiento, que responde a las solicitudes de las vistas del frontend y se comunica tanto con la base de datos como con el dispositivo. Abarca toda la parte lógica de un software. (“Frontend Y Backend: Qué Son, En Qué Se Diferencian Y Ejemplos,” 2022)

Para comenzar a trabajar en el backend se crea una carpeta llamada en este caso “server” en la que se ejecuta un comando “npm init” para poder inicializar el proyecto en node.js y por otro lado se instala Express con el comando “npm install express” para configurar el servidor. (de, 2021)

El backend se divide en 5 puntos importantes:

**Config**

Dentro de esta carpeta se encuentra el archivo “db.js” que configura la conexión con la base de datos MySQL en la que se importan los módulos para soportar las características de MySQL e interactuar con la base de datos desde Node.js y se llama al ‘dotenv’ que importa el archivo “.env”.

Después se realiza la configuración de la conexión, en la que se crea una función para crear la conexión, se definen los parámetros necesarios como el host (dirección de la base de datos), port (Puerto de la base de datos), database (el nombre de la base de datos), user (Nombre de usuario) y password (contraseña para acceder a la base de datos), todos esos valores se definen más tarde en el archivo “.env”.

Por último, se inicia la conexión con la base de datos y se utiliza el manejo de errores en caso de fallas.

**Routes**

Aquí se almacena un archivo denominado “api.js” que se construyó con “express.js” para que podamos realizar las interacciones con la base de datos, donde se almacenan todos los endpoint y las funciones de cada uno para el funcionamiento de cada una de las vistas que requieran el uso de los datos.

**.env**

En este archivo se definen todos los valores necesarios para la conexión con la base de datos y el dispositivo Arduino.

**index.js**

Tal como en otros archivos, aquí se importan las librerías necesarias para poder conectar un puerto serial y obtener datos de los UID leídos.

Acá se realizan las importaciones de los módulos necesarios para el funcionamiento

const express = require('express'); Esta se utiliza para crear el servidor web

const cors = require('cors'); Esta nos facilita las peticiones desde diferentes dominios

const dotenv = require('dotenv'); Aquí se cargan las variables definidas en el archivo “.env”

const apiRoutes = require('./routes/api'); Se llama al archivo donde estan las rutas principales

const { SerialPort } = require('serialport'); Esto nos permite la interacción con dispositivos conectados mediante puertos seriales

const { ReadlineParser } = require('@serialport/parser-readline'); Nos permite analizar los datos

Se debe configurar el servidor para poder ejecutarlo

dotenv.config();

const app = express();

const port = process.env.PORT || 3000;

// Configuración de middleware

app.use(cors());

app.use(express.json());

En este caso se comienza cargando las variables de entorno con “dotenv.config();” que se definieron en el archivo “.env” a continuación se define el puerto 3000 “const port = process.env.PORT || 3000;” es el utilizado por defecto y se llama al “api.js” definida en la siguiente ruta “app.use('/api', apiRoutes)”

Luego se debe realizar la conexión con el puerto serial:

const serialPortPath = process.env.SERIAL\_PORT || 'COM6';

const baudRate = parseInt(process.env.BAUD\_RATE, 10) || 9600;

Se debe definir el puerto serial en el que está conectado el dispositivo “const serialPortPath = process.env.SERIAL\_PORT || 'COM6';” que puede variar de acuerdo con el dispositivo y puerto USB al que se conecta el arduino, por otro lado, se debe definir el baudRate que por defecto queda en 9600, esta es la velocidad de comunicación.

A continuación, se crea la conexión con el puerto especificado y se configura con los valores previamente definidos

const portSerial = new SerialPort({

path: serialPortPath,

baudRate: baudRate

});

Finalmente se procesan los datos y los separa con un salto de línea “\n”

const parser = portSerial.pipe(new ReadlineParser({ delimiter: '\n' }));

Luego se realiza el manejo de los datos obtenidos desde el puerto COM, en este caso los UID’s de los chips, para esto se crea un evento ‘data’ que se activa al momento de recibir datos, se limpian los espacios que contengan los identificadores y se elimina el prefijo ’UID:’ en el caso que exista, luego de eso se almacena junto con la hora de la lectura y por último se creó un endpoint que obtiene el último valor leído como se muestra en el siguiente código:

let latestUID = null;

let lastReadTime = null;

parser.on('data', (data) => {

let uid = data.trim().replace(/\s+/g, ''); // Eliminar espacios

if (uid.startsWith('UID:')) {

uid = uid.slice(4); // Remueve el prefijo "UID:"

}

console.log(`UID procesado: ${uid}`);

latestUID = uid;

lastReadTime = Date.now(); // Actualiza la marca de tiempo

});

Se definieron algunos console.log para comprobar la conexión con el puerto COM, la confirmación de la ejecución del servidor y el manejo de errores en caso de que algo falle.

## ETAPA N°3

**Hardware**

**Herramientas utilizadas**

Arduino IDE: Esta es una herramienta que permite desarrollar el grabar todo el código necesario para el funcionamiento de nuestro Arduino. (Software de Arduino | Arduino.cl - Compra Tu Arduino En Línea, 2014)

TinkerCAD: Esta es una herramienta online que nos permite crear modelos 3D y circuitos electrónicos y realizar las simulaciones, en nuestro caso fue usado para crear el modelo 3D. (Tinkercad. Dando Volumen a Las Ideas - INTEF, 2022)

Cura Slicer: Este es un software utilizado para la impresión 3D, que nos permite configurar parámetros como la velocidad, soporte, relleno, entre otros, para imprimir nuestra caja. (UltiMaker Cura, 2024)

**Librerías utilizadas**

1. MFRC522: Esta librería permite leer y escribir datos en tarjetas RFID. (pablo-sampaio, 2022)

**Descripción y armado**

Para poder registrar las pecheras reutilizables al sistema y poder tener seguimiento de estas se llegó a la conclusión de crear un dispositivo con Arduino y un módulo de lectura.

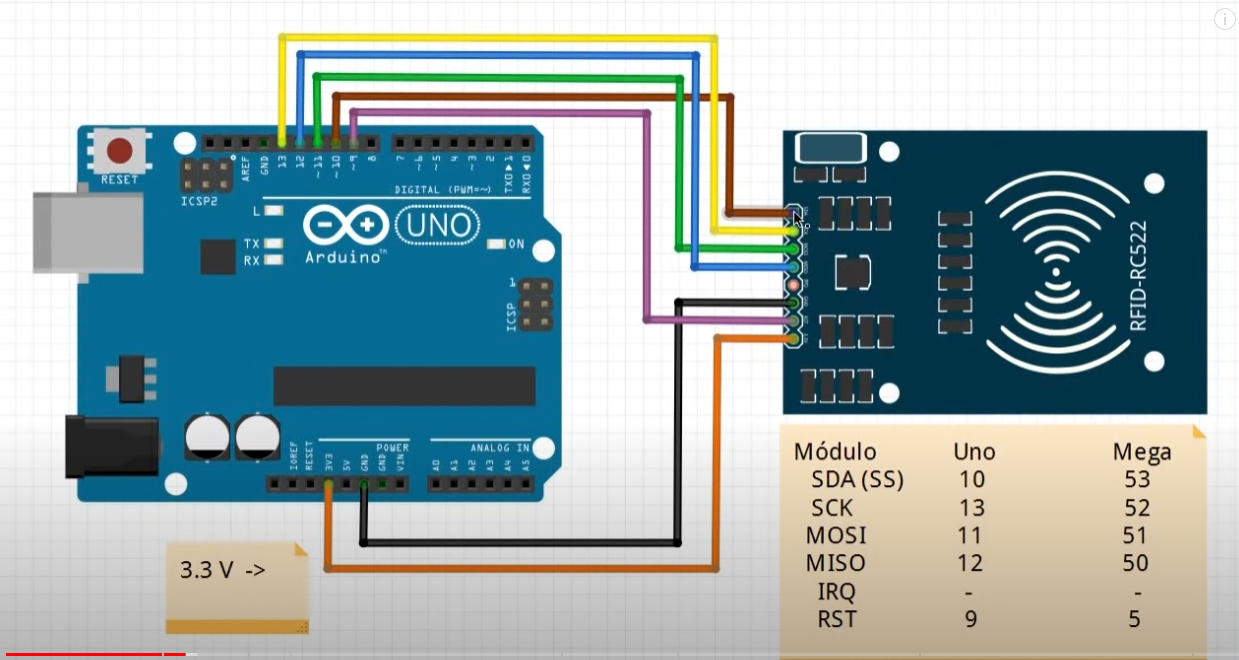
Primero para poder abarcar el tema del hardware se debe entender lo que es un Arduino, esto es una placa de circuito impreso de hardware libre que se incorpora con un microcontrolador reprogramable con una serie de pines que sirven para conectar diferentes módulos, en nuestro caso el RC522 para programar se utilizan un lenguaje de programación basado en C++. (¿Qué es Arduino? | Arduino.cl - Compra Tu Arduino En Línea, 2014)

Hay una gran variedad de modelos de este dispositivo que tienen diferentes microcontroladores, los puertos de entrada y salida, memoria, velocidad de reloj, la conectividad y tamaño. (▷ Modelos de Placas Arduino. Comparativa, Características, Comprar,.., 2020)

En nuestro caso usaremos el **Arduino MKR Wifi 1010** ya que tiene un microprocesador potente y eficiente como es el SAMD21 esto nos da un rendimiento más rápido, y útil a la hora de realizar múltiples tareas, además el bajo consumo eléctrico ya que se tendrá en funcionamiento constantemente, otro punto importante es la seguridad y los protocolos de comunicación que nos facilitan la integración con el software que puede ser tanto alámbrica como inalámbrica. (Electrónica Embajadores, 2022)

Por otro lado, está el módulo RC522 esto es un lector RFID que nos permitirá leer el UID de los chips que tienen las pecheras.

Por otro lado, tenemos que contar con los cables para la conexión de ambos dispositivos que en este caso se utilizaron 7 los cuales fueron soldados el módulo RC522 y conectados al Arduino como se muestra en la ‘Figura 1’.



##### Figura 2: Conexión cables (kit, 2018)

**Programación de Arduino**

Para poder darle el funcionamiento tanto al Arduino como al lector se programó en lenguaje C++ utilizando Arduino IDE, el objetivo es que permite la lectura de los chips RFID entregando el UID en pantalla, a continuación, se explican los códigos paso por paso:

1. **Se deben importar las librerías necesarias para que el Arduino se pueda comunicar con el lector (RC522).**

#include <SPI.h>

#include <MFRC522.h>

La librería SPI.h permite la comunicación serial entre Arduino y el módulo mediante el protocolo Serial peripheral Interface.

La librería MFrc522.h se utiliza para manejar el módulo RC522 que nos entrega funciones para poder inicializar y manejar el lector.

1. **Después se deben definir los pines más importantes que en este caso son el SS\_PIN y RST\_PIN**

#define SS\_PIN 10

#define RST\_PIN 9

SS\_PIN es el pin que selecciona el chip para el SPI, este le indica cuando debe activar el módulo RFID.

RST\_PIN Es el encargado de reiniciar el módulo de lectura en caso de ser necesario.

1. **Se crea la instancia**

MFRC522 mfrc522(SS\_PIN, RST\_PIN); // Crear instancia del MFRC522

El código anterior crea un objeto llamado mfrc522 utilizando la clase MFRC522 y asignado dos parámetros, que en este caso son los pines definidos previamente, lo que nos dará paso a manipular el lector.

1. **Se realiza la configuración inicial**

void setup() {

Serial.begin(9600);

SPI.begin();

mfrc522.PCD\_Init();

configureRF();

}

Dentro de la función setup () se realizan tres pasos:

”Serial.begin(9600);” Es utilizado para iniciar la comunicación y así poder enviar y recibir datos en el monitor serie de Arduino IDE este se inicializa a una velocidad de 9600 baudios (Velocidad estándar para la transmisión de datos),

“SPI.begin();” Inicializa la comunicación SPI

“mfrc522.PCD\_Init();” Inicia el módulo de lectura RC522

configureRF(); Este se utiliza para que podamos maximizar el rango de lectura y poder minimizar las interferencias que puedan existir.

1. Se desarrolla el bucle principal (loop ())

void loop() {

if (!mfrc522.PICC\_IsNewCardPresent()) {

return;

}

if (!mfrc522.PICC\_ReadCardSerial()) {

return;

}

Serial.print("UID: ");

for (byte i = 0; i < mfrc522.uid.size; i++) {

Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");

Serial.print(mfrc522.uid.uidByte[i], HEX);

}

Serial.println();

delay(1000);

}

Lo primero que se realiza en el código anterior es verificar si hay una tarjeta en el rango de lectura del módulo, para esto se llama al método “PICC\_IsNewCardPresen()”,

Si hay una tarjeta presente entonces intenta leer el UID con el método “PICC\_ReadCardSerial()”, en caso contrario termina el loop

Al leer el UID se envía al monitor serie con un “Serial.print("UID: ");” pero se inicia un bucle “for” en el que se analiza el tamaño del UID y da formato para imprimir el valor en formato hexadecimal.

1. Se realiza la configuración para aumentar los rangos de lectura

void configureRF() {

Configurar la potencia de transmisión al máximo

mfrc522.PCD\_WriteRegister(mfrc522.RFCfgReg, 0x70); // 48dB de ganancia

Ajustar el tiempo de retardo para estabilizar la señal

byte modulation = mfrc522.PCD\_ReadRegister(mfrc522.TxControlReg);

modulation |= 0x03; // Encender ambas antenas TX1 y TX2

mfrc522.PCD\_WriteRegister(mfrc522.TxControlReg, modulation);

**Modelo 3D**

Se diseñó un modelo 3D para poder guardar el hardware y quede completamente protegido para esto se utilizó la herramienta TinkerCAD. A continuación, se muestra el diseño (Figura 2, Figura 3 y Figura 4)

##### Figura 3: Elaboración propia Figura 4: Elaboración propia Figura 5: Elaboración propia

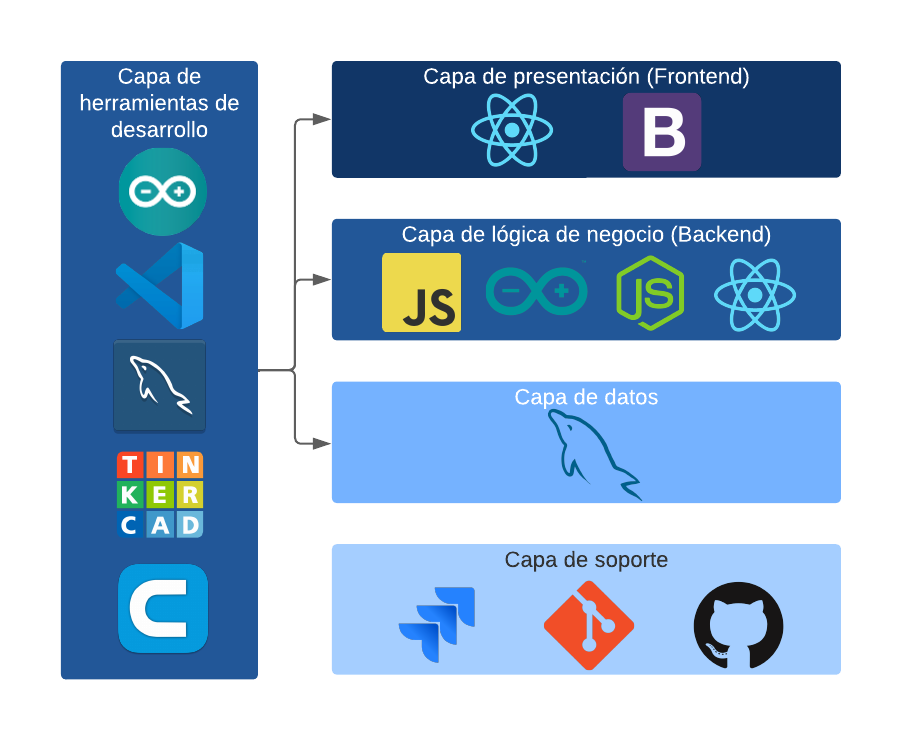
**Impresión 3D**

En este paso básicamente el modelo realizado previamente se lleva a un software utilizado para configurar los diferentes parámetros necesarios para la impresora 3D que se utilizara.

## ETAPA N°4

Arquitectura de software

La arquitectura del sistema “Software DeLaCruz” está basada en la “Arquitectura por capas” la cuál consta en dividir la aplicación en capas, con la intención de que cada capa tenga un rol muy definido, como podría ser, una capa de presentación (Frontend), una capa de lógica de negocio (Backend) y una capa de acceso a datos. (Arquitectura En Capas, 2021)



##### Figura 6: Elaboración propia

**Alcance**

Con el nuevo software la empresa va a poder registrar de forma continua (de 10 o más a la vez), modificar y eliminar las pecheras que necesiten, podrán registrar los lavados que la empresa realice a las pecheras, también se podrá registrar nuevos centros de trabajo, modificarlos y habilitarlo o deshabilitarlos de según sea necesario.

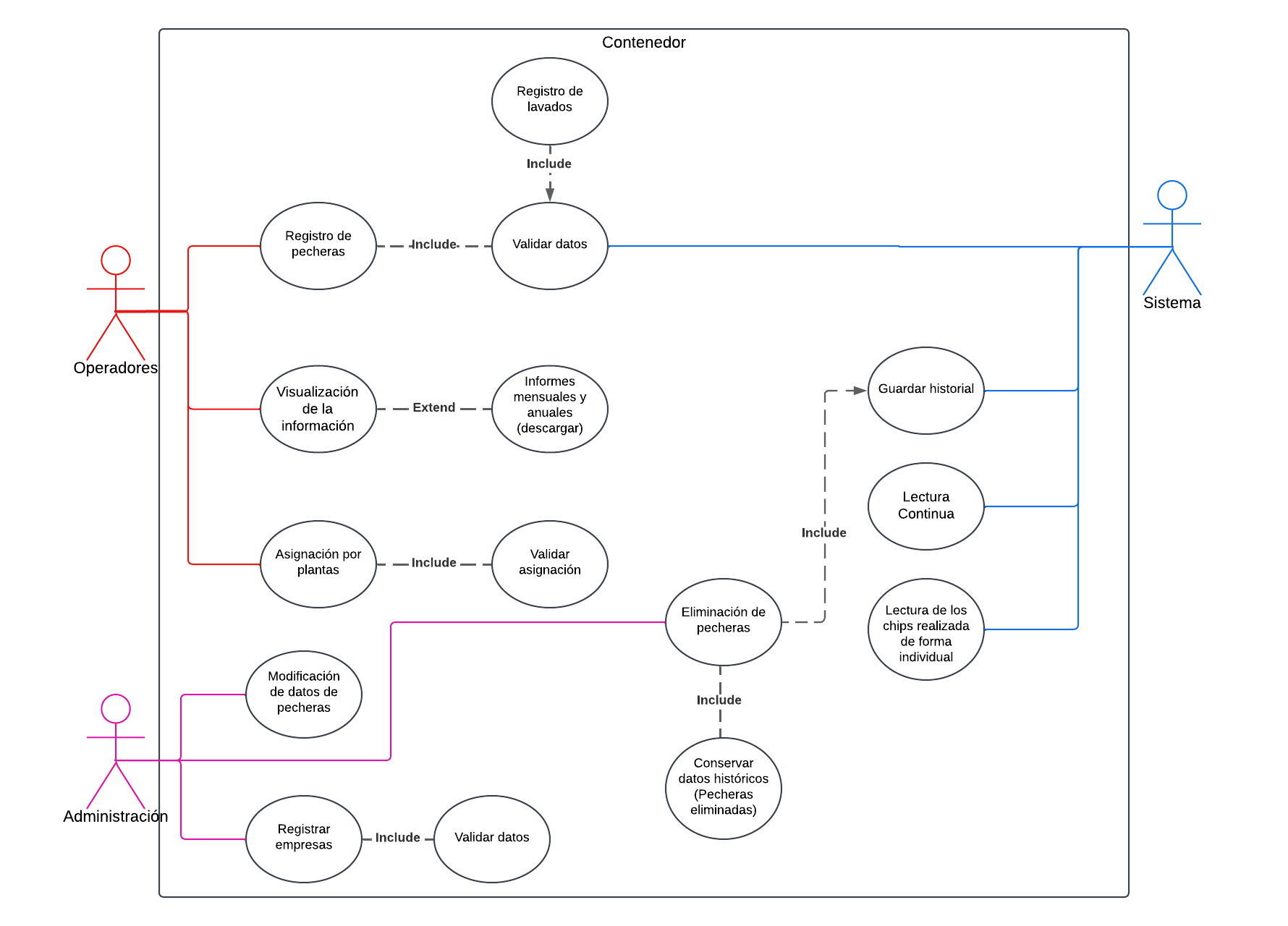
El software va a contar con un sistema de usuarios para poder proteger de mejor manera la información.

El dispositivo de lectura va a ser más robusto y de mejor calidad que el anterior.

**Referencias**

1. Caso de usos
2. Requerimientos funcionales y no funcionales
3. Diagramas de actividad, clases, secuencia, etc.

**Vista de Casos de Uso y Escenarios de Calidad:**



##### Figura 7: Elaboración propia

El actor más importante para este software va a ser el que lleve los registros de las pecheras, en este caso va a ver una persona asignada para el uso de este software

**Actores:**

Operador: Van a ser las principales personas quienes operen el software del área de lavandería.

Administración: Personal encargado de la administración de los datos del sistema.

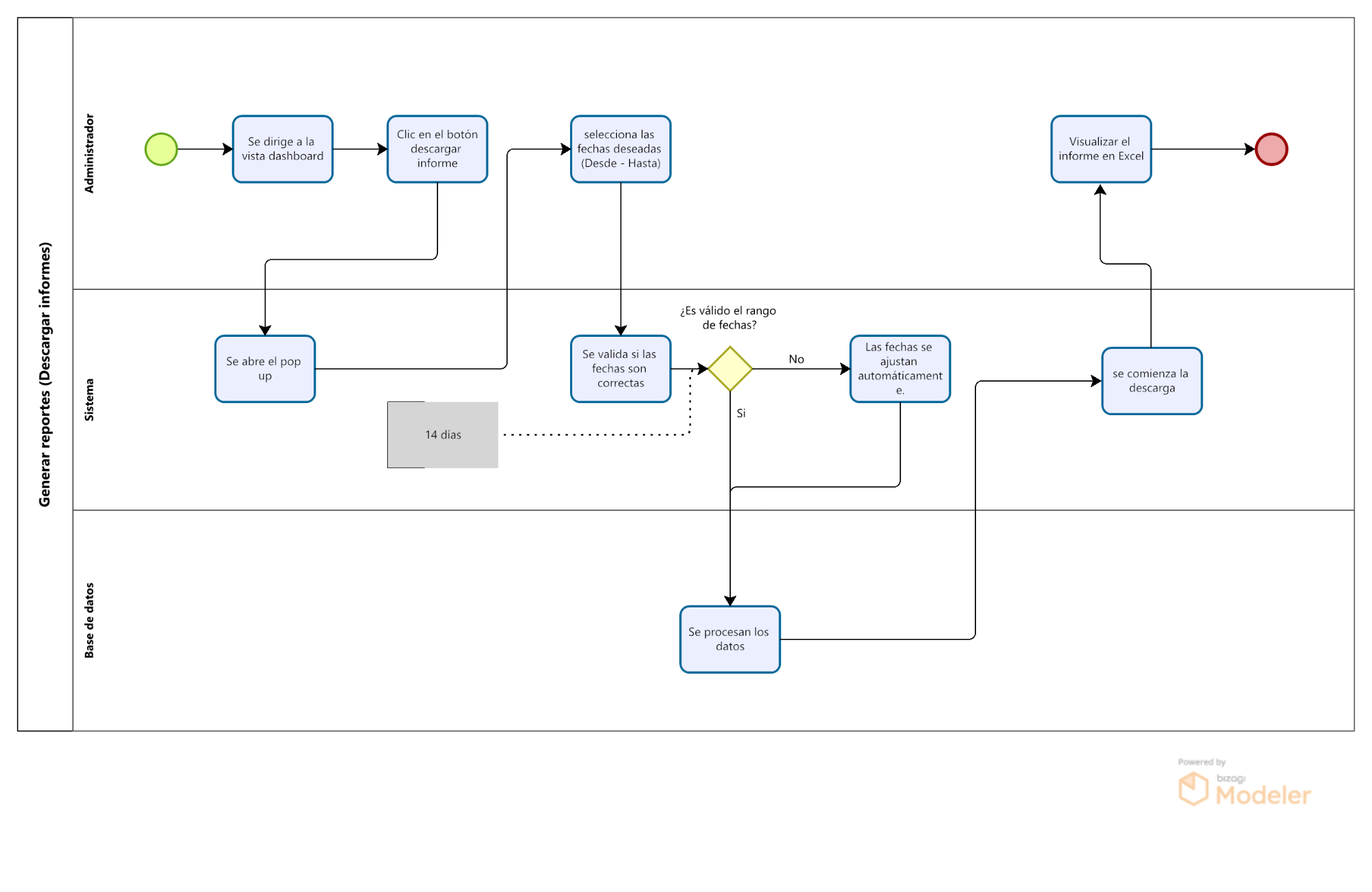
Sistema: El software va a poder realizar registros de lavados, de nuevas pecheras y guardarlos en la base de datos, a su vez realizar modificación en la información de estas. Podrá ingresar nuevos centros de trabajo, modificarlos e inhabilitarlos.

**Vista de Procesos**

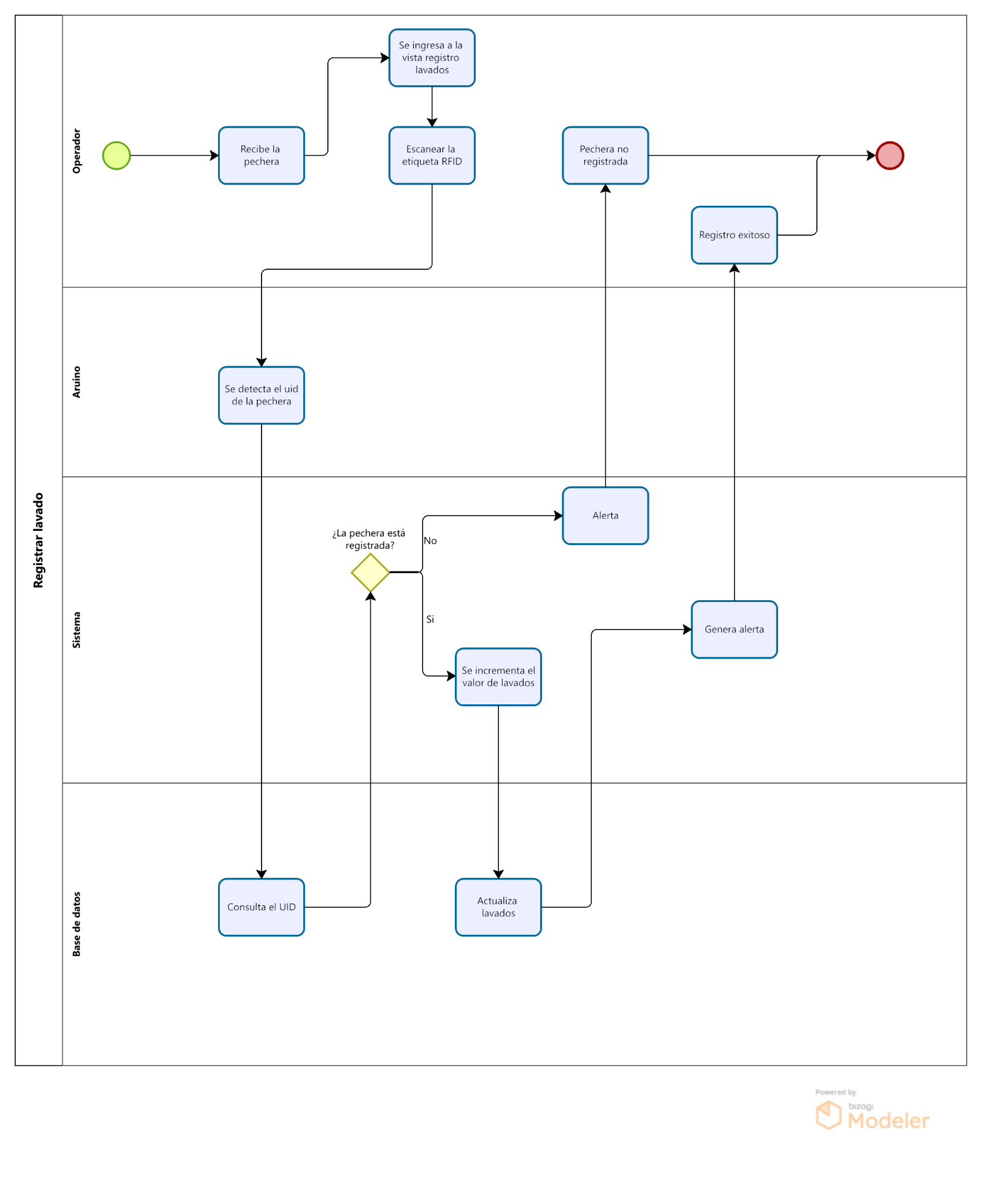
Registro: Se permite el ingreso de información sobre las pecheras por medio de la lectura de RFID y se solicita información como talla (obligatorio), centro de trabajo (opcional), empresas, usuarios y datos de lavados.

Generación de informes: Se generan reportes en formato Excel con la información filtrada por rango de fechas y así poder visualizar las pecheras con su cantidad de lavados.

**Diagrama de procesos**

****Figura 8: Elaboración propia

##### 

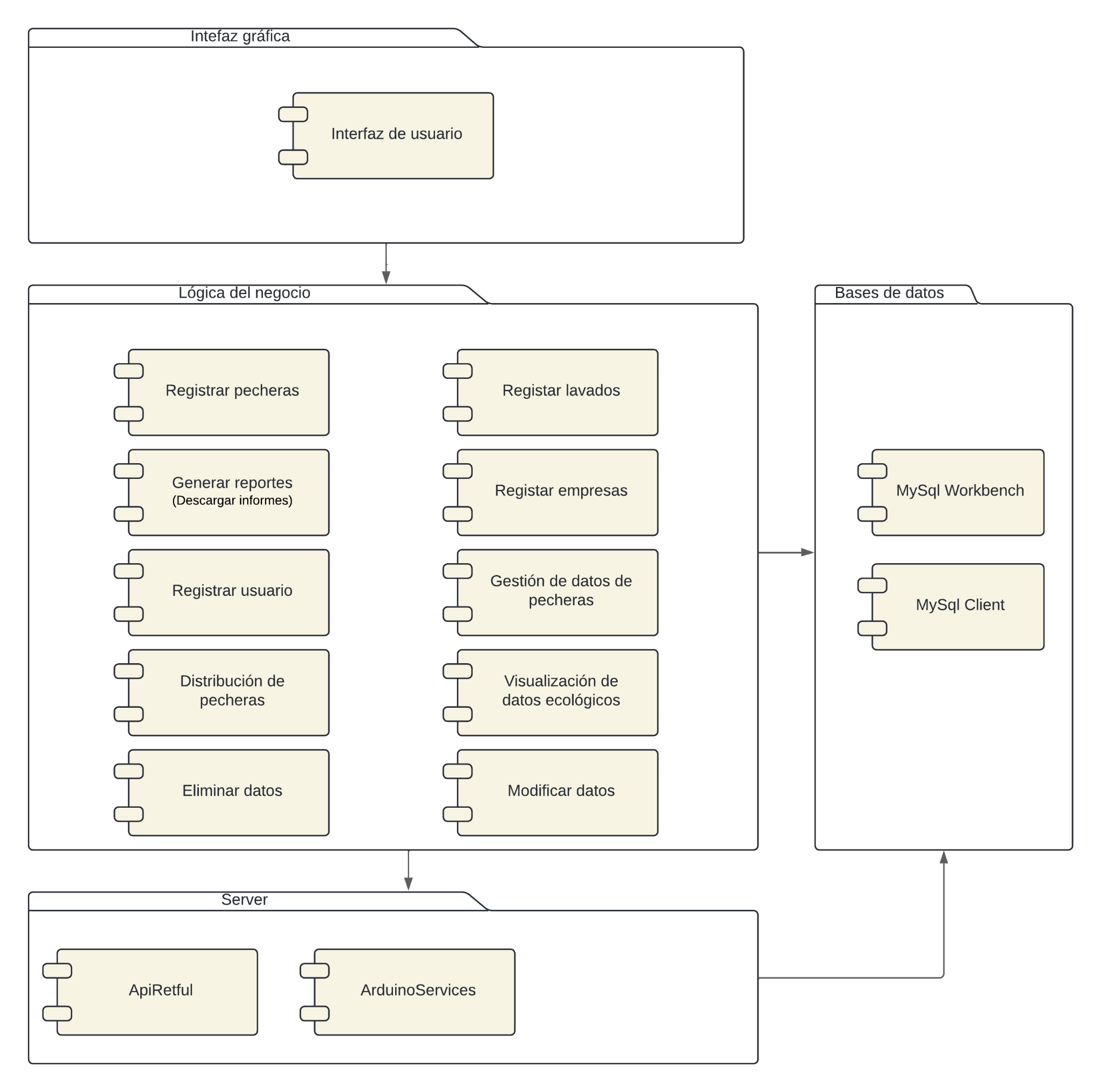


##### Figura 9: Elaboración propia

##### 

**Arquitectura de desarrollo**

En esta vista se aprecia que existirán dos módulos principales que contendrán distintas funcionalidades de la aplicación. A continuación, se describen:



##### Figura 10: Elaboración propia

## ETAPA N°5

**Base de datos**

Para el desarrollo de la base de datos se realizó una búsqueda para encontrar la base de datos que más nos sirva, tanto en calidad para poder realizar todos los requerimientos que nos pidieron y que tenga un buen precio para poder ser rentable.

**Herramientas utilizadas**

MySQL workbench 8.0 CE: El MySQL workbench es la herramienta que proporciona MySQL para poder realizar todas las funciones de la base de datos, ya sea, tanto modelar, crear o administrar una base de datos. Aquí se realizan las consultas para poder visualizar de mejor manera los datos que se están manejando.

MySQL client: El MySQL client es la herramienta para poder administrar y configurar los puertos, las credenciales de las bases de datos.

**Estructura y creación de la base de datos**

**Credenciales y configuración:**

Cuando creamos la base de datos primero tuvimos que configurar toda las credenciales y usuario que van a poder usarlo, en nuestro caso utilizamos las credenciales por defecto, user: root y password:12345.

**Creación:**

Realizada la creación de usuario y configuración de puertos para las bases de datos procedemos a crear la base de datos en MySQL workbench, creando un nuevo schema, llamado “Lavandería”, creado el esquema procedemos a la creación de las tablas que vamos a utilizar para almacenar todos los datos, se crean diferente para que cada actor u objeto se almacene de forma que no se creen redundancias y se repitan datos, este paso se llama normalización.

**Tablas**

Procediendo con las tablas, se le asigna un nombre a la tabla, este nombre va en base a lo que quieres guardar en ella, por ejemplo, Pecheras, una vez definida el nombre de la tabla procedemos a asignarles sus propiedades o también llamadas columnas, estas columnas tienen un formato, este formato es así, NOMBRE\_COLUMNA/TIPO\_DATO(TAMAÑO)/RESTRICCIÓN.

Aquí es donde asignamos que va a tener esta tabla, ya sea su id, el nombre, tomando el ejemplo anterior, para la tabla Pecheras sus propiedades o las columnas que va a tener esa tabla podrían ser: id\_pechera\_registro, nombre\_empleado, gracias a estas propiedades podemos filtrar de mejor manera los datos que estamos guardando.

Una vez asignada las propiedades de las tablas, se le asigna el tipo de dato que van a tener, este paso es uno de los más importantes, porque es aquí donde le asignamos el tipo de información que va a guardar, ya sea información solo numérica (int), información alfanumérica (Varchar), entre otras, también a los tipo de datos se le asigna un tamaño dependiendo de los necesitando, por ejemplo en nuestra tabla Pecheras el id\_pechera\_registro sería un int, con esto solo se podrá ingresar datos numéricos a la columna id\_pechera\_registro o también si queremos que la columna Parámetros se ingresen datos con letras y números, se le agregaría Varchar y para asignarle un tamaño límite a ese nombre se le agregaría Varchar(100), siendo 100 la cantidad de caracteres que puede tener y si quieres ingresar una fecha se le agrega *DATE* o *TIMESTAMP,* en resumen el tipo de dato es la parte en donde uno define que tipo información queremos guardar o restringirla de algún tipo de dato.

Después de definido el tipo de dato que va a tener nuestra columna o propiedad se le asigna una restricción, generalmente en una tabla existe una propiedad o columna que es el que le da la identificación al objeto o entidad que estamos almacenando, este se llama “ID” o clave primaria y se puede asignar de una o más formas, pero las más comunes son estas:

Asignarla cuando estás creando la tabla, ocupando el ejemplo de “*Pecheras*” nuestra columna id\_pechera\_registro para poder asignarle su restricción con clave primaria quedaría de esta forma: id\_pechera\_registro int primary key, el primary key es la restricción que le estamos dando, con esto, la columna id\_pechera\_registro queda como obligatoria, porque no podremos ingresar una pechera sin su identificador.

La otra forma de asignar una clave primaria es agregándola cuando la tabla ya está creada, este método se ocupa cuando quieres tener un mejor orden al momento de crear una tabla o quieres cambiar o agregarla a una tabla que no tenía clave primaria de un comienzo. Para poder hacer este método se tiene que alterar la tabla que queremos modificar para poder agregar la restricción y esta quedaría así:

alter table empleado add CONSTRAINT pk\_idempleado primary key(id\_pechera\_registro*)*

Aquí como se observa se está alterando la tabla empleado con alter table empleado, se está agregando la restricción y su nombre con add CONSTRAINT pk\_idempleado (aquí el nombre de la restricción es totalmente al gusto del usuario) y se está definiendo la restricción y la columna o propiedad a la que está dirigida primary key(id\_pechera\_registro).

Continuando con las restricciones, una vez ya definida tu clave primaria se continúa con las otras columnas, por defecto si uno deja en blanco la restricción la columna queda como null, que quiere decir esto, que cuando ingreses algo a tu tabla, la columna sin restricción puede quedar en blanco, de la contrario, si tú quieres agregarle la restricción para que la columna no pueda quedar en blanco uno le agrega not null al final de esta, quedaría algo así, por ejemplo:

Parámetros Varchar(100)not null,

Con esto el usuario está obligado a ingresar ese campo, ya que no está permitido dejar en blanco o en este caso null (nulo).

Otras restricciones que se ocuparon en la base de datos son:

UNIQUE, esta restricción hace que la columna tenga un valor único, que no se puede repetir. Este quedaría así:

id\_pechera\_registro VARCHAR(100) NOT NULL UNIQUE PRIMARY KEY

AUTO\_INCREMENT, esta restricción normalmente se les asigna a claves primarias, esto hace que cada vez que se ingrese algo a una tabla el valor que tenga empiece desde el 1 y aumentando cada vez que se ingrese un nuevo registro a la tabla, esto quedaría así:

id\_pechera\_registro int AUTO\_INCREMENT PRIMARY KEY

DEFAULT, esta restricción nos permite asignarle un valor predeterminado a una columna, por ejemplo, si queremos ingresar en nuestra tabla Planta en la columna pais\_planta y que esta sea por defecto Chile quedaría algo así:

pais\_planta varchar(100) Default ‘Chile’

Aquí en este caso se define Chile como país predefinido, esto quiere decir que al momento de insertar un empleado y se deja en blanco la columna pais\_empleado automáticamente quedará con el valor “Chile”, pero si al momento de ingresar se escribe otro valor se saltará la restricción y quedará con el que el usuario escribió.

CURRENT\_TIMESTAMP, esta restricción se utiliza en las columnas que tienen como tipo de dato un DATE o TIMESTAMP, al colocar esta restricción hace que al ingresar un dato en una tabla y se deja en blanco la columna con esa restricción, automáticamente se ingresará la fecha exacta en la que se ingresó el registro y normalmente se le agrega la restricción DEFAULT para poder asignarla por defecto. Y quedaría así:

fecha\_registro TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

Y por último FOREIGN KEY, esta una de las restricciones más importante ya que permite unir dos tablas mediante columnas compartidas, creando tablas hijas y tablas padre, con esto podemos compartir información entre las dos tablas al momento de querer realizar una consulta y además aumenta la seguridad de las tablas dando limitaciones al momento de querer eliminar o modificar una tabla. Estas foreign key se agregan al momento de crear o ya creada la tabla, por ejemplo, tengo la tabla Pechera y la tabla Planta al momento de crear Pechera se le agregaría al final de esto:

CONSTRAINT fk\_id\_planta FOREIGN KEY (id\_planta) REFERENCES Planta(id\_planta)

Aquí agregamos la restricción con su nombre, el nombre de la columna que está en Pechera y a que tabla hace referencia, en este caso Planta con el nombre de la columna que es id\_planta*.*

Ahora una vez entendido las partes que se ocuparon para crear la base de datos explicaremos cuales son las tablas que se crearon más el formato de creación de estas y una breve descripción.

En nuestro proyecto creamos una base de datos con nombre Lavandería a la cual se le crearon 6 tablas para poder almacenar los datos de esta empresa.

**Planta**

CREATE TABLE planta(

id\_planta INT AUTO\_INCREMENT PRIMARY KEY,

nombre\_planta Varchar(100) unique not null,

cantidad int,

cantidad\_asignada int not null,

kilo int not null,

estado BOOLEAN DEFAULT true

);

Descripción: En esta tabla se almacenan los datos de las plantas o centro de trabajo con los que trabaja la empresa, sus columnas son:

***id\_planta*** es la clave primaria tiene esa planta.

***nombre\_planta*** para asignarle un nombre a planta y que no se repita.

***cantidad*** es donde se define la *cantidad* de pecheras que tiene esa planta o centro.

***cantidad\_asignada*** es la cantidad de pecheras que la empresa necesita.

***kilo***es la cantidad de kilos que la empresa tiene en relación con la cantidad de pecheras que tiene.

***estado*,** aquí se define si la planta está activa o inactiva, según el valor que tenga el booleano, true como activa, false como inactiva.

**Pechera**

CREATE TABLE pechera (

id\_pechera\_registro VARCHAR(100) NOT NULL UNIQUE PRIMARY KEY,

fecha\_registro TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

Cantidad\_Lavados INT DEFAULT 0,

Talla VARCHAR(100) NOT NULL,

Parametros VARCHAR(100),

Observaciones VARCHAR(100),

Índice\_Microbiológico VARCHAR(100),

id\_planta INT

);

Descripción: En esta tabla se almacena toda la información de las pecheras que la empresa crea, sus columnas son:

**id\_pechera\_registro** es la clave primaria de la pechera y no se puede repetir.

**fecha\_registro** aquí se guarda la fecha actual del registro de la pechera.

**Cantidad\_Lavados** es la cantidad de lavados que tiene la pechera.

**Talla** la talla de la pechera que se registra.

**Parametros** los parámetros que va a tener la pechera.

**Observaciones** aquí se almacenará cualquier observación que tenga la pechera.

**Índice\_Microbiológico** es si la pechera cuenta con o sin índice microbiológico.

**id\_planta** es la clave foránea para que pechera tenga el acceso a la información de Planta y así poder asignarles una planta a las pecheras.

**Lavado**

CREATE TABLE lavado(

id\_pechera\_registro Varchar(100) not null,

Fecha\_lavado TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Descripción: En esta tabla se guardan los registros de los lavados que realiza a las diferentes pecheras, sus columnas son:

**id\_pechera\_registro** es la clave foránea para que la tabla lavado tenga acceso a los datos de las pecheras y se le pueda ir sumando la cantidad de lavados.

**Fecha\_lavado** aquí se registrará la fecha del lavado realizado.

**Historial**

CREATE TABLE historial(

id\_historial INT AUTO\_INCREMENT PRIMARY KEY,

id\_pechera\_registro Varchar(100) not null,

fecha\_registro date not null,

Cantidad\_Lavados int,

Talla Varchar(100) not null,

Observaciones Varchar(500),

fecha\_eliminado TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

id\_planta INT

);

Descripción: La tabla historial es donde se guardará toda la información de una pechera cuando se eliminan, por ejemplo, si elimino una pechera, aquí se almacenará toda la información que esa pechera tubo y se tendrá un registro de ella y sus columnas son:

**id\_historial** es el id que se le asigna a cada pechera una vez eliminada

**id\_pechera\_registro** es la clave primaria de la pechera que fue eliminada

**fecha\_registro** aquí se guarda la fecha de creación de la pechera.

**Cantidad\_Lavados** es la cantidad de lavados que tuvo la pechera.

**Talla** la talla de la pechera tenía.

**Observaciones** aquí se almacenará cualquier observación que tuvo la pechera.

fecha\_eliminado aquí se almacena la fecha de eliminación de la pechera

**id\_planta** es la clave foránea para que pechera tenga el acceso a la información de Planta y así poder asignarles una planta a las pecheras.

**Login**

CREATE TABLE login (

id\_login INT AUTO\_INCREMENT PRIMARY KEY,

nombre\_completo VARCHAR(100) NOT NULL,

correo VARCHAR(100) UNIQUE,

contraseña VARCHAR(100) NOT NULL,

id\_planta INT not null

);

Descripción: En esta tabla se almacenan los datos de los usuarios que van a tener acceso al software, sus columnas son:

**id\_login** es el id que se le asigna a cada usuario registrado.

**nombre\_completo** el nombre del usuario que se va a ingresar.

**correo** el correo del usuario.

**contraseña** la contraseña del usuario.

**id\_planta** el id de la tabla planta para poder asignarle una planta o centro de trabajo a ese usuario.

**Taller**

CREATE TABLE Taller(

id\_pechera\_registro Varchar(100) not null,

fecha\_llegada date not null

);

Descripción: En la tabla Taller se almacenarán todas las pecheras que son enviadas a Taller y sus columnas son:

**id\_pechera\_registro** es la clave foránea para que se tenga acceso a toda la información de la pechera mediante su id.

**fecha\_llegada** la fecha de cuando fue enviada.

Alter table.

El alter table es una forma de poder modificar las tablas ya creadas, ya sea modificando el tipo de dato de alguna columna, eliminar una de esta de alguna tabla o agregar una nueva columna, en nuestro caso lo utilizamos para poder agregar las foreign key a las tablas, los “alter table” utilizados son:

alter table pechera add foreign key (id\_planta) references planta (id\_planta);

Descripción: Alterar la tabla **pechera** para agregarle una foreign key, en este caso **id\_planta** pasa a ser la foreign key de esta tabla y hace referencia a la tabla **planta** con el **id\_planta**

alter table historial add foreign key (id\_planta) references planta (id\_planta);

Descripción: Alterar la tabla **historial** para agregarle una foreign key, en este caso **id\_planta** pasa a ser la foreign key de esta tabla y hace referencia a la tabla **planta** con el **id\_planta**

alter table Taller add foreign key (id\_pechera\_registro) references pechera (id\_pechera\_registro);

Descripción: Alterar la tabla **Taller** para agregarle una foreign key, en este caso **id\_pechera\_registro** pasa a ser la foreign key de esta tabla y hace referencia a la tabla **pechera** con el **id\_pechera\_registro**

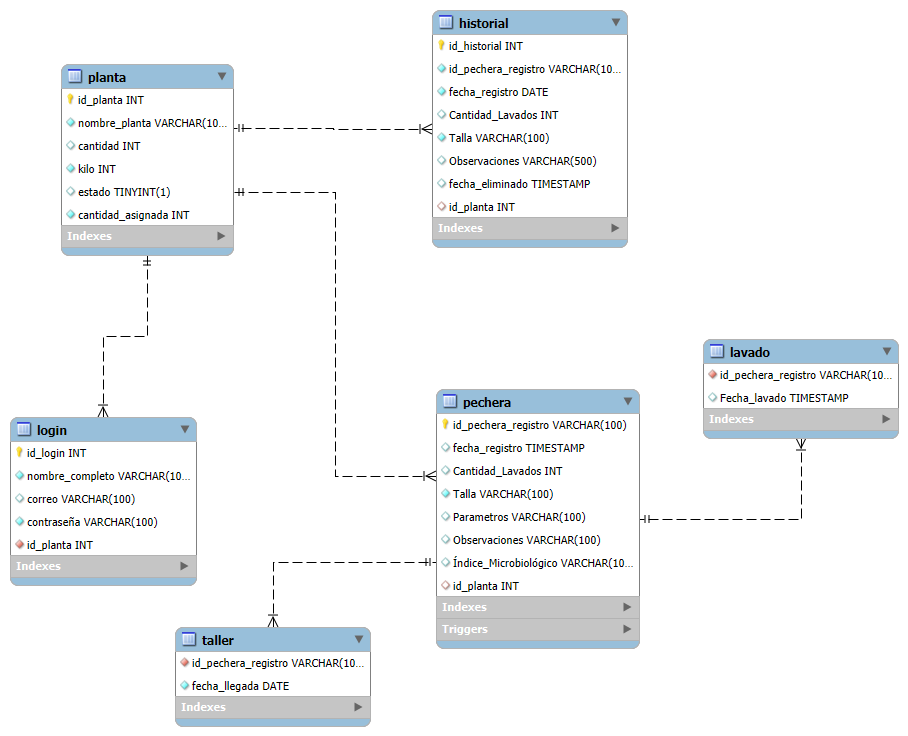
alter table login add foreign key (id\_planta) references planta (id\_planta);

Descripción: Alterar la tabla **login** para agregarle una foreign key, en este caso **id\_planta** pasa a ser la foreign key de esta tabla y hace referencia a la tabla **planta** con el **id\_planta**

alter table lavado add foreign key (id\_pechera\_registro ) references pechera (id\_pechera\_registro );

Descripción: Alterar la tabla lavadopara agregarle una foreign key, en este caso id\_pechera\_registro pasa a ser la foreign key de esta tabla y hace referencia a la tabla pechera con el id\_pechera\_registro .

**Modelo entidad relación** (MER)



**Explicación del MER:**

En el MER se muestra cómo se relacionan las diferentes tablas que tiene la base de datos, en nuestro caso tenemos un total de 6 tablas y estas se relacionan e interactúan de esta forma:

Las principales tablas son pechera y planta, estas tablas son consideradas tablas madre ya que son las que comparten sus claves primarias a las otras tablas para entablar una relación entre ellas.

En la tabla pechera se almacenan todas las pecheras que la lavandería quiere registrar en el sistema, para así guardar toda la información relacionada a estas y poder distribuirlas a las diferentes plantas con las que ellos trabajan.

Las relaciones que tiene la tabla pechera son estas:

lavado. En la tabla lavado se almacena todas las veces que la pechera se le ingresa un lavado, almacenando en esta tabla el id\_pechera\_registro y la fecha en la que realizó el registro con fecha\_lavado.

taller. En taller se almacenan todas las pecheras que son mandadas a reparación o inspección y para que se almacenen se guarda el id\_pechera\_registro de la pechera que se está mandando y la fecha en la que fue mandada, con fecha\_llegada.

Con estas relaciones estas tablas podrán recibir todos los datos de la tabla pecheras, pero existe una tabla que es una excepción a la regla que es:

historial. Historial es en donde se almacenan todas las pecheras que son eliminadas utilizando un trigger (que se explicará más adelante), en esta tabla se gradara tanto el id\_pechera\_registro, como el id\_planta, pero a diferencia de id\_planta, id\_pechera\_registro no es una clave foránea, sino una columna norma, porque si nosotros queremos reutilizar el chip de la pechera eliminada nos daría error porque se estarían repitiendo las claves id\_pechera\_registro y para que eso no suceda a cada id\_pechera\_registro se le asigna un id\_historial y gracias a este id podemos eliminar cuantas veces queramos la misma pechera y no dará error por claves primarias repetidas.

Luego está la tabla planta. En esta tabla se almacenan todas las plantas o centro de trabajo con los que trabaja la lavandería, aquí se almacenará la cantidad de pecheras que tiene esa planta y la cantidad que fue solicitada, sumado a la cantidad de kilos de plástico asociado.

Las relaciones que tiene la tabla planta son estas:

login. En la tabla login se almacenan los datos del usuario que va a tener permiso al software, aquí se almacena el id\_planta para poder saber de qué planta o centro de trabajo es este usuario.

historial. En historial se almacenan todos los datos de la tabla pechera, como se había mencionado, pero también se almacena id\_planta para poder saber de qué planta era esa pecera eliminada y así poder tener un registro de cuantas pecheras eliminadas son de cada empresa.

pechera. En pecheras se almacena toda la información de las pecheras más el id\_planta, con el id\_planta podemos saber a qué planta está dirigida o tienen la pechera.

Funciones **INSERT**, **UPDATE** Y **DELETE.**

En las bases de datos existen funciones que nos permitirán poder administrar nuestras tablas, ya sea agregando datos, eliminar datos o modificarlos, para eso ocupamos INSERT, UPDATE Y DELETE.

Se explicará detalladamente cada una de estas funciones y cómo la utilizamos en nuestra base de datos.

**INSERT**

El **INSERT** o insertar en español, es la función que nos permite poblar una tabla con los datos que necesitemos.

El formato del **INSERT** es este:

INSERT into Mi\_Tabla (columna1,columna2,..) VALUES (‘Valor\_columna1',’valor\_columna2',...);

El insert nos pedirá la tabla a la que vamos a insertar los datos, en este caso “**Mi\_Tabla**” las columnas a las que vamos a poblar columna1,columna2 o todas las que tenga tu tabla y luego con insertamos los valores (VALUES) que queremos en nuestra tabla según el orden que le asignamos al comienzo, el formato de ingresar estos valores es importante ya que si no lo hacemos correctamente no nos permitirá insertar datos, para los datos o valores alfanuméricos y fechas se escriben entre comillas simple **‘’** y los datos numéricos se ingresan sin comillas simples.

Un ejemplo utilizado en nuestra base de datos para poderla ingresar pecheras a la tabla Pecheras es este:

insert into pechera (id\_pechera\_registro,Talla) values ('13','L');

Aquí insertamos valores en las columnas id\_pechera\_registro y Talla ya que las otras columnas no son necesarias y obligatorias al momento de ingresar una pechera nueva.

**UPDATE**

El **UPDATE** se utiliza para actualizar valores existentes dentro de una tabla, puedes actualizar varios datos a la vez o puedes filtrar y actualizar un dato en específico.

El formato del **UPDATE** es este:

UPDATE Mi\_tabla SET columna1 = nuevo\_valor\_columna1 WHERE ID\_Tabla = valor\_ID\_Tabla

El update nos permite actualizar los datos que nosotros indiquemos y a la tabla que queremos cambiar, en este caso se está actualizando la tabla Mi\_tabla y la columna columna1, se le define un nuevo valor a esta columna mediante un igual y luego lo más importante es definir qué dato queremos modificar, a qué me refiero con esto, si yo ahora no defino el dato en específico que quiero modificar, se modificarán todos los datos de columna1 y para evitar eso nosotros agregamos el WHERE.

El WHERE (donde) es la cláusula que filtra los datos que vamos a actualizar, en este caso lo estamos filtrando con el ID\_Tabla, aquí es donde le asignamos qué dato en específico queremos modificar, en este caso es valor\_ID\_Tabla, que sería el id del dato que estamos actualizando y si nosotros queremos modificar una fila en su totalidad quedaría así la consulta:

UPDATE Mi\_tabla SET columna1 = valor\_ID\_Tabla

**DELETE**

El delete se utiliza para eliminar registros de nuestras tablas, puedes eliminar múltiples datos a la vez o puedes eliminar datos de forma específica.

El formato del **DELETE** es este:

DELETE FROM Mi\_tabla WHERE ID\_Tabla= valor\_ID\_Tabla

Aquí el DELETE está eliminando datos de la tabla Mi\_tabla y se le está agregando una cláusula WHERE para poder filtrar el dato que queremos eliminar, en este caso estamos filtrando por ID\_Tabla y le asignamos el valor de la id que queremos eliminar en este caso valor\_ID\_Tabla y si no le asignamos un valor por el cual filtrar eliminaremos todos los registros de la tabla y su consulta es esta:

DELETE FROM Mi\_tabla

**Triggers**

Los trigger o disparadores son procedimientos almacenados que se ejecutan automáticamente en respuesta a ciertos eventos que ocurren en una tabla o en la base de datos. Estos pueden ejecutarse antes de que pase el evento, llamado **BEFORE** o después de que ocurra el evento, llamado **AFTER.** Los eventos que ejecutan estos triggers son el **UPDATE**, **INSERT** o **DELETE.**

Estos trigger sea pueden utilizar para tener una base de datos en constante actualización, ya que estos nos permiten modificar varias cosas al mismo tiempo que ocurre una acción, por ejemplo, en nuestro caso tenemos la tabla “**historial**” que es la que almacena todas las pecheras que son eliminadas, gracias a un trigger, todas pecheras eliminadas se guardan en la tabla **historial** antes de ser eliminadas por completo**.**

Una vez explicados que son los triggers les mostraremos los triggers utilizados para nuestra base de datos.

Trigger “**Eliminar\_pechera**”

DELIMITER $$

CREATE TRIGGER Eliminar\_pechera

BEFORE DELETE ON pechera

FOR EACH ROW

BEGIN

INSERT INTO historial (id\_pechera\_registro,fecha\_registro,Cantidad\_Lavados,Talla,

Observaciones,fecha\_eliminado,id\_planta) VALUES(OLD.id\_pechera\_registro,OLD.fecha\_registro,OLD.Cantidad\_Lavados,OLD.Talla,OLD.Observaciones, NOW(),OLD.id\_planta);

END$$

DELIMITER ;

Descripción y explicación:

Definición del trigger:

El trigger se parte con un DELIMITER $$ para que MySQL ejecute todo el bloque completo de una sola vez. Luego se procede a crear el trigger en sí, partiendo, asignándole en nombre del trigger, que es Eliminar\_pechera.

Se especifica cuando se ejecuta el trigger, en este caso “antes de” ya que se está usando un BEFORE, lo que quiere decir que se ejecutará antes de que ocurra la función asignada. Se especifica qué función que provocará que el trigger se ejecute y a que tabla va dirigida, en este caso un DELETE ON pechera, esto quiere decir que cualquier eliminación en pecheras ejecuta el trigger.

Y por último se le agrega el FOR EACH ROW para indicar que el trigger se ejecutará por cada dato que se elimine de la tabla pechera.

Luego de explicada la definición del trigger se prosigue con la consulta que ejecuta el trigger al ser accionado.

Cuerpo del Trigger:

Al eliminar un dato de pechera se ejecutará una consulta INSERT en la tabla historial, aquí se insertarán los datos de la tabla pechera que anteriormente fueron eliminados y esto se ingresarán con este formato:

VALUES(OLD.id\_pechera\_registro,OLD.fecha\_registro,OLD.Cantidad\_Lavados,OLD.Talla,OLD.Observaciones, NOW(),OLD.id\_planta);

Para poder agregar los datos eliminados de la tabla pechera a los valores se le tienen que agregar el OLD. con esto llamamos a los datos de la tabla pechera antes de que sean eliminados. También le agregamos NOW()para poder ingresar la fecha y hora actual, para así poder indicar cuando se realizó la eliminación.

Y finalizamos ingresando END$$ para cerrar el trigger y DELIMITER ; para cerrar el bloque de la consulta.

Resumen:

El trigger hace que los datos eliminados de pechera se guarden en la tabla historial para tener un registro de todas las pecheras eliminadas, con los datos de esta y la fecha en la que fue eliminada.

Trigger **pechera\_insert\_trigger**

DELIMITER $$

CREATE TRIGGER pechera\_insert\_trigger

AFTER INSERT ON pechera

FOR EACH ROW

BEGIN

IF NEW.id\_planta IS NOT NULL THEN

UPDATE planta

SET cantidad = cantidad + 1

WHERE id\_planta = NEW.id\_planta;

END IF;

END$$

DELIMITER ;

Descripción:

Definición del trigger:

El trigger parte con un DELIMITER $$ para que MySQL ejecute todo el bloque completo de una sola vez. Luego se procede a crear el trigger en sí, partiendo, asignándole en nombre del trigger, que es: pechera\_insert\_trigger.

Se especifica cuando se ejecuta el trigger, en este caso “después de” ya que se está usando un AFTER, lo que quiere decir que se ejecutará antes de que ocurra la función asignada. Luego, se indica qué acción provocará que el trigger se ejecute y a cuál tabla va dirigida. En este caso un INSERT ON pechera, lo que significa que el trigger se ejecutará cuando se inserte una pechera.

Y por último se le agrega el FOR EACH ROW para indicar que el trigger se ejecutará por cada dato insertado.

Cuerpo del Trigger:

Al insertar un dato en pechera se ejecutará una consulta IF para comprobar si la columna id\_planta de la pechera no es NULL y si este no es NULL se continúa con la consulta utilizando el THEN.

Luego de definida la condición para que se ejecute el trigger, se ejecutará una consulta UPDATE que actualizará la cantidad de pecheras que tiene la planta en un +1, y para definir qué planta vamos a modificar se ingresa una cláusula WHERE para comparar el id\_planta de las empresas existentes con el nuevo NEW.id\_planta de la nueva pechera recién ingresada.

Y finalizamos ingresando END IF;para finalizar la función IF, el END$$ para cerrar el trigger y DELIMITER ; para cerrar el bloque de la consulta.

Resumen:

El trigger se ejecuta solo cuando se ingrese una pechera con un id\_planta asignada, para luego en la tabla planta sumar una nueva pechera a su cantidad de pecheras.

Trigger **pechera\_update\_trigger**

DELIMITER $$

CREATE TRIGGER pechera\_update\_trigger

AFTER UPDATE ON pechera

FOR EACH ROW

BEGIN

IF OLD.id\_planta != NEW.id\_planta THEN

IF OLD.id\_planta IS NOT NULL THEN

UPDATE planta

SET cantidad = cantidad - 1

WHERE id\_planta = OLD.id\_planta;

END IF;

IF NEW.id\_planta IS NOT NULL THEN

UPDATE planta

SET cantidad = cantidad + 1

WHERE id\_planta = NEW.id\_planta;

END IF;

END IF;

END$$

DELIMITER ;

**Descripción**

Definición del trigger:

El trigger se parte con un DELIMITER $$ para que MySQL ejecute todo el bloque completo de una sola vez. Luego se procede a crear el trigger en sí, partiendo, asignándole en nombre del trigger, que es: pechera\_update\_trigger.

Se especifica cuando se ejecuta el trigger, en este caso “después de” ya que se está usando un AFTER, lo que quiere decir que se ejecutará antes de que ocurra la función asignada. Se especifica qué función que provocará que el trigger se ejecute y a que tabla va dirigida, en este caso un UPDATE ON pechera

,esto quiere decir que al momento de insertar una pechera se ejecutara el trigger.

Y por último se le agrega el FOR EACH ROW para indicar que el trigger se ejecutará por cada dato insertado.

Cuerpo del Trigger:

Al modificar un dato en pechera se ejecutará una consulta IF para comprobar si la antigua columna id\_planta no es igual a la nueva id\_planta, esto se especifica aquí:

IF OLD.id\_planta != NEW.id\_planta

Aquí lo que estamos haciendo es utilizar “!=”para indicar que no tiene que ser igual a algo, en este caso

OLD.id\_planta con NEW.id\_planta, para poder saber si lo que se modificó fue el id\_planta de la pechera y poder continuar con el trigger y se continúa con la consulta utilizando el THEN.

Luego de definida la condición para que se ejecute el trigger, se ejecutará una consulta UPDATE que actualizará la cantidad de pecheras que tiene la planta en un +1 y -1, ya que estaremos sumando a una nueva planta cuando le cambiemos el id\_planta a una pechera y le estaremos restando a la actual. Y para definir qué planta vamos a modificar se ingresa una cláusula WHERE para comparar el id\_planta de la empresa actual ha cambiado, si el id\_planta es igual al OLD.id\_planta se le restará un -1 a la cantidad y si el id\_planta coincide con el nuevo NEW.id\_planta se le sumará +1.

Y finalizamos ingresando END IF; para finalizar la función IF, el END$$ para cerrar el trigger y DELIMITER ; para cerrar el bloque de la consulta.

Resumen:

La función del trigger es actualizar la cantidad de pecheras que tiene la tabla planta cuando se modifica el id\_planta de una pechera, restando a la antigua planta en la que estaba la pechera y sumándole a la nueva planta en la que está la pechera

trigger **pechera\_delete\_trigger**

DELIMITER $$

CREATE TRIGGER pechera\_delete\_trigger

AFTER DELETE ON pechera

FOR EACH ROW

BEGIN

IF OLD.id\_planta IS NOT NULL THEN

UPDATE planta

SET cantidad = cantidad - 1

WHERE id\_planta = OLD.id\_planta;

END IF;

END$$

DELIMITER ;

Descripción:

Definición del trigger:

El trigger se parte con un DELIMITER $$ para que MySQL ejecute todo el bloque completo de una sola vez. Luego se procede a crear el trigger en sí, partiendo, asignándole en nombre del trigger, que es: pechera\_delete\_trigger

Se especifica cuando se ejecuta el trigger, en este caso “después de” ya que se está usando un AFTER, lo que quiere decir que se ejecutará antes de que ocurra la función asignada. Se especifica qué función que provocará que el trigger se ejecute y a que tabla va dirigida, en este caso un DELETE ON pechera

,esto quiere decir que al momento de eliminar una pechera se ejecutara el trigger. Y por último se le agrega el FOR EACH ROW para indicar que el trigger se ejecutará por cada dato insertado.

Cuerpo del Trigger:

Al insertar un dato en pechera se ejecutará una consulta IF para comprobar si la columna id\_planta de la pechera no es NULL y si este no es NULL se continúa con la consulta utilizando el THEN.

Luego de definida la condición para que se ejecute el trigger, se ejecutará una consulta UPDATE que actualizará la cantidad de pecheras que tiene la planta en un -1, y para definir qué planta vamos a modificar se ingresa una cláusula WHERE para comparar el id\_planta de la empresa existente con la antigua OLD.id\_planta de la pechera eliminada y corroborar si esta tenía un id\_planta válida .

Y finalizamos ingresando END IF;para finalizar la función IF, el END$$ para cerrar el trigger y DELIMITER ; para cerrar el bloque de la consulta.

Resumen:

Este trigger se ejecuta cuando una pechera es eliminada y si esta tenía un id\_planta asignada disminuir la cantidad de pecheras de la planta.

trigger **Eliminar\_lavado**

DELIMITER $$

CREATE TRIGGER eliminar\_lavado

BEFORE DELETE ON pechera

FOR EACH ROW

BEGIN

DELETE FROM lavado

WHERE id\_pechera\_registro = OLD.id\_pechera\_registro;

END$$

DELIMITER ;

Descripción:

Definición del trigger:

El trigger se parte con un DELIMITER $$ para que MySQL ejecute todo el bloque completo de una sola vez. Luego se procede a crear el trigger en sí, partiendo, asignándole en nombre del trigger, que es: pechera\_delete\_trigger.

Se especifica cuando se ejecuta el trigger, en este caso “antes de ” ya que se está usando un BEFORE, lo que quiere decir que se ejecutará antes de que ocurra la función asignada. Se especifica qué función que provocará que el trigger se ejecute y a que tabla va dirigida, en este caso un DELETE ON pechera

,esto quiere decir que al momento de eliminar una pechera se ejecutara el trigger. Y por último se le agrega el FOR EACH ROW para indicar que el trigger se ejecutará por cada dato insertado.

Cuerpo del Trigger:

Al eliminar un dato en pechera automáticamente se eliminará el mismo dato en la tabla lavado, y para saber que pechera se va a eliminar se hace una cláusula WHERE para comparar id\_pechera\_registro con la antigua OLD.id\_pechera\_registro y corroborar que esa pechera estaba en la tabla lavado. Y finalizamos ingresando END IF; para finalizar la función IF, el END$$ para cerrar el trigger y DELIMITER ; para cerrar el bloque de la consulta.

Resumen:

Al momento de eliminar un dato en pechera, si esta pechera tenía un lavado y se eliminará también de la tabla lavado.

**Consultas en la base de datos**:

Las consultas son una parte esencial al momento de mostrar datos, permitiendo filtrar información con las diferentes cláusulas, aquí es donde también se unen tablas para poder llamar datos de estas y combinar datos de ambas tablas. Ahora se explicará más detalladamente

El formato que tiene las consultas que ocupamos es el siguiente:

SELECT nombre\_columna1 FROM Nombre\_Tabla WHERE CONDICION1 = CONDICION2

Aquí lo que estamos haciendo es llamar los datos de la nombre\_columna1 de la tabla Nombre\_Tabla y filtramos el dato utilizando el WHERE y comparando dos datos, por ejemplo:

SELECT id\_planta, cantidad FROM planta WHERE id\_planta = 4

Descripción:

En este ejemplo estamos llamando los datos id\_planta y cantidad de la tabla planta y le agregamos una cláusula WHERE para que nos muestre solo los datos en donde id\_planta sea 4.

Consultas **JOIN**

Otro tipo de consultas que se ocuparon son las consultas JOIN estas consultas sirven para poder llamar datos de una tabla y combinarla con otra tabla existen diferentes JOIN, el inner join que sirve para ver si existen datos iguales entre las dos tablas, el left join, que sirve para ver si existen datos iguales en las dos tablas, pero si no existen datos iguales en la tabla principal, las muestran en nulo (NULL) y el right join, qué es lo mismo que el left join pero en vez de ser los datos de la tabla principal es la tabla a la que estamos llamando, esto quiere decir que si los datos de la tabla que estamos llamando no coinciden con la principal estos se mostraran en null.

Una vez explicado los tipos de consultas JOIN se mostrará un ejemplo con explicación y una consulta real de la base de datos

SELECT n.nombre\_columna1,p.nombre\_columna1 FROM Nombre\_Tabla\_1 n

inner join Nombre\_Tabla\_2 p on(p.id\_tabla2 = n.id\_tabla1)

En esta consulta primero llamamos las columnas que queremos mostrar n.nombre\_columna1 y p.nombre\_columna1 se le agrega una letra y un punto para identificar de qué tabla las estamos llamando, este identificador se define al llamar la tabla principal, en este caso FROM Nombre\_Tabla\_1 n siendo n el identificador de la Nombre\_Tabla\_1 y para poder definir la otra tabla se ocupa el inner join.

El inner join nos permite llamar a la tabla que queramos siempre y cuando tengan una columna en común, y para eso se ocupan las foreign key, estas nos permiten conectar tablas con la columna que sea hayan compartido, en este caso estamos llamando a la tabla Nombre\_Tabla\_2 con p como su identificador para luego definir qué columnas son las que comparten, en este caso p.id\_tabla2 = n.id\_tabla1.

Con esta comparación MySQL sabrá si n.id\_tabla1 existe en p.id\_tabla2 y así compartir los datos de id\_tabla2 a la consulta de id\_tabla1.

La consulta en un caso real que fue utilizada para que dos tablas compartan información ocupando la foreign key compartida es así:

SELECT p.id\_pechera\_registro, p.fecha\_registro, p.Talla, c.nombre\_planta FROM pechera p inner JOIN planta c ON p.id\_planta = c.id\_planta

Descripción:

Se realiza un llamado de los datos de la tabla pechera y se le asigna un identificador, en este caso p, las columnas relacionadas con esta tabla son: p.id\_pechera\_registro, p.fecha\_registro, p.Talla .Luego hacemos un inner JOIN a la tabla planta con identificador ’c’ y comparamos si p.id\_planta de la tabla pechera es igual a c.id\_planta de la tabla planta, eso ocurre aquí: ON p.id\_planta = c.id\_planta.

Con eso ya eso podemos llamar datos de la tabla planta en este caso c.nombre\_planta para poder saber qué plantas están asociadas a las pecheras.

Consultas utilizando **CASE**

El uso del CASE nos permite realizar evaluaciones condicionales dentro de una consulta. Aquí está el formato de una consulta con un CASE:

select columna1,

CASE

WHEN columna2= 'opcion\_columna2' THEN resultado

ELSE resultado\_por\_defecto

END AS nombre\_case

from Mi\_Tabla

En esta consulta podemos apreciar que estamos llamando a columna1 y seguido el CASE, en el CASE primero partimos con un WHEN (Cuando en español) para definir una condición, en este caso la condición es columna2= 'opcion\_columna2', lo que quiere decir que cuando columna2 sea igual a 'opcion\_columna2' muestre otro resultado en la consulta.

Ese resultado se define con el THEN más el resultado que queremos, en este caso resultado y si no se cumple esa condición se ejecutará el ELSE que nos mostrará el resultado que le asignemos si la condición no se cumple en la columna2, en este caso es resultado\_por\_defecto.

Y finalizamos el case con un END para luego asignarle un nombre a esa consulta, ese nombre lo definimos con el AS más el nombre que le queremos colocar, en este caso AS nombre\_case. Finalizamos la consulta con el nombre de la tabla from Mi\_Tabla .

Ahora se mostrará una consulta real utilizada en nuestra base de datos:

SELECT nombre\_planta,

CASE WHEN estado = true THEN 'Activa' ELSE 'Inactiva' END AS estado FROM planta

Descripción:

Aquí se están llamando las columnas nombre\_planta y estado, pero ocupamos un CASE para definir el estado de la planta, cuando estado es true entonces se mostrará como activa y si no cumple la condición con el ELSE se mostrará como inactiva.

**Consultas del backend.**

Las consultas ya sean para insertar, actualizar, visualizar o eliminar datos de forma específica tienen una ligera diferencia comparadas con las que ocupamos en MySQL. En las consultas SQL, los datos que se ingresan a través del software son reemplazados por signos de interrogación (?), llamados Placeholders. Estos símbolos actúan como espacios reservados para los valores que se proporcionarán cuando se ejecute la consulta.

Estos Placeholders nos permiten poder ingresar cualquier tipo de dato en la consulta (dependiendo la consulta que se esté realizando), ya sea numérico o de texto, ya que la base de datos determina si se le agregan comillas o no al momento de ingresar algo.

Los Placeholders permiten que la consulta sea dinámica, ya que los valores que se pasan en tiempo de ejecución no están directamente en la consulta, sino que se insertan de manera segura ayudando a prevenir SQL injection, ya que los valores que se ingresan quedan como cadena de texto normales y cualquier tipo de código malicioso se ingresa como texto normal.

**Consultas select**:

En el select, al momento de ocupar un WHERE = en vez de especificar lo que queremos que nos muestre el sistema, ingresamos un Placeholders (?), este Placeholder nos permite realizar las consultas según el valor que se ingrese dentro del software, un ejemplo simple de esto es el siguiente:

SELECT id\_planta, nombre\_planta, cantidad\_asignada, kilo,

CASE WHEN estado = 1 THEN 'Activa' ELSE 'Inactiva' END AS estado

FROM planta WHERE id\_planta = ?

Aquí se está ocupando WHERE id\_planta = ?, en Placeholders es en donde va a ir el id que se ingrese utilizando la página, mostrando los datos de la planta que se haya seleccionado.

**INSERT**:

Al momento de insertar datos mediante la página nosotros ocupamos los Placeholders en la sección de los valores que vamos a ingresar.

Aquí un ejemplo:

INSERT INTO planta (nombre\_planta,cantidad\_asignada, kilo) VALUES (?, ?, ?)

Como se puede ver los Placeholders van después del VALUES y dentro de paréntesis, para así poder ingresar los datos según el campo que esté indicado.

**UPDATE**:

Para poder modificar algún dato dentro de la página, primero debemos definir que queremos modificar, dentro del software existen vistas para poder modificar según lo que el usuario seleccione, aquí es en donde trabaja los select ya mencionados, ya que al momento de seleccionar un registro dentro de la página (Si es que este tiene la opción de modificarlo), este ya va a tener su ID ya seleccionado, permitiendo visualizar los datos que queremos modificar y así no tenemos que ingresar manualmente el ID de lo que queremos modificar, ya que será automático. Aquí un ejemplo de la consulta:

UPDATE planta SET nombre\_planta = ?,cantidad\_asignada = ?, kilo = ?, estado = ? WHERE id\_planta = ?

En este caso estamos modificando una planta previamente seleccionada, los valores que estaremos ingresando están marcados con los Placeholders y como mencione anteriormente en WHERE id\_planta = ? será reemplazado por el ID del select que se hizo para mostrar los datos.

**DELETE**:

Por último, tenemos los Delete, esta consulta funciona igual que los select, ya que estaremos definiendo nosotros lo que queremos eliminar mediante el WHERE =. El software permite eliminar registros seleccionándolos de una lista o escaneándolos.

De la primera forma nosotros estaremos utilizando directamente la consulta delete, ya que estaremos seleccionando el registro manualmente junto con su ID para completar la consulta Delete.

Y de la otra forma utilizamos un select en donde se mostrará la información de lo que vamos a eliminar mediante el ID de ese registro que estemos escaneando, y con ese ID vamos a rellenar la consulta del Delete. Por ejemplo:

DELETE FROM pechera WHERE id\_pechera\_registro = ?

Nosotros seleccionamos la pechera que queremos eliminar, ya sea escaneándola en la vista específica para eliminar pecheras o seleccionándola dentro de la tabla de pecheras. Si escaneamos la pechera, se utilizará un SELECT para obtener los datos de la pechera que se va a eliminar junto con su ID. Una vez definido el ID a eliminar, se ejecuta la consulta.

Si elegimos la pechera mediante la tabla, se ejecutará la consulta con el ID asociado a la pechera que seleccionamos.

**Resumen de las consultas en el backend:**

Todas las consultas realizadas en el backend en donde se necesitaba un registro en específico fueron modificadas, los datos se reemplazan por signos de interrogación (?), conocidos como Placeholders, estos son los que reciben la información ingresada a través del software, ayudando a la fluidez y seguridad de las consultas.

**Encriptado de la contraseña**.

En nuestra base de datos se almacena mucha información valiosa para la empresa, esto nos obligó a tener que aplicar diferentes medidas de seguridad, una de ellas fue la encriptación de la contraseña del usuario.

Lo que se hizo fue crear un insert con la modificación de la contraseña, esta modificación fue agregarle SHA2, esto permite que al momento de guardar la contraseña en la tabla login se guarde de manera hasheada para que esta no pueda ser legible.

Formato del insert utilizado:

insert into login (nombre\_completo, correo, contraseña, id\_planta)

values (‘nombre del usuario’,‘correo del usuario’, SHA2(‘contraseña del usuario’, 256), id de la planta);

Aquí se puede ver que en apartado contraseña se está agregando SHA2(‘contraseña del usuario’, 256), aquí se puede ver que se agrega SHA2 con paréntesis, dentro de estos paréntesis se ingresa la contraseña que se quiere encriptar más el número 256, este número indica el nivel de cifrado de la contraseña.

El resultado de esta consulta es:

nombre\_completo: nombre del usuario

correo: correo del usuario

contraseña: ae26107355b55002a01e57bdcffef6707db69e8a66d5e2843942de403282cc6e

id\_planta: 3

Como se puede observar la contraseña se guardó (contraseña del usuario) de una forma que no es legible.

Ahora si nosotros queremos llamar esta contraseña, ya sea para poder modificarla o ingresarla en un login se realiza este select:

SELECT \*

FROM login

WHERE correo = 'correo del usuario' AND contraseña = SHA2('contraseña del usuario', 256);

En este SELECT nos permite confirmar si lo que se ingresó es correcto, ya que si el correo o la contraseña es incorrecta no se mostrará nada en la base de datos. Esta consulta es la que se ocupa para que se pueda ingresar al sistema mediante el login del software.

En nuestro caso ya dentro del backend ocupamos Placeholders (?) y queda así la consulta.

Resumen del encriptado:

Hashear la contraseña nos permite tener una mejor seguridad dentro de la base de datos, estas contraseñas se guardan de manera ilegibles y la única forma de poder descifrarla es mediante programas, ya que no es posible hacerlo directamente con MySQL, por eso se valida mediante correo en caso de pérdida.

**Conclusión de la base de datos:**

Con todo lo anterior explicado se puede tener una idea de cómo fue el desarrollo de esta base de datos, se explicó sobre todas las tablas que fueron creadas junto sus relaciones, los tipos de datos y columnas que fueron agregadas y los elementos que se utilizaron para que funcione correctamente la interacción de las tablas, como son los triggers o las claves foráneas.

Se analizó los tipos de consultas que utilizamos tanto en MySQL como en el backend sumado a los diferentes métodos de seguridad que se implementaron.

## ETAPA N°6

**Etapa de pruebas**

En esta etapa se deben realizar las pruebas necesarias de software y hardware para verificar que todo funcione correctamente y poder hacer una buena entrega al cliente, para este apartado no se consideró necesario realizar pruebas automatizadas.

**Software**

Se realizan distintas pruebas de funcionalidad dentro de cada uno de los módulos del sistema, verificando que cada uno de estos entregue el resultado esperado. El sistema cuenta con módulos o vistas en las cuales se tiene que realizar una lectura antes de realizar otra acción como lo son las vistas de EliminarPechera, RegistrarPechera, Lavados, etc. Vistas en las cuales los resultados fueron los esperados teniendo en las versiones más tempranas algunos detalles como, por ejemplo, que en la vista se mostraba constantemente el último UID leído, lo que podía traer problemas al registrar lavados o registrar nuevas pecheras al sistema. En las otras vistas se tuvo de igual manera problemas, por ejemplo, con el apartado de las rutas privadas en un momento no funcionaba la autenticación con los datos de la base de datos, una vez solucionado eso nos dimos cuenta de que todo el estilo se veía afectado dentro del sistema dejando un sistema desorganizado y para nada atractivo. Una vez fueron detectados cada problema surgido en esta fase fue resuelto cuanto antes para poder entregar un buen producto.

Hardware

En la etapa de la construcción del hardware se tuvieron problemas por los cables utilizados en primera instancia, los cuales quedaban sueltos y sin darnos cuenta se salían de vez en cuando, haciendo que no podamos probar de manera correcta los procesos constantemente, por lo que se tuvieron que cambiar los cables. Una vez terminada la construcción y configuración del hardware este fue probado dentro del mismo IDE de programación, dando buenos resultados al leer de manera continua los chips y capturando de manera correcta los UID. Al ser integrado al sistema, probamos las funciones donde es necesario escanear chips, obteniendo resultados satisfactorios ya que obtuvimos la lectura de manera esperada.

## ETAPA N°7

**Marcha blanca y entrega final**

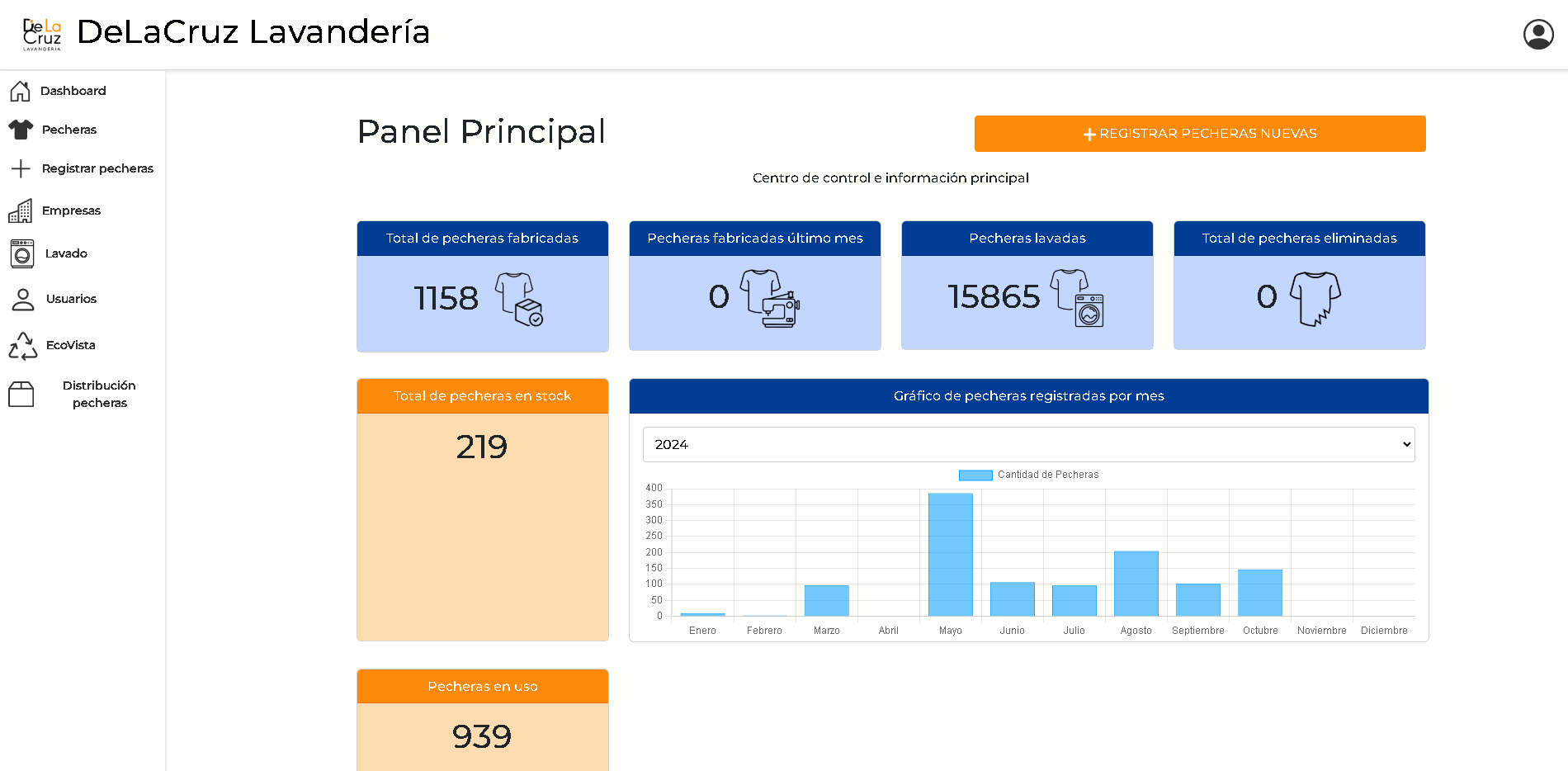
Con el visto bueno luego de varias reuniones con el equipo de la empresa y mostrándole cada avance y aplicando las mejoras propuestas por el personal, pudimos concretar el inicio de una marcha blanca en la cual surgieron algunos inconvenientes, principalmente con el tema de los datos, que en un principio no se mostraban correctamente y se buscó solucionar este problema de la manera más rápida posible, luego uno de los inconvenientes que surgieron fue la ralentización del computador, culpa de demasiados programas instalados y siendo ejecutados al mismo tiempo por lo que se tuvo que instalar el sistema en otro computador provisorio para evitar el corte de las actividades, una vez el dispositivo original se arregló se instaló nuevamente el sistema y las actividades del personal siguieron correctamente con un funcionamiento óptimo del software, por lo que se decidió hacer la entrega final del software después del feedback positivo por parte de los usuarios, entregando el sistema como tal, acompañado de un video explicativo sobre el funcionamiento del propio programa, siendo este último junto con la versión de la aplicación, bien recibidas por parte de los usuarios.

# RESULTADOS

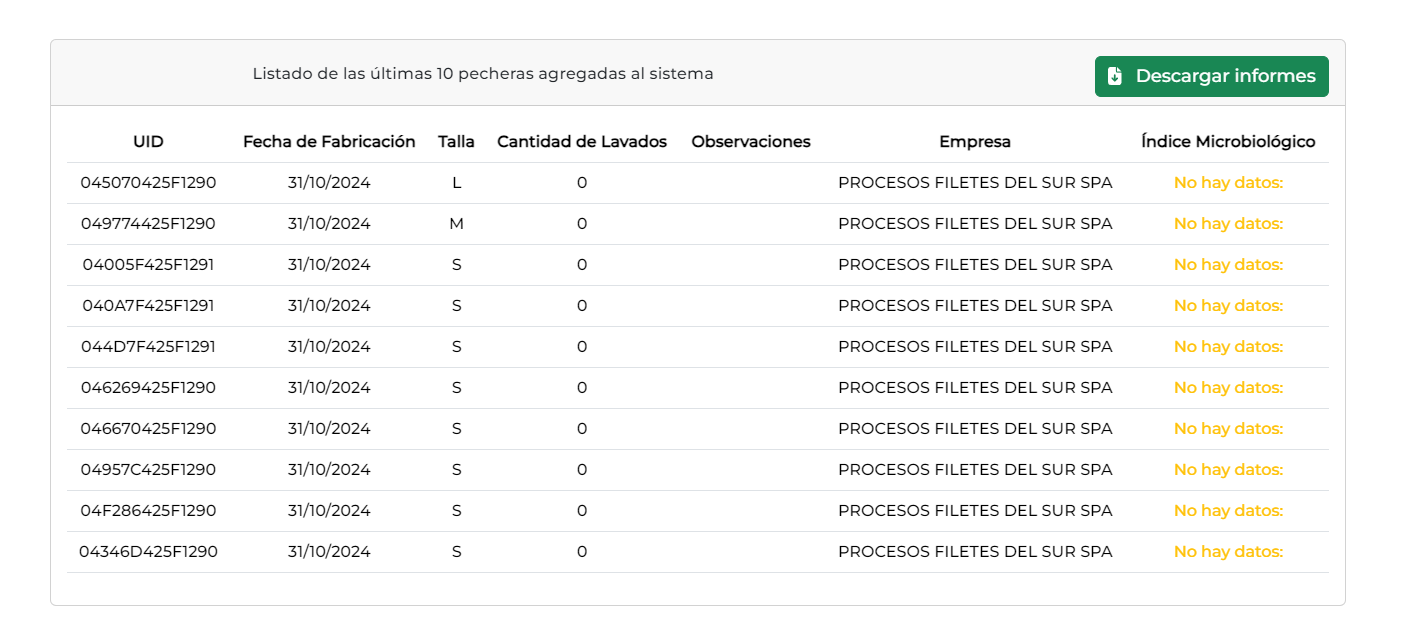
Después de haber realizado el trabajo basado en los requerimientos y objetivos iniciales, habiendo superado todos los inconvenientes que surgieron durante el desarrollo del propio sistema. Como equipo logramos llegar a un producto del cuál nosotros y el cliente nos sentimos satisfechos, ya que permite el cumplimiento de las necesidades de la empresa, facilitando y optimizando el trabajo del personal. Profesionalmente, el proyecto realizado, nos acercó bastante a lo que sería la experiencia de trabajo en el mundo laboral además de a nuestros intereses profesionales.

**Dashboard**: Muestra un resumen del total de pecheras fabricadas, fabricadas el último mes, registro de lavados históricos, pecheras en uso, en stock, las eliminadas o dadas de bajas y por último un gráfico que representa las pecheras registradas por mes.

Estos datos fueron solicitados directamente por el cliente, ya que son los más relevantes para tener un seguimiento claro de su avance y crecimiento.

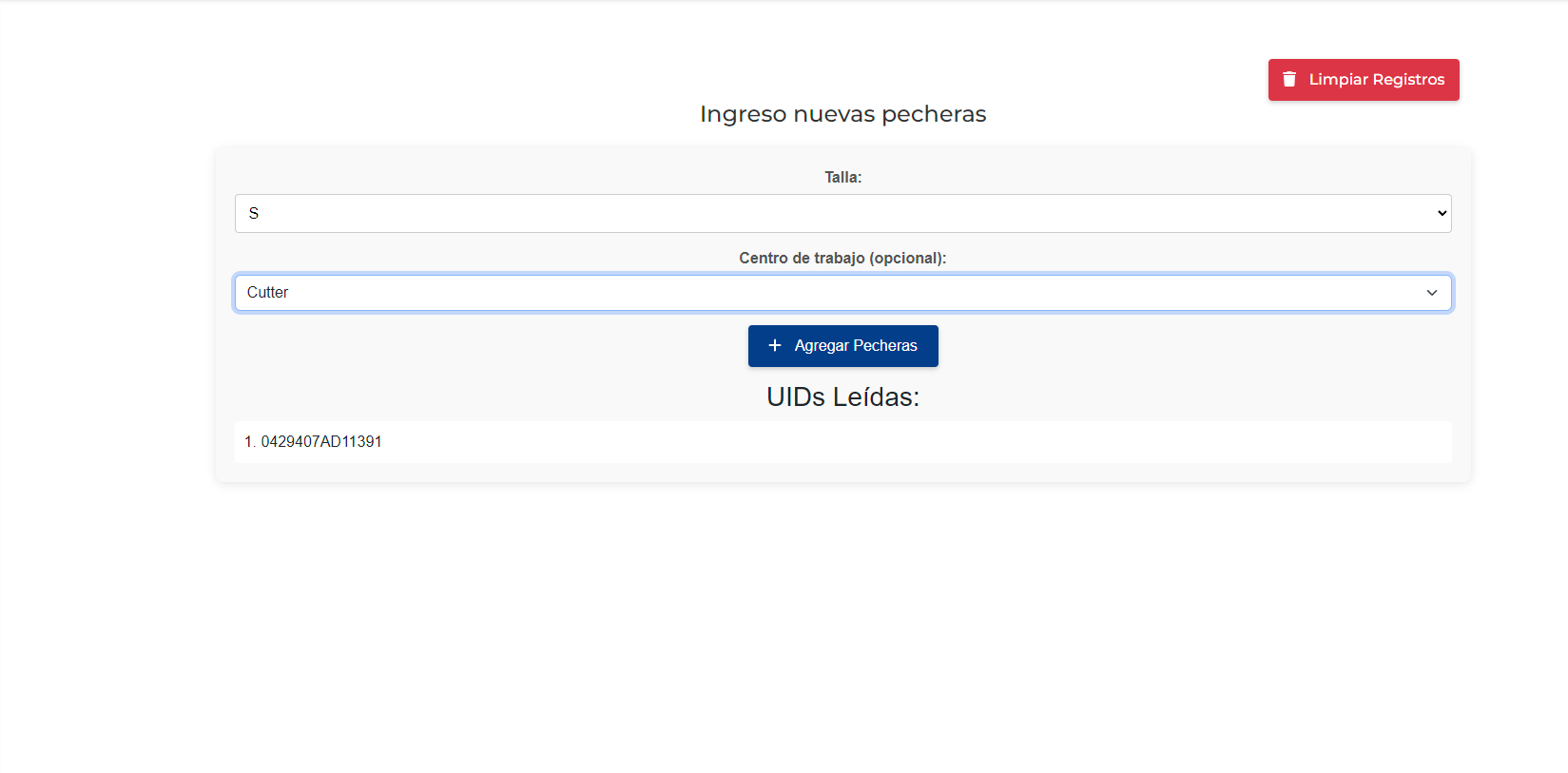


##### Figura 11: Software DeLaCruz



##### Figura 12: Software DeLaCruz

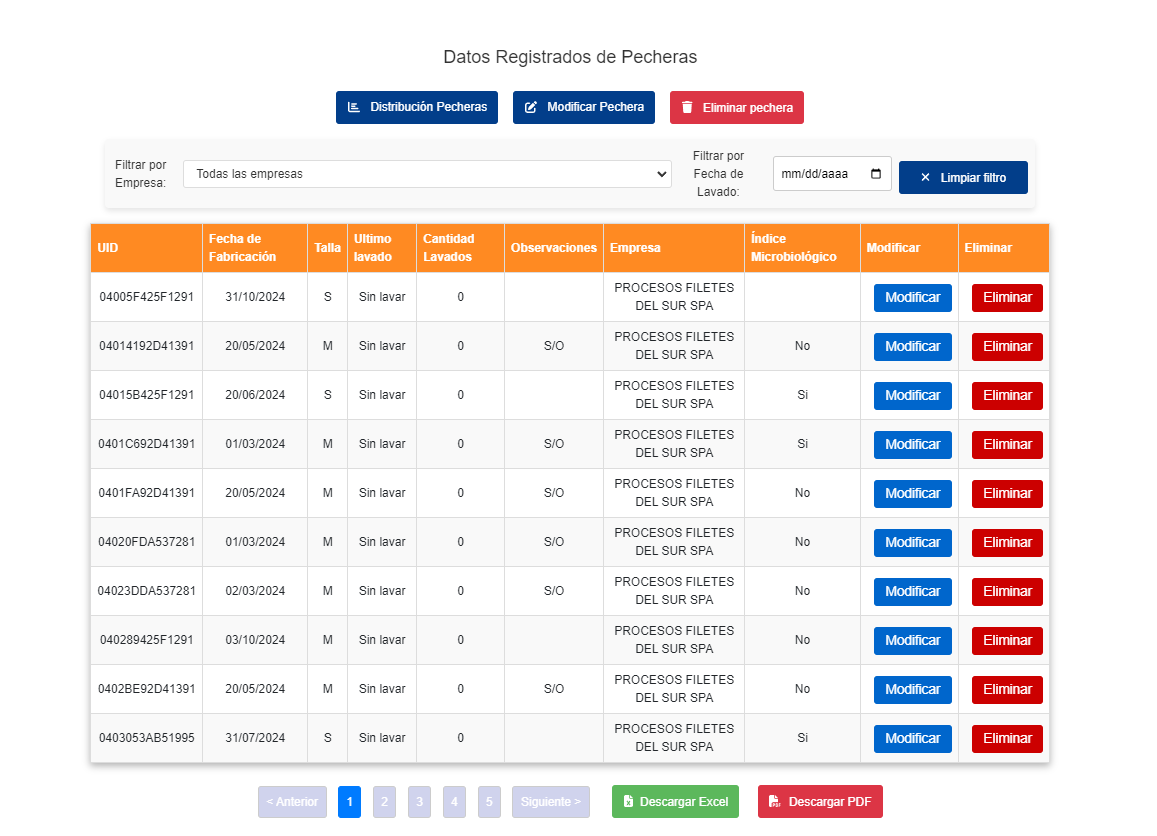
**Registro pechera**: Esta vista se utiliza en conjunto con el lector para realizar los ingresos de nuevas pecheras al sistema, se debe asignar la talla de forma obligatoria y el centro de trabajo de forma opcional.



##### Figura 13: Software DeLaCruz

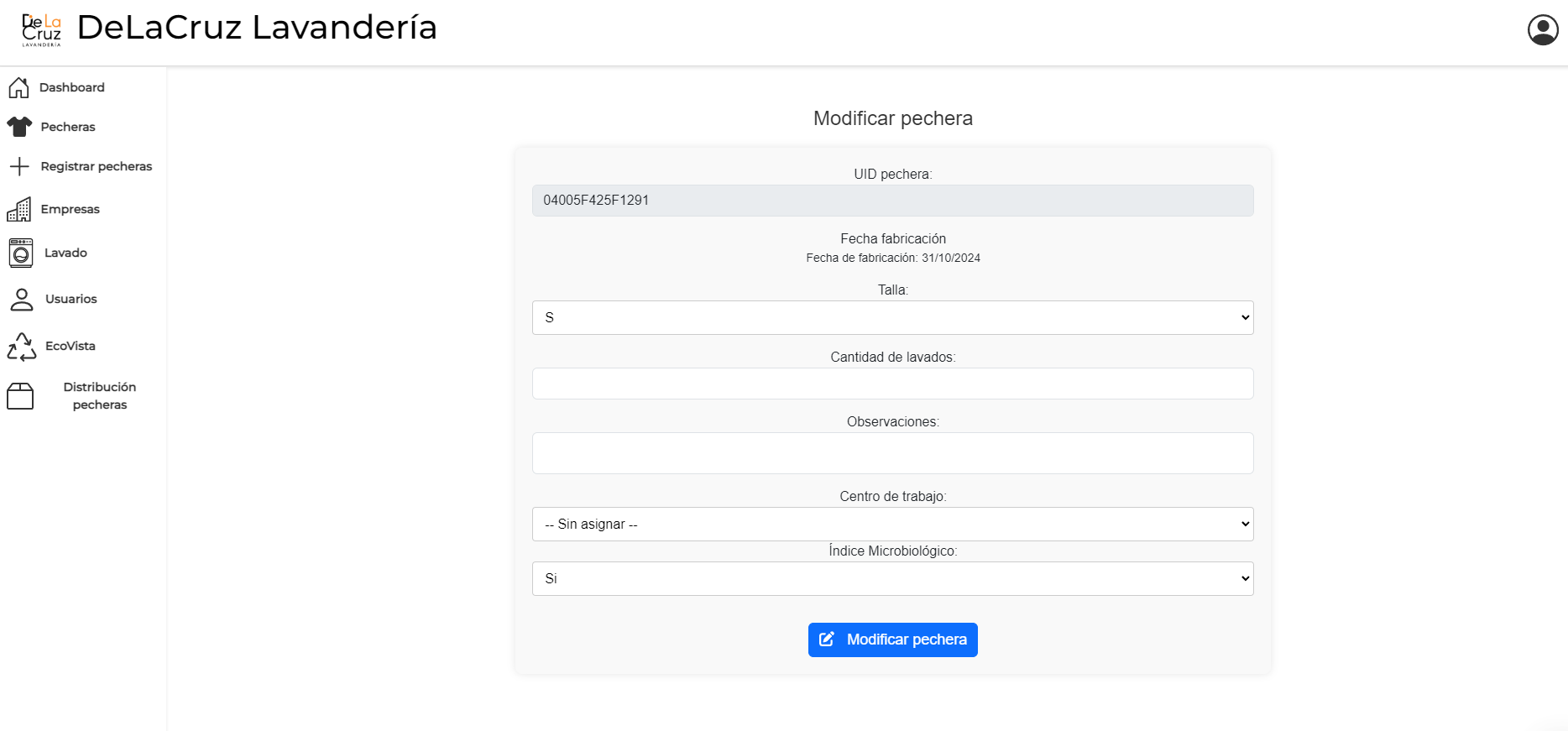
**Pecheras**: Esta vista muestra todos los datos de las pecheras, al recién registrarla solo mostrará UID, fecha fabricación, talla y opcionalmente el centro de trabajo, los demás campos se irán rellenando de acuerdo sea necesario o se vaya ingresando nueva información.

Además, se puede filtrar los datos tanto por empresa como por fecha de lavado y descargar informes de estos datos filtrados.



##### Figura 14: Software DeLaCruz

**Modificar pechera**: Aquí sirve para registrar información faltante o modificar algún dato erróneo a excepción del Uy fecha fabricación esto se puede realizar seleccionando el botón que está en la tabla de datos en la vista “Pechera”.



##### Figura 15: Software DeLaCruz

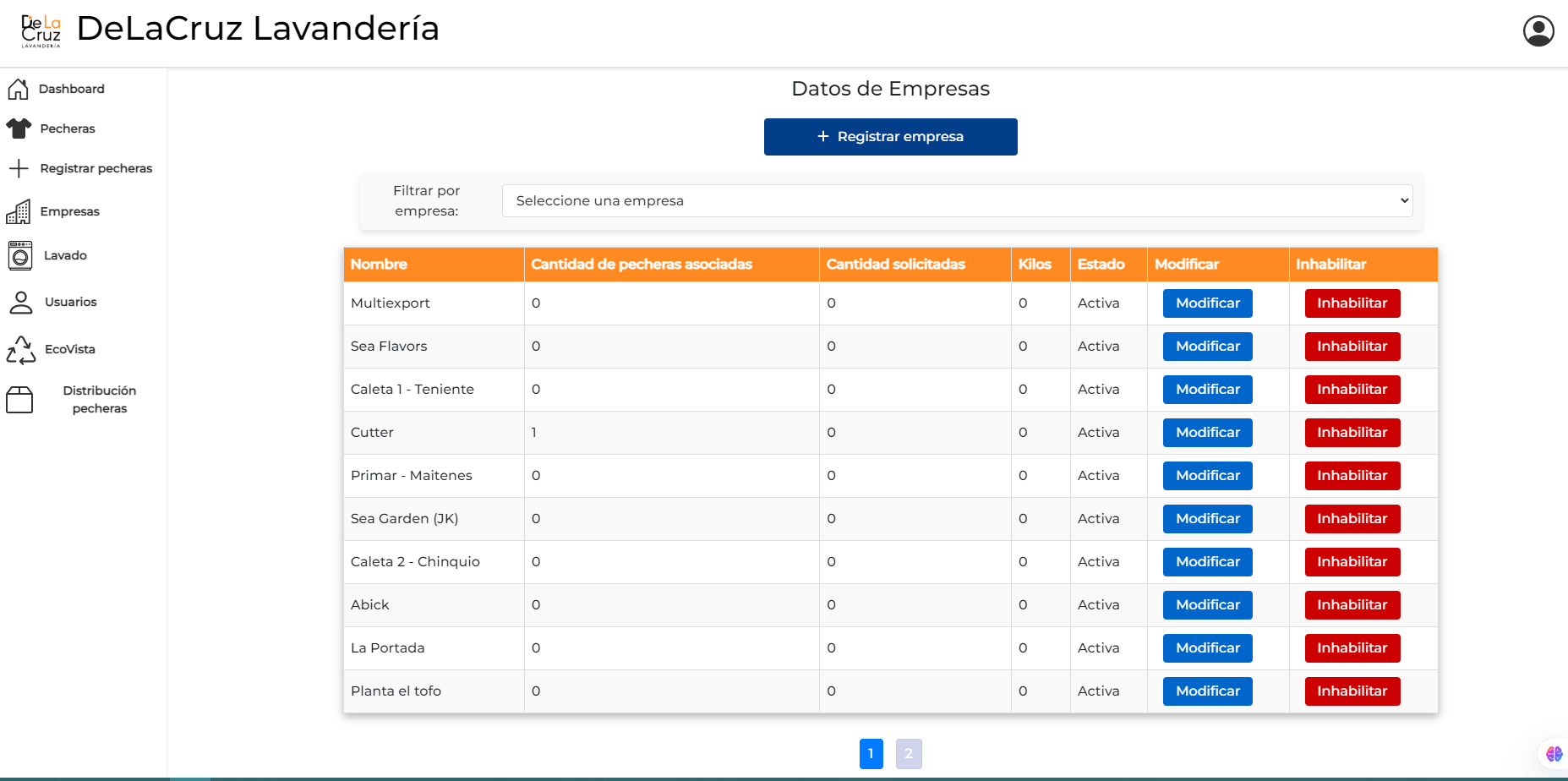
**Eliminar pechera**: En esta vista se pueden eliminar pecheras específicas por medio de la lectura de su chip RFID



##### Figura 16: Software DeLaCruz

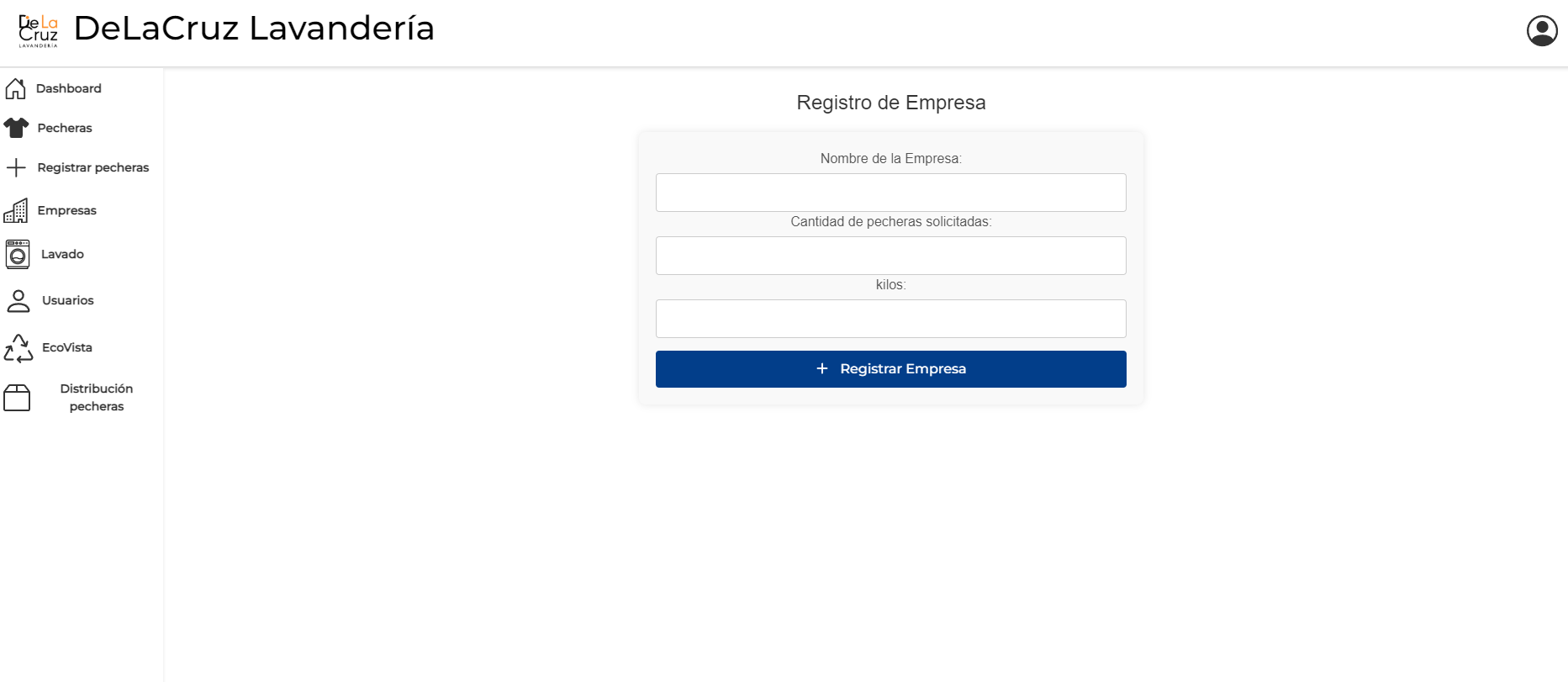
**Empresa**: Muestra la información de los centros de costos que están vinculados a la lavandería y a futuro se les asignan pecheras, esta también cuenta con un filtro por empresa.

A esta vista se vinculan el registro y la modificación de los datos de las empresas.



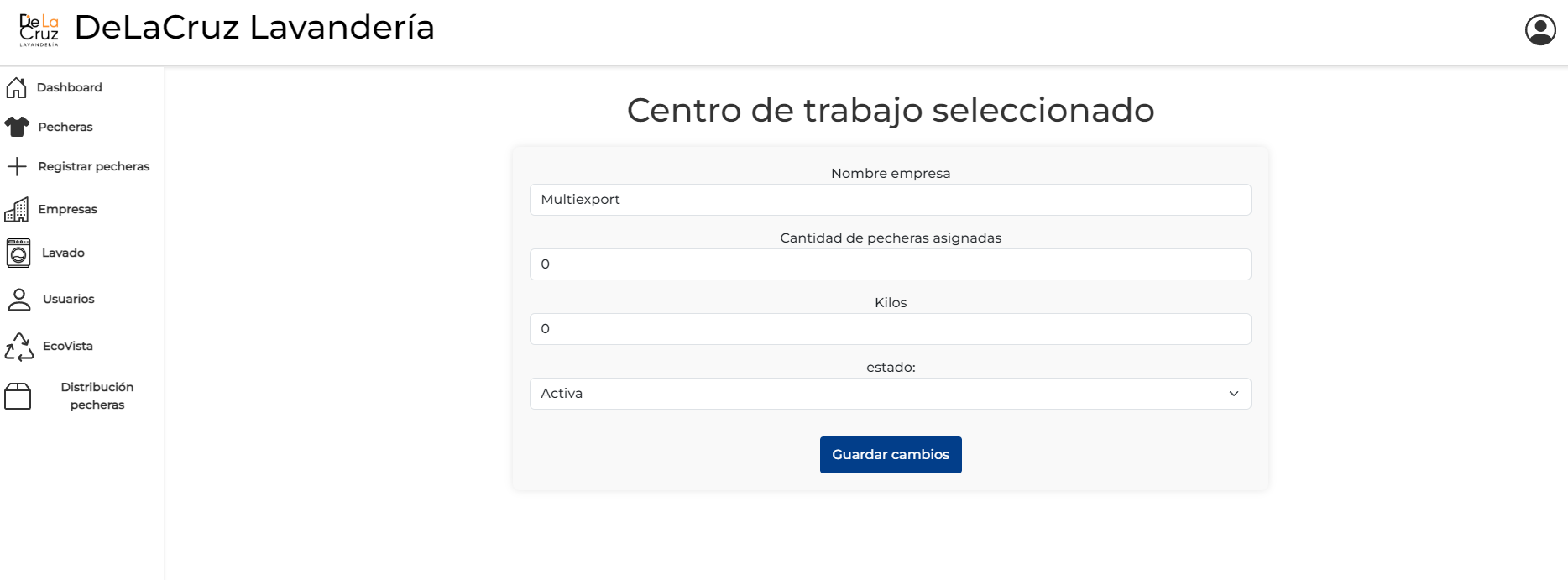
##### Figura 17: Software DeLaCruz

**Registro empresa**



##### Figura 18: Software DeLaCruz

**Modificación empresa**



##### Figura 19: Software DeLaCruz

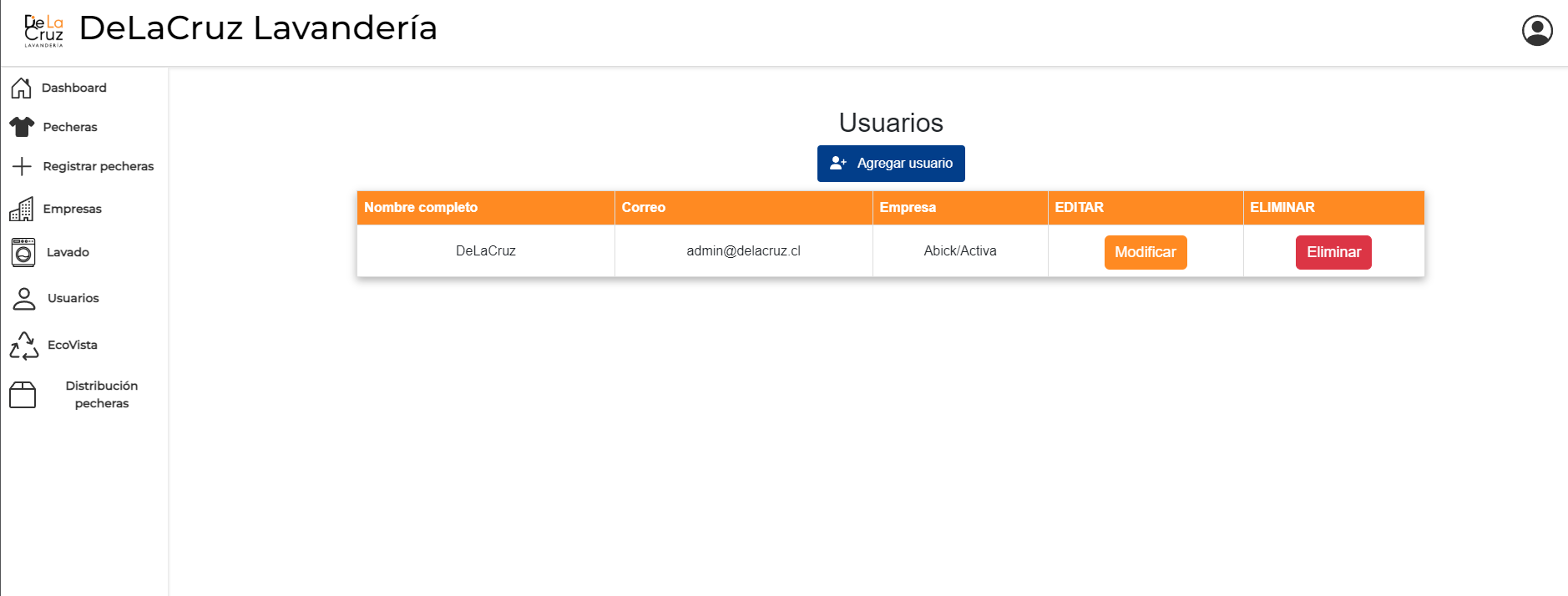
**Registro lavado**: Al igual que registró pechera esa puede registrar usando el dispositivo de lectura para identificar la pechera y sumarle un lavado



##### Figura 20: Software DeLaCruz

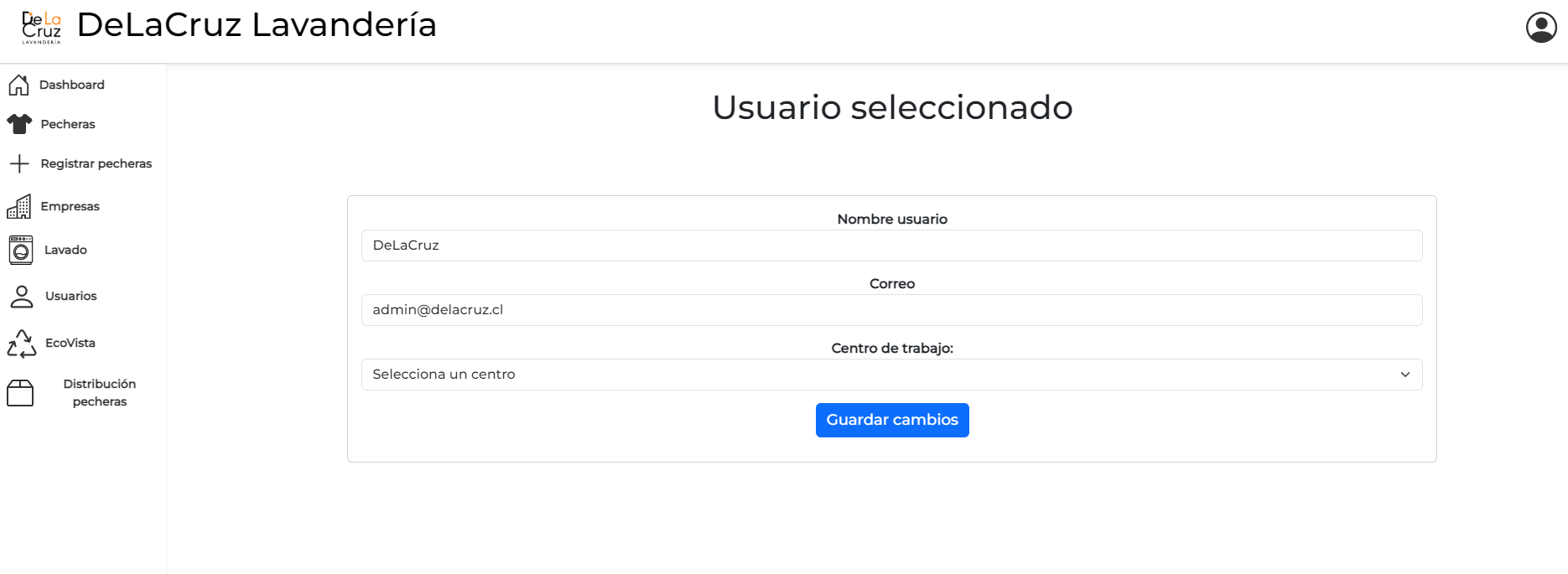
**Usuario**: Acá se mostrará la lista de usuarios registrados que podrán acceder al sistema, se muestra su nombre, correo y empresa a la que pertenece.

A esta vista se le vinculan otras dos que son modificar que permite modificar algún dato y el registro que será para ingresar nuevos usuarios.



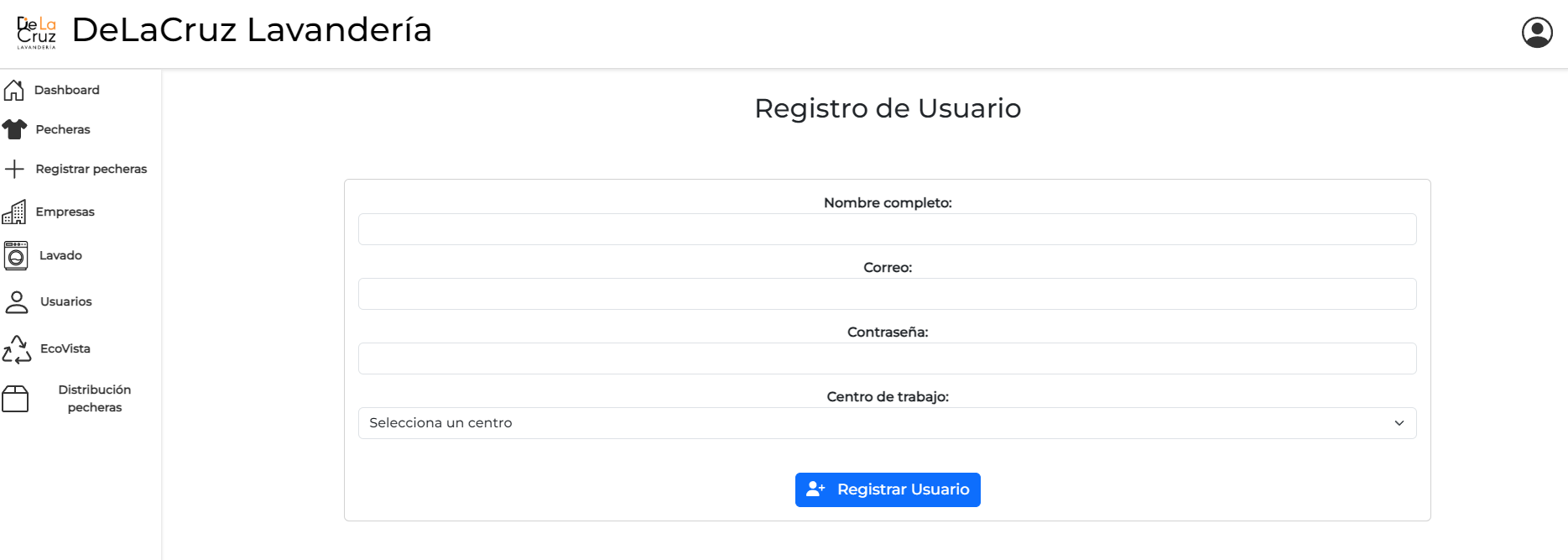
##### Figura 21: Software DeLaCruz

**Modificar usuario**



##### Figura 22: Software DeLaCruz

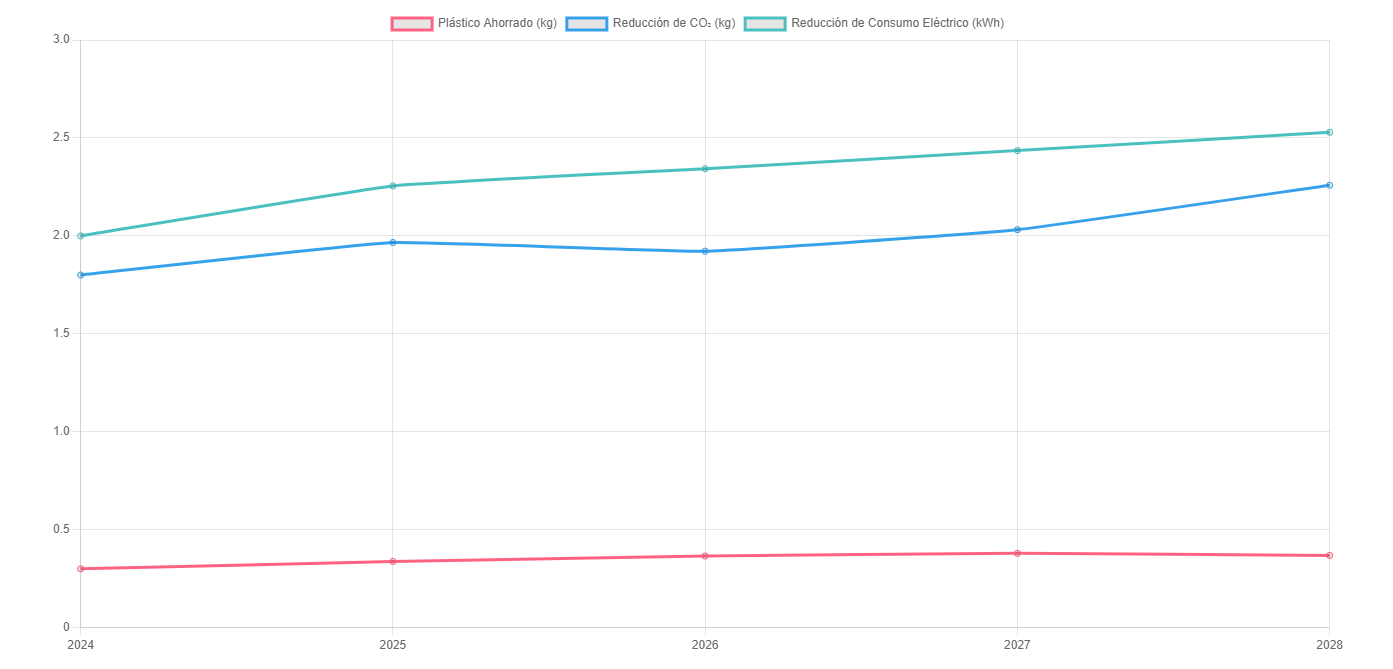
**Registro usuario**



##### Figura 23: Software DeLaCruz

**EcoVista:** En esta vista se busca mostrar cómo afecta el uso de pecheras reutilizables a lo largo de los años, se puede analizar gracias a que los datos son asignados manualmente lo que permite tener un mejor entendimiento y guiarse de acuerdo con las variaciones que se presentan a lo largo de los años.

Figura 24: Software DeLaCruz



##### Figura 25: Software DeLaCruz

**Distribución pechera**: En esta vista se pueden centros de costos a pecheras que aún no han sido distribuidas previamente en el registro, esto funciona con el dispositivo.



##### Figura 26: Software DeLaCruz

# CONCLUSIONES Y RECOMENDACIONES

## Conclusiones

Pilar Bonnault

El ser parte de este proyecto fue una experiencia bastante buena y emocionante, que se acerca bastante al mundo laboral al que nos debemos insertar a futuro, además que permitieron poner en práctica los conocimientos adquiridos a lo largo de la carrera, aprender a manejar situaciones problemáticas que se fueron dando a lo largo del desarrollo, durante este tiempo se utilizaron diferentes tecnologías tanto conocidas como nuevas en el caso de React y Node.js que se usaron para desarrollar el software.

Con este proyecto se le pudo dar a la lavandería un sistema robusto en el que podrán manejar el registro y seguimiento de sus pecheras de una forma eficaz y rápida y así incentivar el su compromiso con el medio ambiente.

Nicolás González

Dentro del tiempo trabajado en este proyecto puedo decir que fue una experiencia realmente grata, me permitió desarrollar nuevas habilidades, conocer nuevas tecnologías y acercarme más a una experiencia real de trabajo grupal y con un cliente al cuál se le debe ayudar a cubrir una o varias necesidades. Software DeLaCruz creo que es un sistema bastante completo, que permite cumplir los objetivos del proyecto al cuál la empresa De La Cruz Lavandería está apostando, dando una optimización igualmente al trabajo del personal de esta empresa, dándole la opción de completar sus tareas de una manera más cómoda y eficiente, el sistema creado tiene mucho potencial para ser actualizado y perfeccionado en un futuro por cualquier desarrollador ya que se buscó mantener una ética y metodología de trabajo adaptable a cualquier persona al momento de programar. En general el programa “Software DeLaCruz” cumple con todos los requerimientos solicitados y está diseñado para ser actualizado y mejorado.

Christofer Ruiz

El proyecto del software DeLaCruz fue una experiencia buena, ya que es lo más cercano a un trabajo real. Desarrolle nuevas habilidades de comunicación gracias a las reuniones que tuvimos con el cliente para conocer lo que ellos querían y así entender de mejor manera cómo manejar problemas y presentar soluciones, gestionar requerimientos entre otras cosas y también desarrolle mis conocimientos en los campos que más me interesaban, campos que también quiero utilizar laboralmente, como son la base de datos con el MySql y el backend.

## Limitaciones y Recomendaciones.

Durante el desarrollo del proyecto se presentaron una serie de limitaciones como:

1. La falta de reuniones con el personal que usaría directamente el software, lo que nos provocó muchos cambios previos a la puesta en marcha, para mejorar esto le planteamos a la lavandería que durante las reuniones estén sí o sí las personas que lo usaran.
2. La gestión de las compras necesarias para avanzar en el proyecto, ya que muchas veces debían consultar presupuesto y esperar que se nos enviara del dinero, para mejorar esto lo mejor sería presentar un presupuesto previo con todo lo necesario para cumplir las metas requeridas.
3. Falta de información, muchas veces le solicitamos a la lavandería información que ellos requerían en el sistema que no fue entregada a tiempo, para poder mejorar esto les pedimos que nos hagan envió de todo dato requerido en el mismo momento que fue solicitado.

# BIBLIOGRAFÍA

# LINKOGRAFÍA

Ingeniería en Informática – Duoc UC. (2024). Duoc.cl. <https://www.duoc.cl/carreras/ingenieria-informatica/>

https://www.facebook.com/APDasociacion. (2024, April 9). ¿Qué es la metodología Scrum y cómo aplicarla? APD España. <https://www.apd.es/metodologia-scrum-que-es/>

Visual Studio: IDE y Editor de código para desarrolladores de software y Teams. (2020). Visual Studio. <https://visualstudio.microsoft.com/es/#vscode-section>

Andrus, B. (2024, March 4). ¿Qué Es Node.js? Una Introducción Completa + Casos de Uso. Guías Para Sitios Web, Tips & Conocimiento; DreamHost. <https://www.dreamhost.com/blog/es/que-es-node-js-introduccion-completa/>

mijacobs. (2023, October 5). ¿Qué es Git? - Azure DevOps. Microsoft.com. <https://learn.microsoft.com/es-es/devops/develop/git/what-is-git>

‌Deyimar A. (2020, February 4). Qué es React: definición, características y funcionamiento. Tutoriales Hostinger. <https://www.hostinger.es/tutoriales/que-es-react>

Esteban, D. (2024, June 15). Agrega iconos de Font Awesome en tu sitio Web - Diego Esteban C 🧢 - Medium. Medium. <https://medium.com/@diego.coder/agrega-iconos-de-font-awesome-en-tu-sitio-web-49660efc50ee#:~:text=Font%20Awesome%20es%20una%20biblioteca,de%20manera%20f%C3%A1cil%20y%20r%C3%A1pida>.

Gonzales, D. (2023, May 19). ¿Por qué los desarrolladores adoran Axios? Meltstudio. <https://www.meltstudio.co/post/por-que-los-desarrolladores-adoran-axios#:~:text=Axios%20es%20una%20librer%C3%ADa%20de,desde%20un%20servidor%20usando%20Node>.

Santander Universidades. (2023, June 26). Qué es bootstrap | Blog Santander Open Academy. Santanderopenacademy.com; Santander Open Academy. <https://www.santanderopenacademy.com/es/blog/que-es-bootstrap.html>

https://www.facebook.com/grokkeepcoding. (2023, October 23). Bootstrap en proyectos React: ¿Cómo integrar la librería? KeepCoding Bootcamps. <https://keepcoding.io/blog/bootstrap-en-proyectos-react/#:~:text=La%20combinaci%C3%B3n%20de%20React%20y,tu%20aplicaci%C3%B3n%20de%20manera%20eficiente>.

Cabrera, A. (2024, July 17). Qué es React Router DOM, Rutas Dinámicas y Anidadas, Navegación y Redirección. 10Code Software Design. <https://10code.es/react-router-dom/#:~:text=React%20Router%20DOM%20es%20una%20biblioteca%20que%20permite%20implementar%20el,sin%20recargar%20la%20p%C3%A1gina%20completa>.

Serial \ Libraries \ Wiring. (2024). Wiring.org.co. <https://wiring.org.co/reference/es/libraries/Serial/index.html#:~:text=La%20libreria%20Serial%20de%20Wiring,en%20tu%20programa%20de%20Wiring>.

xlsx-style. (2016, May 25). Npm. <https://www.npmjs.com/package/xlsx-style>

dotnet-bot. (2024). SerialPort Clase (System.IO.Ports). Microsoft.com. <https://learn.microsoft.com/es-es/dotnet/api/system.io.ports.serialport?view=netframework-4.8.1>

Llamas, L. (2024, April 29). Cómo usar una base de datos MySQL con Node.js. Luis Llamas. <https://www.luisllamas.es/como-usar-mysql-con-nodejs/>

Estructura de archivos – React. (2021). Reactjs.org. <https://es.legacy.reactjs.org/docs/faq-structure.html>

Frontend y backend: qué son, en qué se diferencian y ejemplos. (2022). Hubspot.es. <https://doi.org/109517657415/1730819309594>

de, A. (2021). Crear nuestro primer proyecto en NodeJs - Taller de NodeJS. Github.io. <https://aulasoftwarelibre.github.io/taller-nodeJs/crearProyecto/>

¿Qué es Arduino? | Arduino.cl - Compra tu Arduino en Línea. (2014, November 20). Arduino.cl - Compra Tu Arduino En Línea. <https://arduino.cl/que-es-arduino/#elementor-toc__heading-anchor-0>

Electrónica Embajadores. (2022). Arduino MKR WiFi 1010 - Módulo Conectividad WiFi y Autenticación Criptográfica - ABX00023. Electrónica Embajadores. <https://www.electronicaembajadores.com/es/Productos/Detalle/LCA3105/modulos-electronicos/arduino-mkr/arduino-mkr-wifi-1010-modulo-conectividad-wifi-y-autenticacion-criptografica-abx00023/>

Software de Arduino | Arduino.cl - Compra tu Arduino en Línea. (2014, September 26). Arduino.cl - Compra Tu Arduino En Línea. <https://arduino.cl/programacion/>

Tinkercad. Dando volumen a las ideas - INTEF. (2022, March 8). INTEF. <https://intef.es/observatorio_tecno/tinkercad-dando-volumen-a-las-ideas/#:~:text=Tinkercad%20es%20una%20herramienta%20online,la%20posibilidad%20de%20invitar%20a>

UltiMaker Cura. (2024, November 6). UltiMaker. <https://ultimaker.com/es/software/ultimaker-cura/>

‌kit. (2018, August 18). Arduino desde cero en Español - Capítulo 40 - RFID RC522 kit lector, tarjeta, llavero 13.56 Mhz. YouTube. <https://www.youtube.com/watch?v=LvRfxGTUEpE>

pablo-sampaio. (2022, April 11). GitHub - pablo-sampaio/easy\_mfrc522: Library to allow easily reading and writing to RFID tags using MFRC522 module. GitHub. <https://github.com/pablo-sampaio/easy_mfrc522>

Arquitectura en Capas. (2021). Reactiveprogramming.io. <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas>

###### **ANEXO A:**

[**Anexo Carta Gantt**](https://duoccl0-my.sharepoint.com/:x:/r/personal/chr_ruiz_duocuc_cl/_layouts/15/Doc.aspx?sourcedoc=%7B06418261-8CAE-4C04-89B4-FF6BB9207189%7D&file=Diagrama%20de%20Gantt%20Capstone.xlsx&fromShare=true&action=default&mobileredirect=true)

###### **ANEXO B:**

[**Anexo Sprint Backlog**](https://docs.google.com/spreadsheets/d/1PCJKVoJEZ9SMfDNv9huzM7L5iVRHqtYV/edit?gid=1663657871#gid=1663657871)

###### 