UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Diseño de sistema de anaqueles de autoservicio para tráfico de productos operado por aplicación móvil

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Morales Anaya Norman Rodrigo

DIRECTOR DE TESIS

Dr. Jaime Baltazar Morales Sandoval

Ciudad Universitaria, Cd. Mx., 2019

Índice desglosado:

Capítulo 1 Introducción

1.1 Antecedentes:

Actualmente la automatización de procesos es cada vez más común, ya que la tecnología nos brinda muchas herramientas para sustituir en tareas rutinarias, hasta cierto punto, la mano de obra humana.

Tenemos como ejemplo la automatización que realizan los bancos al implementar cajeros automáticos en los cuales ya se pueden hacer transacciones más complejas que anteriormente solo podía realizar algún empleado, como el depósito o transferencia de dinero a otras cuentas. Pero la automatización bancaria no solo se enfocó en sus cajeros automáticos, puesto que desarrollaron aplicaciones para móviles con muchas funcionalidades que antes sólo se podían hacer asistiendo a un banco.

Como segundo ejemplo de automatización tenemos las máquinas expendedoras de todo tipo de productos en ciudades japonesas. Estas máquinas han logrado abarcar gran parte de los artículos personales que utilizan sus ciudadanos, ya que a diferencia de nuestro país en el cual las máquinas expendedoras solo ofrecen productos como dulces o refrescos, en Japón son utilizadas para vender ropa, sombrillas, revistas, fruta e incluso hasta comida rápida.

Estos ejemplos son indicadores de que podría llegar el punto en el cual la intervención de la mano humana, para la venta, compra y entrega de productos al igual que para realizar transacciones o trabajos pueda ser muy mínima, gracias a que habrá máquinas, aplicaciones, robots o tecnología en general que liberen casi totalmente al ser humano de esas tareas.

Los empleados dedicados a esas tareas no solo implican un gasto continuo, también representan un riesgo latente porque pueden cometer errores o generar problemas al momento de realizar un trabajo o una transacción, dado que pueden cambiar sus actitudes para atender a un usuario dependiendo de su estado de ánimo y tienen un horario fijo que puede terminar afectando al usuario. Las máquinas en cambio, pueden trabajar las 24 horas del día, sin que se tenga que pagar horas extras u otras prestaciones. Una maquina funcionando bien jamás ofenderá a los clientes y siempre atenderá sin tener cambios de humor, generando hasta cierto punto mayor confianza que una persona. Estos errores y problemas están presentes en muchos negocios en nuestro país, como en las tiendas, las lavanderías, papelerías, fondas, e incluso servicios de paquetería. Este último ejemplo es una de las mejores formas de ejemplificar el por qué se trata de automatizar las tareas.

Hoy en día las ventas por internet son cada vez más comunes, y la única manera de entregar los productos de manera segura, rápida y eficiente es entregándolos personalmente o realizando un envío por paquetería. Las entregas personales, pueden resultar muy difíciles debido a que frecuentemente los horarios de los involucrados en la entrega no coinciden, causando que la transacción no se complete y/o se recurra a realizar el envío varias veces. Por otro lado, los servicios de paquetería pueden resultar muy eficientes para envíos a otros estados o incluso envíos internacionales, pero cuando estos envíos son o ya están en la misma ciudad del receptor, suelen tardar hasta dos días en llegar al destinatario generando un costo extra al producto, que muchas de las veces ya no resulta conveniente para el vendedor o para el comprador.

Por ello una manera de mejorar la forma en la cual se compra, se vende o se da mantenimiento a productos, es mediante la automatización, quitando en medida de lo posible el problema de la sincronización humana, tal y como lo muestran los bancos y las máquinas expendedoras japonesas.

El problema de instalar anaqueles que permitan el intercambio de objetos entre proveedores de partes o servicios y los usuarios finales o clientes, se plantea en el entorno del comercio mexicano actual. Donde estos servicios no cuentan con partidas fuertes para justificar equipos y componentes de importación, además de costos de mantenimiento que involucren regalías o viajes internacionales de técnicos. Por ello se especifica la idea de analizar la viabilidad de desarrollar éstos equipos con elementos, tanto de partes como de servicios, de fácil acceso en el país.

1.2 **Posibles soluciones:**

Para lograr la automatización de los casilleros que se acaban de describir genéricamente, se plantea emplear varias tecnologías que actualmente tienen mucho trabajo y tiempo en el mercado, tales como placas desarrollo, aplicaciones web, aplicaciones para teléfonos móviles etc.
La primera solución se puede basar en algún tipo de registro electrónico para los casilleros y las personas involucradas en la transacción de modo que aquellos que tengan su registro, puedan acceder por completo a los casilleros empleado cerraduras electro-mecánicas. Pero esto resulta peligroso si no se tiene un orden y controles de los casilleros además de algún registro histórico de la evolución de cada operación que permita rastrear las acciones de cada uno de los usuarios. Tanto los que reciban y los que entreguen los productos deben dejar rastros para resolver posibles conflictos, además de que se tenga seguridad extra para los casilleros para minimizar robos o actos vandálicos. Los registros quedan en el sitio de los casilleros más un respaldo periódico por los operadores o responsables del funcionamiento de ellos.
La segunda herramienta que podría ser útil es el empleo de alguna tarjeta magnética, la cerradura eléctrica y alguna placa de desarrollo o micro controlador. La placa de desarrollo y el micro controlador controlarían las cerraduras y el lector de tarjetas y serían las encargadas de almacenar los datos de estas tarjetas y mantener un registro de los involucrados en las transacciones. El problema con esta tecnología es que cada usuario debería tener su tarjeta magnética generando un gasto fuerte para los usuarios y en caso de extravío resultaría más difícil de configurar el sistema completo, también tiene el inconveniente de no poder generar un registro de las personas que usan el sistema. Pero en general podría ser un buen sistema.
El lector de huella dactilar o sensor biométrico, es una herramienta buena y bastante empleada desde hace ya algunos años, pues permite el registro de varios usuarios. La lectura, rapidez y seguridad de estos sensores es bastante buena, haciendo que esta herramienta pueda ser muy útil para generar registros de acceso para los usuarios.
Otra forma en la cual se pueden diseñar los casilleros, es usando herramientas tecnológicas bastante básicas tales como una cerradura eléctrica que funcione como seguro en los casilleros, un teclado matricial, un LCD de 16x2 y alguna placa de desarrollo o un micro controlador. El teclado, el LCD y la cerradura serían controlados por la placa de desarrollo o el micro controlador que también será encargado de almacenar todos los datos de las personas que realizarán alguna transacción en los casilleros, y de guardar los datos completos de las transacciones realizadas. El teclado y el LCD tendrán la función de mostrar un menú para los usuarios y de elegir las opciones para el almacenamiento de diferentes tipos de datos.

Como última herramienta y empleando elementos un poco más avanzados que requieren menos hardware, se puede desarrollar alguna aplicación web, la cual empleando internet en cualquier dispositivo, mantenga una base de datos que dé servicio a los usuarios y controle las operaciones que tengan en proceso tanto proveedores como clientes. El funcionamiento correcto de los casilleros automáticos podrá ser supervisados por una empresa o bien por los dueños que tendrán el control total sobre los casilleros, teniendo acceso a estos desde cualquier parte del mundo.
Convirtiendo estas últimas dos formas en las opciones más viables para el desarrollo del sistema. Sin embargo, el propósito de éste trabajo se establece a continuación.

1.3 **Propósito de éste proyecto**:

Se planteó ésta investigación con el propósito de estimar la viabilidad de diseñar y construir comercialmente, para las micro-empresas nacionales, anaqueles automatizados y reconfigurables que permitan el intercambio asíncrono de bienes y servicios entre proveedores y clientes.

Como hemos visto, éstas funciones se parecen inicialmente a los servicios que prestan muchas de las máquinas expendedoras de bebidas, alimentos u objetos como juguetes o regalos; sin embargo, existen fuertes diferencias en los requerimientos de diseño.
El proceso de intercambio de un servicio u objeto puede darse sin considerar la intervención de éstos dispositivos o máquinas, que aquí llamaré MIBoS (para asignar un acrónimo en referencia a "máquinas de intercambio de bienes o servicios"), mientras que a las "máquinas expendedoras" las referiré como ME. Si no se usa la MIBoS resulta entonces, indispensable la sincronización entre proveedor y cliente, como se hace regularmente.

Actualmente, es más usual que la negociación de un bien o servicio se inicie por Internet o la asistencia a un sitio donde el cliente recibe de alguna manera la información básica de lo que ofrece el proveedor.
Para ser un poco más específico, ejemplificaré en este documento a "un bien de intercambio", como el de la compra de un componente electrónico a una empresa que los anuncia y vende por Internet; mientras que un "servicio" puede ser el de tintorería a prendas con una empresa que similarmente lo ofrezca por Internet. En ambos casos los objetos de intercambio se desean transferir por medio de los anaqueles, MIBoS propósito de ésta investigación.

Cualquiera que sea la solución que se seleccione para estudiar en detalle, ésta debe permitir:
1. contar con una base de datos que incluya: proveedores o prestadores de servicio, usuarios o clientes, casilleros de anaquel o anaqueles, esquemas de operación,
2. estado de la evolución de la entrega-recepción,
3. interfaces con proveedores y/o usuarios, así como para técnicos de mantenimiento y/o administradores o gerentes,
3. validaciones y niveles de seguridad para los diferentes usuarios,
4. protecciones físicas y resguardo de equipos y bienes de intercambio,
5. Procedimientos de instalación y mantenimiento

El alcance de la tesis es realizar un diseño conceptual completo y verificar solo aquellos elementos críticos para la funcionalidad del diseño. No se pretende construir un prototipo completo, ni probar

cada parte para verificar o cuantificar costos precisos de construcción y/o mantenimiento de los anaqueles y servicios de red o conexión. Sin embargo, los componentes construidos y probados deben permitir un primer estimado de esos costos aun cuando los márgenes de error puedan ser de hasta el 100 %.

1.4 Resumen del contenido de la tesis

En éste primer capítulo hemos descrito el problema de transferencia de bienes o servicios entre personas o empresas, de manera que el proceso pueda hacerse asíncronamente como con ocurre con el uso y atención de los cajeros bancarios. El propósito de la investigación a realizar, así como el alcance del proyecto y algunas consideraciones básicas del anaquel a diseñar, son brevemente establecidos.

En el capítulo 2, se hace una descripción más detallada de los requerimientos esperados de proveedores y clientes del anaquel. La base de datos y sus capacidades o alcances son descritas en mayor detalle. Las interfaces físicas para los usuarios, proveedores, son analizadas y comparadas. El diseño conceptual de las partes del sistema se completa con diagramas de flujo y bloques técnicos de las componentes.

En el capítulo 3, se listan y describen en detalle los partes sensibles del diseño y como construirlas y probarlas. Los resultados de construcción y pruebas de esas partes es documentada en el capítulo 4.

Finalmente en el capítulo 5 se analizan posibles modificaciones al sistema, algunas consideraciones para elaborar procedimientos de mantenimiento, así como las conclusiones de ésta investigación.

La bibliografía y los apéndices de éste trabajo se adjuntan al final del documento.

Capítulo 2

Diseño conceptual:

El diseño estándar de las MIBoS consistirá en 9 casilleros, estos serán controlados mediante una tablet, y cada casillero tendrá su cerradura independiente. Las cerraduras se podrán abrir mediante un sensor biométrico, o mediante una aplicación móvil, la cual podrá ser ejecutada desde la tablet del MIBoS o desde los teléfonos inteligentes de los clientes. En cada casillero se podrán almacenar distintos productos, dependiendo de las necesidades de cada local. La cantidad de casilleros y el tamaño de los casilleros que tenga cada MIBoS, podrán variar o tendrá la opción de adecuarse, dependiendo de las necesidades del cliente, usuarios o dueños.

Los dueños de locales donde se instalen los MIBoS, podrán usar también las MIBoS, para entregar sus productos, o incluso recibir y transferir productos de terceros. Ellos también decidirán que usuarios podrán hacer uso de los casilleros (en un caso normal, se le asignaría el "privilegio" de usar los casilleros a los clientes más confiables), ya que lo que se transfiera podría ser peligroso si hay usuarios mal intencionados. Ésto es, los usuarios de los casilleros deberán ser identificables plenamente, así como los productos depositados.

Los clientes o usuarios que usen los casilleros, deberán de ser registrados por el dueño o por los dependientes del local, después de su registro podrán acceder a cualquiera de los casilleros mediante una aplicación móvil o empleando su huella dactilar.

Cada casillero tendrá un registro histórico de las aperturas y cierres realizados. Estos registros, normalmente, podrán ser visualizados únicamente por el dueño o encargado del local.

Los dependientes podrán hacer entrega de los productos a los usuarios empleando estos casilleros. De igual manera los dependientes podrán recibir productos que sean depositados en los casilleros.

La manera de entrega y recepción podrá variar dependiendo de las necesidades del local. Pero el proceso que deben hacer los usuarios son los siguientes:

1.- El usuario es registrado empleando la aplicación móvil y registrando su huella dactilar en el sensor biométrico.

2.- El usuario podrá abrir un solo casillero empleando la aplicación móvil o el sensor biométrico.

3.- El usuario podrá retirar el producto del casillero (en el caso de que un dependiente haya colocado el producto con anterioridad) o podrá depositar un producto dentro del casillero.

Mientras que los pasos que deberán realizar los dependientes son los siguientes:

1.- El dependiente es registrado empleando la aplicación móvil.

2.- El dependiente podrá abrir los casilleros empleando la aplicación móvil.

3.- El dependiente podrá retirar el producto del casillero (en el caso de que un usuario haya colocado el producto con anterioridad) o podrá depositar un producto dentro del casillero.

2.1 Diseño Funcional de las MIBoS

Para aclarar el funcionamiento de las MIBoS, se describirán dos posibles procedimientos en los cuales pueden ser usadas.
El primer ejemplo se basará en el caso de un negocio independiente, el cual requiere automatizar ciertos procedimientos en cuanto a la entrega y recepción de productos se refiere, y en este caso específico, el negocio será una tintorería.
El segundo ejemplo sintetizará la entrega de una venta de un artículo electrónico a través de internet.

2.1.2 Ejemplo de procedimiento de una tintorería empleando MIBoS

En este ejemplo se toma en cuenta un negocio independiente cuyo trabajo principal es el de lavar ropa, pero otros servicios de mantenimiento de prendas, como planchado y teñido también son considerados.
El procedimiento normal de una tintorería consiste en recibir la ropa de un usuario o cliente en un local. Esta ropa es revisada por el dependiente, para después será procesada si convienen en alcance y costo del servicio, y por último será entregada al usuario.
La problemática con este procedimiento es afectada por las dos variables humanas, el dependiente y el usuario.
Si un dependiente llegase a tener un mal día, puede atender de manera poco agradable al cliente, haciendo que el cliente cambie de tintorería debido al trato que recibió por parte del dependiente.
Por otro lado los horarios son un tema muy importante para los usuarios, ya que si su horario es muy limitado, o tiene alguna emergencia, puede que ni si quiera tenga la oportunidad de dejar su ropa en la lavandería o recogerla, ya que las lavanderías tienen también un horario. Este horario se debe a que las lavanderías tienen empleados que no pueden trabajar las 24 horas al día, y en caso de que la lavandería fuese 24 horas, sería un problema para los dueños pagar horas extra y horarios nocturnos a trabajadores.

Justo para resolver estos problemas es que se plantean las MIBoS, porque las MIBoS no tienen un horario de trabajo, y si algún usuario desea ir a dejar su ropa en un horario nocturno, puede ir a depositarlas directamente en un casillero y recibir una notificación cuando su ropa pase a ser procesada. Todo esto reduce la posibilidad de que el usuario reciba una mala atención por un dependiente con mala actitud.

El procedimiento que realizarán las MIBoS es el siguiente:

Procedimiento para recepción-entrega de prendas de tintorería por medio de casilleros de anaqueles (c-a) asíncronos.

1. Alta en base de datos central (bdc), la cual solo puede ser aprobada por gerente autorizado, de:
    - 1.1 Casilleros, inicio operacional y de registro de estados
    - 1.2 Clientes, INE y número de celular
    - 1.3 Proveedores, rfc y celular de responsable
2. Solicitud de uso de casillero-anaquel (c-a) por cliente de tintorería:
    - 2.1 En local por medio de tableta de c-a, o bien

2.2 Vía celular o página web

3. Asignación de casillero-anaquel y un código de apertura a cliente

3.1 En local

3.2 Vía celular o página web

3.3 La asignación tiene validez de media hora

1. Depósito de prendas en c-a asignado

4.1 Llenado de orden de lavado mediante:

4.1.1 Nota adjunta,

4.1.2 Tablet o pantalla local,

4.1.3 Página web, o por

4.1.4 Vía telefónica

4.2 El cerrado de c-a exige el siguiente paso

1. Solicitud de servicio de tintorería se transmite desde c-a

5.1 Se cierra el c-a asignado

5.2 Al concluir el cierre de c-a el código anterior pierde validez

2. Negociación de alcance del servicio entre dependiente y cliente

6.1 Retiro de Prendas por Dependiente de tintorería

6.1.1 Aclaración de condiciones de prendas/servicio/costo

6.1.2 Aceptación por parte del cliente sobre el alcance del servicio

6. 1.3 De no aceptarse el servicio se pasará el paso 11

3. Acordado el servicio se genera nueva clave de apertura para el dependiente

7.1 Con la clave nueva el dependiente abre el c-a

7.1.1 Procede a retirar las prendas

7.1.2 El dependiente adquiere responsabilidad de las prendas al retirarlas

7.2 Se cierra el c-a

7.3 El casillero se libera permitiendo una re-asignación a otro usuario

7.4 Se trasladan las prendas al centro de servicio

4. Solicitud de uso de otro c-a por dependiente de tintorería

8.2 Se solicita localmente un casillero para regresar prendas listas

8.3 Se asigna c-a al dependiente de manera local

8.3.1 Dependiente ingresa las prendas después del servicio

8.3.2 El dependiente cierra el c-a

8.4 Actualiza BD

1. Verificación de pago empleando:

9.1 Efectivo, de manera local

9.1 Transferencia bancaria

9.2 Otros medios

10. El cliente recibe clave e información de c-a para retiro de sus prendas

10.1 El cliente retira sus prendas de c-a

10.2 Se libera el c-a

10.3 El cliente emite nivel de satisfacción del servicio

11. Finaliza procedimiento

Como puede verse en un casillero de cualquier anaquel puede pasar por los siguientes estados:

e1. libre y cerrado (pasos 1 y 2 del procedimiento anterior)
e2. asignado y en proceso de llenado (pasos 3 y 4)
e3. asignado y cerrado (pasos 5 y 6)
e4. asignado y en proceso de retiro de prendas (7), al finalizar este paso el casillero va al e1.
e5. libre y cerrado (pasos 1 y 2 del procedimiento anterior)
e6. asignado y en proceso de llenado (Paso 8)
e7. asignado y cerrado (pasos 8, 9 y 10)
e8. asignado y en proceso de retiro de prendas (paso 11), al finalizar este paso el casillero va al e1.

Como podemos ver de ésta secuencia realmente el c-a solo tiene cuatro estados:

e1, e2, e3 y e4

Puesto que los estados e5, e6, e7 y e8 son nuevamente los primeros cuatro. Éste es un resultado útil desde el punto de vista del c-a puesto que el hardware debe responder a las indicaciones que se envíen desde la bdc. Sin embargo, la bdc si debe llevar un registro histórico de las operaciones que están realizando clientes y proveedores con la finalidad de resolver posibles conflictos o variaciones al procedimiento.

2.1.2 Ejemplo de procedimiento de entrega de un artículo comprado por internet

En este ejemplo se tomará en cuanta una persona o empresa que vende artículos electrónicos usando páginas que hay en internet. Sus ventas funcionan muy bien para personas que viven en otros estados y que compran varios artículos que hasta cierto punto pueden considerarse caros. Pero cuando sus clientes desean comprar pocos artículos o artículos con precios bajos en su mismo estado, resulta problemático para el vendedor concluir la venta, ya que al precio final agrega el costo de envío. Por ello muchos de sus clientes no realizan la compra, porque el costo agregado final del producto incluyendo el envío ya no resulta viable para el comprador.
Y para el caso en el cual el vendedor y el comprador son de la misma ciudad o estado, si los horarios de ambas partes no coinciden, la compra no será efectuada.
Para resolver estos problemas de entregas personales y algunos posibles fallos que se puedan dar con el envío, se pueden instalas MIBoS en diferentes partes de las ciudades, para que el vendedor coloque sus productos en los casilleros, y el comprador vaya a recogerlos cuando su horario se lo permita.

Tomando este ejemplo en cuenta el procedimiento que ejecutara las MIBoS será el siguiente:

Procedimiento para recepción-entrega de componente electrónico por medio de casilleros de anaqueles (c-a) asíncronos.

1. Alta en base de datos central (bdc), la cual solo puede ser aprobada por gerente autorizado, de:
   1.1 Casilleros, inicio operacional y de registro de estados
   1.2 Clientes, INE y número de celular
   1.3 Proveedores, rfc y celular de responsable

2. El proveedor de componentes electrónicos realizará una solicitud de uso del casillero-anaquel (c-a) cuando el cliente haya pagado (en efectivo, localmente, por transferencia bancaria u otros medios) por los componentes:
    - 2.1 En local por medio de tableta de c-a
    - 2.2 Vía celular o página web
3. Asignación de casillero-anaquel y un código de apertura a proveedor
    - 3.1 En local
    - 3.2 Vía celular o página web
        - 3.2.1 La asignación tiene validez de media hora
4. Depósito de objeto electrónico en c-a asignado
    - 4.1 Llenado de orden recibida del cliente mediante:
        - 4.1.1 Nota adjunta,
        - 4.1.2 Tablet o pantalla local,
        - 4.1.3 Por página web, o por
        - 4.1.4 Vía telefónica
    - 4.2 Se cierra el c-a
5. Se envía clave e información de c-a al comprador
    - 5.1 El cliente retira su producto electrónico
    - 5.2 Se libera el c-a
    - 5.2 El cliente emite nivel de satisfacción del servicio
6. Finaliza procedimiento

Como puede verse en un casillero de cualquier anaquel puede pasar por los siguientes estados:

e1. libre y cerrado (pasos 1 y 2 del procedimiento anterior)
e2. asignado y en proceso de llenado (pasos 3 y 4)
e3. asignado y cerrado (paso 4 y 5)
e4. asignado y en proceso de retiro de prendas (paso 5), al finalizar este paso el casillero va al e1.

2.2 Sistemas similares disponibles en México

Actualmente existen muchas ME (máquinas expendedoras) en el mercado nacional, las cuales se enfocan prácticamente en la venta de botanas y refrescos. En cambio las ME de Japón, que como se mencionó anteriormente, manejan una gama de productos mucho mayor que incluso pueden vender comida rápida y otros productos de uso personal.

El problema con estos sistemas es que solo se pueden vender productos específicos limitando la variedad de los mismos productos. A diferencia de esto el sistema propuesto (MIBoS) no solo tiene como meta vender los productos mostrados, también permitirá que el usuario escoja el producto que se

venderá y la persona que lo recogerá o comprará, generando un sistema más complejo y de mayor alcance que las ME.
Las cerraduras y la manera en la que se controla las mismas, es un punto clave en el diseño del sistema propuesto y es otro beneficio por sobre las ME. Estas cerraduras permitirán a las personas un control sobre las MIBoS, y proporcionarán diferentes tipos de acceso a los usuarios, dependientes o dueños.

Así que la comparación con sistemas similares se dividirá en dos partes:

a) Los diferentes tipos de ME
b) Las diferentes cerraduras inteligentes

2.2.1 Listado de máquinas expendedoras similares

Las máquinas expendedoras de nuestro país, casi en su totalidad, solo se encargan de la venta de dos tipos de productos: bebidas y comida chatarra. Limitando de manera sustancial, todas las capacidades que puede ofrecer una máquina expendedora.
La mejor manera de obtener el máximo provecho de estas máquinas, es ofreciendo una mayor cantidad de productos, que reemplazan tiendas o negocios completos. En algunos países, en especial Japón, las máquinas expendedoras no solo venden golosinas y bebidas, también ofrecen gran variedad de productos, como comida rápida, sopas instantáneas, y prácticamente todos los artículos que puedan sustituir algún local.
De esta manera vemos como existe un gran nicho de oportunidad con alguna máquina que sustituya la entrega o venta de productos.

Por otro lado, existen algunas máquinas en desarrollo o trabajos relacionados a ellas en Latinoamérica que pretenden mejorar las máquinas expendedoras, para cambiar o mejorar la manera en la cual se entregan algunos productos.
A continuación se mencionan algunos trabajos sobre estas nuevas máquinas.

Dispensadora portable de medicamentos para personas con enfermedades crónicas.

Esta máquina dispensadora portable de medicamentos, de bajo costo, permite la programación de los horarios mediante una aplicación móvil comunicándose por medio de tecnología Bluetooth con un prototipo dispensador de medicamentos el cual puede albergar tres tipos de medicamentos, para las enfermedades crónicas más comunes identificados según el Instituto Nacional de Salud en Colombia. El dispositivo cuenta con un sistema de alarmas lumínicas, sonoras y vibratorias las cuales se encargan de informar al usuario del momento de ingerir el medicamento respectivo de acuerdo a la programación realizada en la aplicación.

Diseño de una máquina expendedora (vending) para suministrar productos de acceso general y otros de acceso restringido por medio de identificación digital.

Este diseño es de una máquina expendedora que pueda suministrar tanto productos de acceso general como otros de acceso restringido (tabaco, etc.) servirá tanto para facilitar y reducir trabajo en muchos establecimientos como para aumentar las ventas de las máquinas expendedoras en España, debido a que la diferenciación del público en el acceso a los productos restringidos será por medio de una identificación digital, como es el DNI electrónico implantado a todos los españoles en los últimos años. Esta máquina es para una empresa expendedora de 24 horas situada en la calle sin ningún tipo de supervisión la cual podrá vender en su instalación habitual tabaco y bebidas alcohólicas.

Será viable colocar una maquina con este tipo de sistema de identificación en zonas hospitalarias, ambulatorios y farmacias de guardia, para poder proporcionar el servicio de medicamentos comunes, consumidos cotidianamente a personas mayores de edad.
Este servicio reduciría y agilizaría trabajo en las farmacias, y sobre todo mejoraría el servicio de farmacia de guardia.
Esta máquina también podrá ser situada en áreas de servicio, fábricas u oficinas para la venta de tabaco y alcohol sin necesidad de un supervisor.

Diseño y construcción de un prototipo de máquina vending inversa para la aceptación, compactación y almacenamiento de botellas de PET

El diseño y construcción de esta máquina de vending tiene el objetivo de realizar modificaciones conceptuales a partir de especificaciones y características de modelos similares construidos en la entidad auspiciante buscando obtener mejoras en el sistema de identificación y compactación en los cuales se detectaron condiciones no adecuadas de funcionamiento al validar y almacenar botellas vacías de plástico PET de volúmenes comprendidos entre los 250 a 3000cm3

Empleando un proceso de diseño mecatrónico se identifican las posibles mejoras al sistema de identificación y transporte, implementando un software de prueba dedicado a la inspección de objetos a través de la aplicación de visión artificial y un controlador de posición, y las mejoras respectivas para el sistema de compactación mediante la aplicación del diseño y construcción de un sistema neumático. Por otra parte se implementa una interfaz visual interactiva de fácil modificación la cual muestra las diferentes etapas del proceso de manera agradable y de fácil comprensión para el usuario el cual recibe un ticket impreso a manera de incentivo económico por el reciclaje de sus envases vacíos cumpliendo así con el funcionamiento y características de una máquina inversa dedicada a la actividad del reciclaje adecuado de botellas PET.

2.2.2 Listado de cerraduras inteligentes

Las cerraduras comienzan a desarrollarse cada día más, debido a la inseguridad latente que existe en nuestro país. Por otro lado este desarrollo de seguridad, también depende de los tipos de tecnologías que se empleen, modificando la manera en la cual las cerraduras tendrán la opción de abrirse o cerrarse según el usuario lo desee.
Estas tecnologías tienen varios puntos fuertes, ya que como se menciona, lo primordial es la seguridad y los registros que estas cerraduras puedan generar.
Este apartado está enfocado a las tecnologías que emplean las cerraduras, con el fin de encontrar la viabilidad de desarrollar una nueva cerradura o emplear un diseño existente, que nos ayude a almacenar diferente tipo de información respecto a las personas que abren o cierran las cerraduras.

August Smart Lock
Es un seguro inteligente, que funciona únicamente mediante el uso de una aplicación. Esta aplicación permite al usuario decidir si abrir o cerrar el seguro de una manera muy intuitiva, también nos brinda la opción de decidir cuánto tiempo durará el seguro. Otro beneficio de este seguro es que te permitirá activar o desactivar el seguro desde cualquier parte del mundo gracias a que la aplicación se conecta mediante Wi-Fi. Otro beneficio es que permitirá que otras personas accedan según un horario específico, manteniendo un orden de las personas que salen y que entran, y para que a otras personas se les permita el acceso tendrán que descargar la aplicación y conectarse desde su celular. Por otro lado y como medida de seguridad, la cerradura tiene la opción de abrirse o cerrarse de manera manual.

La instalación de este seguro inteligente, es empleando una cerradura existente, o comprando una cerradura redonda, esto podría ser considerado una desventaja para muchas personas, ya que implica el gasto de la cerradura redonda y el gasto del seguro inteligente, eso sin contar que podría causar muchos problemas si el seguro no coincide con la cerradura existente o con la cerradura que se tuvo que adquirir.
Y por último, el seguro funciona empleando dos baterías doble A, sin necesidad de cableado externo. O sin posibilidad de conectarlo a otra fuente de alimentación.

Samsung SHS 1321 Digital lock

Esta cerradura emplea dos partes, la parte interna y la parte externa, la parte externa incluye un teclado digital para poder ingresar el código de acceso, además cuenta con un lector de imán, es decir que esta cerradura se abre mediante una tarjeta, algo que es bastante común en los hoteles, modernos.
La instalación de esta cerradura es muy difícil y poco intuitiva, además de que el espacio que emplea para su colocación es considerable, debido a que emplea dos módulos que deben estar conectados de manera simétrica y esto ocupa espacio tanto interno como externo. La parte interna contiene el seguro y fuente de poder que necesita cuatro baterías doble A, y la parte externa que tiene una pantalla y un lector de tarjeta magnéticos.
Y la única manera de abrir esta cerradura es mediante el código que se ingresa en el teclado o usando una tarjeta magnética.

Kevo kwikset

Esta es una cerradura bastante completa, ya que cuenta con dos partes, la interna y la externa, las cuales se comunican, para saber si el usuario se encuentra dentro o fuera de la casa. La cerradura se puede abrir de 3 maneras. Se puede abrir mediante bluetooth ya sea conectándose al celular o con un token que viene incluido en la compra de la cerradura, y también se puede abrir de manera mecánica con una llave que abre el seguro como las cerraduras comunes. La cerradura se alimenta empleando cuatro baterías doble A, y la aplicación permite el acceso y registro de diferentes personas empleando la aplicación de la compañía.

Ultralock U-tech

La cerradura ultralock es una de las cerraduras más parecidas al modelo que se busca realizar, ya que se puede abrir mediante el celular por conexión bluetooth, con un teclado digital y mediante huella digital.
El lector de huella que emplea este sistema es muy rápido al momento de registrar una huella, y al momento de leer la huella, y puede registrar hasta 95 huellas dactilares.
El diseño es bastante sencillo y como otros modelos cuentan con dos partes la frontal y la trasera. La parte frontal incluye el lector de huellas, el teclado digital, y la palanca para abrir la puerta.
Y la parte trasera cuenta con la palanca para abrir la puerta y con la fuente de alimentación
La instalación es bastante sencilla y se alimenta con 3 baterías doble A.
La aplicación para usar la cerradura permite el acceso, el registro de más usuarios y tener los datos de las personas que acceden empleando su celular.

2.1.2 Tabla comparativa de cerraduras

A continuación se emplea una tabla que emplea ciertos valores numéricos para calificar las funciones de los sistemas, que nos servirá para evaluar la viabilidad del desarrollo de un sistema nuevo.
Todas las funciones son calificadas en este apartado, con una puntuación de 0 a 10, donde 0 es cuando la cerradura no cuenta con la función y 10 cuando la función esta optimizada y cumple con los aspectos que se mencionan en su descripción.

Descripción de las funciones:
-Sensor biometrico: La cerradura emplea un sensor biometrico para abrirse, se toma en cuenta la velocidad con la que detecta la huella dactilar, la facilidad con la que se registran las huellas y la cantidad de huellas digitales que almacena.
-Teclado Físico: Emplea una teclado físico, que se encuentra activo todo el tiempo. Se toma en cuenta las funciones que tiene y las acciones a las cuales podrías acceder empleándolo, como abrir la cerradura con un usuario y contraseña, registrar y eliminar usuarios, crear contraseñas y cambiarlas.
-Tarjeta Magnética: La tarjeta abre la cerradura al momento de acercarla.
-Conexión Bluetooth: La conexión a la cerradura es mediante la tecnología bluetooth, esta conexión permitirá la conexión de celulares, tablets o tokens, que permitan la apertura de la cerradura.
-Conexión a Internet: La cerradura podría abrirse y registrar distintos movimientos como registro y eliminación de usuarios, registro de usuarios que abren la cerradura, cambio de contraseñas y cambio de las mismas.
-Aplicación Móvil: La aplicación funcionara para tener una interfaz amigable entre la cerradura y cualquier dispositivo móvil, la conexión de la aplicación móvil, puede ser mediante bluetooth o mediante internet.
-Partes: Las partes son resistentes y eficientes. No se dañan con facilidad, son seguras, intenta evitar la filtración de agua.
-Servicio: El mantenimiento de la cerradura es sencillo y no requiere de terceros, esto implica que no sea difícil brindarle mantenimiento a la cerradura. Esta parte también incluye en servicio con un precio accesible y con una garantía contra daños de fábrica.
-Partes Modulares: Al momento de tener que reemplazar alguna parte es posible dividir la cerradura en módulos para evitar el desperdicio de las demás partes o en último caso evitar el desperdicio de la cerradura completa. Otro beneficio de la modularidad es que el sistema puede funcionar incluso sin algunas partes, así al remplazar algún modulo, no se detiene para completar la funcionalidad de la cerradura.
-Instalación: Las partes que serán instaladas son adaptables a los posibles lugares donde se vayan a colocar, intentado evitar ser estorbosas o que una o varias piezas de la cerradura no puedan colocarse de manera armónica.
-Costo al Usuario: En esta parte se evalúa el precio aproximado de fabricación contra el precio al cual sale a la venta la cerradura. El precio para la venta es aún más importante que el precio de fabricación ya que de eso dependerá la venta y posible éxito del producto. Y se tomará en cuenta el costo final dependiendo de la marca, y la capacidad de los usuarios finales para adquirir la cerradura.

Funciones	**August Smart Lock**	**Samsung SHS 1321 Digital lock**	**Kevo kwikset**	**Ultralock U-tech**	Propio	Aspectos a evaluar
Sensor biométrico	0	0	0	10	8	Se tomará a cuenta la velocidad con la cual detecta y guarda las huellas digitales.
Teclado Físico	0	8 (Teclado poco intuitivo)	0	10	0	El teclado permite hacer las funciones de registrar usuario, de registrar.
Tarjeta Magnética	0	10	0	0	0	Permite abrir la cerradura con el empleo de una tarjeta magnética.
Conexión Bluetooth	10	0	10	10	10	La conexión es sencilla y automática empleando la tecnología bluetooth, y sirve para abrir la cerradura.
Conexión a internet	10	0	10	10	10	La cerradura emplea conexión a internet para poder abrirse empleando alguna aplicación web.
Apertura Mecánica	10	0	10	10	10	Como seguro extra, o en caso de alguna emergencia se podrá abrir el seguro de manera mecánica sin intervenir en apertura electrónica.
Aplicación Móvil	10	0	10	10	10	Se podrá usar la aplicación para abrir la cerradura, administrar usuarios y generar una base de datos general.
Token	0	0	10	0	0	El token podrá abrir el seguro de la cerradura de manera automática al ser detectado por la conexión bluetooth
Fuente de Alimentación	8	8 (Solo cuenta con un tipo de fuente de alimentación)	8	8	10	Tiene diversas maneras de alimentar el dispositivo, tales como conexión directa a la toma de corriente, por el uso de baterías, mediante carga solar, etc.
Partes	10	10	10	10	10	Se medirá la resistencia, seguridad y adaptabilidad de las partes.

Servicio	8 (El servicio no es directamente con el proveedor)	8 (El servicio no es directamente con el proveedor)	8 (El servicio no es directamente con el proveedor)	8 (El servicio no es directamente con el proveedor)	10	Se toma en cuenta la facilidad y rapidez del servicio que brinda la marca.
Partes Modulares	5 (No puede funcionar sin alguna parte de la cerradura)	5 (No puede funcionar sin alguna parte de la cerradura)	5 (No puede funcionar sin alguna parte de la cerradura)	5 (No puede funcionar sin alguna parte de la cerradura)	10	La cerradura puede reemplazar las partes sin comprometer la cerradura completa y puede funcionar sin algunas partes.
Instalación	5 (Necesita una cerradura previa)	5 (No es compacta y su instalación es complicada)	8 (La instalación puede resultar problemática)	8 (La instalación puede resultar problemática)	10	La instalación es sencilla y la cerradura es compacta o adaptable.
Costo al usuario	8 (Costo poco accesible para el público general)	7 (el costo es excesivo para las funciones)	8 (Costo poco accesible para el público general)	8 (Costo poco accesible para el público general)	10	Se compara el precio entre los dispositivos incluyendo las funciones, y la accesibilidad del usuario para adquirirlo
Ligas/referencias						

2.3 Conclusión de viabilidad de desarrollo comparada con otros sistemas

El diseño propuesto tiene varias ventajas en el mercado actual, gracias a que tiene varias ventajas respecto a las ME y a las cerraduras inteligentes actuales.

La primer ventaja de las MIBoS contra las ME se basa en la capacidad de almacenar aún más tipos de productos. Y estos productos dependerán enteramente del local en el cual serán instaladas las MIBoS, y no de la persona que de mantenimiento y venta de las mismas.

Por otro lado, las MIBoS no solo se limitarán a entregar productos, también podrán ser usadas por diferentes usuarios, aumentando aún más la cantidad y el tráfico de productos que se podrán almacenar en los casilleros.

Las ventajas que tendrá el diseño de una cerradura para las MIBoS serán las siguientes:

La primera ventaja es la multifuncionalidad que puede tener la cerradura que se está diseñando, ya que como se mencionó al principio de este capítulo, la función principal de nuestro diseño es construir un anaquel, y este anaquel tendrá varios lockers, por lo tanto tendrá el mismo número de cerraduras que el número de lockers o casilleros. Esto es importante porque todo estará optimizado para controlar un mayor número de cerraduras prácticamente simultáneas, y estás mismas cerraduras podrán ser usadas individualmente, a diferencia de las cerraduras comerciales actuales que solo funcionan de manera doméstica y en nuestro caso específico solo funcionarían para un locker en vez de funcionar para todo nuestro anaquel.

La segunda ventaja que se tiene con respecto a las cerraduras actuales comerciales es la parte modular que se emplea, ya que los dispositivos o cerraduras actuales funcionan prácticamente como si fueran una sola pieza, la única división de piezas que tiene es la parte interna y externa con respecto a la puerta, y que al momento de colocarla la división se pierde casi por completo. El hecho de tener todas las piezas juntas tiene el beneficio de que ocupa hasta cierto punto menor espacio, pero al no ser modular una parte depende de otra por ello si una parte se descompone el tiempo de reemplazo es más tardado y la cerradura queda inutilizable por completo hasta el momento de sustituir la parte dañada. En cambio la cerradura que se está diseñando puede ser dividida en módulos y funcionar correctamente sin otros módulos, generando automáticamente otra ventaja al momento de reemplazar piezas sin necesidad de desinstalar el sistema por completo.

Por último el costo final es mucho mejor al de las demás cerraduras, ya que se intenta emplear hardware y piezas que son fáciles de conseguir, ahorrando muchos gastos extras como el envío, y el servicio para mantenimiento o reparación de estas mismas piezas o partes que se pueden conseguir fácilmente.

Por todo lo anteriores es viable el diseño de una máquina que cumpla con aún más expectativas que las ME actuales. Pero para lograr esto, se necesitará una cerradura inteligente para cada casillero, una cerradura que deberá cumplir con las necesidades que tengan las MIBoS, desde la adaptabilidad a los casilleros, hasta la seguridad que deben brindar.

2.4 Requerimientos de hardware

Para lograr que el diseño de las MIBoS sea el esperado, se necesitaran distintas piezas de hardware, las cuales estarán encargadas, en su mayoría, en la comunicación, validación de la información y finalmente la apertura de los casilleros.

El diseño de las MIBoS contará con distintas maneras de abrir los casilleros. La primer forma de abrir será empleando una cerradura mecánica, la segunda forma, será empleando un sensor biométrico, y por último, el casillero se podrá abrir mediante una aplicación móvil. Para que estas aperturas puedan realizarse, será necesario emplear distintos elementos de hardware, los cuales se describen a continuación, sin mencionar las validaciones que realizará el software.

Cerradura Mecánica

La cerradura mecánica funcionará de la misma manera que funcionan las cerraduras actuales, es decir empleando una llave común y corriente, la cual moverá ciertos mecanismos dentro de la cerradura, permitiendo la liberación del seguro. De esta manera se podrán abrir de manera manual todos los casilleros, empleando una llave individual para cada casillero.

Microcontrolador

El Microcontrolador es un circuito integrado programable, el cual estará encargado de controlar otros elementos de hardware, tales como el sensor biométrico, el módulo Bluetooth y la cerradura eléctrica. Controlará estos elementos de manera simultánea y esperando alguna respuesta de ellos, para poder almacenar distintos tipos de información.

Este Microcontrolador, se encontrará en una placa de desarrollo, la cual permitirá administrar de una manera más sencilla los dispositivos que se van a emplear.

Cerradura eléctrica

La cerradura eléctrica funcionará para abrir los casilleros (una cerradura por cada casillero de las MIBoS), mediante el Microcontrolador empleando diferentes métodos. Esta cerradura estará conectada a una fuente de voltaje para poder activarse, y empleando un relevador que controlará el Microcontrolador, se podrá abrir o cerrar dependiendo del proceso en que se encuentre la entrega o recepción de bienes.

Sensor Biométrico

El usuario podrá abrir las puertas de los casilleros colocando su dedo sobre el sensor, y este de manera automática y con ayuda del microprocesador, abrirá una cerradura eléctrica de algún casillero disponible o el casillero que le corresponda al usuario que colocó su dedo.

Módulo Bluetooth

El usuario se podrá conectar mediante Bluetooth a este módulo, el cual estará conectado al microprocesador y de esta manera podrá abrir la cerradura eléctrica.

Dispositivo Móvil

Para poder conectarse al módulo Bluetooth se necesitara un dispositivo móvil. Este dispositivo móvil contará con alguna aplicación para mandar los datos necesarios al módulo y con esto abrir la cerradura eléctrica.

2.5 Requerimientos de software

Programación arduino

Para programar el microcontrolador que en este caso será empleando una placa de desarrollo (Arduino), se usará la plataforma Arduino IDE. Este programa estará encargado de abrir todos los casilleros de las MIBoS de manera individual, a partir de mandar una señal a los relevadores que estarán conectados a cada una de las cerradura eléctricas de los casilleros. También será el encargado de administrar, almacenar y leer las huellas registradas por el sensor biométrico. Y por último, el programa controlará el módulo Bluetooth, por lo tanto será el encargado de interpretar las señales de entrada y salida registradas por este módulo.

Aplicación Móvil

La aplicación móvil será desarrollada para los dispositivos que cuenten con el sistema operativo Android. Esta aplicación estará encargada de conectar el dispositivo móvil (tableta o smarthphone) con el módulo Bluetooth. Y desde la aplicación se podrá registrar y eliminar usuarios, registrar huellas para el sensor biométrico, agregar dependientes, y administrar todos los casilleros de las MIBoS. También desde la aplicación se podrán ver todos los registros de las bases de datos, para tener un control de las mismas.

Base de datos

La base de datos estará encargada de recibir la información que mande tanto la aplicación móvil como el programa de Arduino. Esta información la tendrá que organizar de tal manera que este disponible en la aplicación móvil.

La base de datos es una de las partes más importantes del proyecto, ya que implica la seguridad de los usuarios y el registro de toda la información de uso de las MIBoS. Esta información puede ser clasificada como información personal, por ello el diseño y la seguridad con la cual se almacene la información será vital para el desarrollo del proyecto.

2.6 Identificación, autenticación y seguridad

Existen varias partes para el desarrollo de las MIBoS, las cuales pueden ser consideradas delicadas. Estas partes son las encargadas de que las MIBoS funcionen de manera segura y así brindar de alguna manera, mayor seguridad a los usuarios.

Identificación y autenticación

Existirán 4 tipos de usuarios que se registrarán en las MIBoS, estos usuarios podrán acceder empleando su huella dactilar en el sensor biométrico o usando su nombre de usuario y clave en la aplicación móvil. En vista de que los MIBoS pueden ser una oportunidad para maleantes o personas mal intencionadas su uso requiere el registro de las operaciones con video de buena resolución de los usuarios y dependientes.

El primer tipo de usuario serán todos los clientes que no sean frecuentes, y los definiremos como invitados. Los invitados podrán hacer uso de cualquier casillero de las MIBoS generando un usuario y una contraseña. Ese usuario y contraseña solo se podrá usar 2 veces. Este tipo de usuario es para las personas que vendan por internet, ya que normalmente solo usarían las MIBoS para colocar algún producto y que la persona que lo haya comprado pase a recogerlo empleando el mismo usuario y contraseña que la persona que dejó el producto. El único registro que se les pedirá será una identificación, que puede ser su correo o su identificación, y lo mismo se le pedirá al invitado que vaya a recoger su producto, para que ambas partes estén seguras.

El segundo tipo de usuario, serán todos los clientes que se registren en la aplicación. Este tipo de usuario está dedicado a los clientes más frecuentes del local en el cual se estén registrando. Con su registro y autenticación podrán hacer uso de las MIBoS, accediendo a los casilleros con su huella dactilar o con su usuario y clave para dejar o recoger sus productos dependiendo del caso. Para que los clientes se puedan registrar deberán proporcionar los siguientes datos:

Nombre completo, dirección, nombre de usuario, contraseña, huella dactilar y correo.

Para corroborar todos sus datos se pedirá una identificación vigente, tanto para su seguridad como para la seguridad del dueño del local y sus instalaciones.

El tercer tipo de usuario, será designado por el dueño del local. Estos usuarios son los dependientes del local, es decir los trabajadores. Para que ellos tengan acceso a las MIBoS, se les pedirán los mismos datos que a los clientes y a los invitados. Con su registro, los dependientes tendrán acceso a las MIBoS, para recoger o introducir los productos a las MIBoS que en algún momento terminarán recogiendo los clientes. También podrán ayudar a nuevos clientes a registrarse para hacer uso de la plataforma.

El último tipo de usuario, se le llamará súper-usuario, y solo podrá haber un súper-usuario por cada MIBoS. Este usuario tendrá acceso a toda la información que proporcionen a las MIBoS. Los usuarios podrán identificarse en las MIBoS, de distintas maneras, ya sea con su huella dactilar, o con su usuario y contraseña. El súper-usuario, será el único que podrá registrar a los dependientes, y al igual que los dependientes podrá registrar a todos los usuarios. También será el único que podrá ver la información de todos los usuarios y administrar las cuentas y los casilleros, para que en caso de que ocurra algún problema, se pueda ver lo que sucedió mediante los registros.

Seguridad

Las MIBoS deberán estar protegidas contra diferentes tipos de problemas, para ello se implementarán medidas de seguridad para el software y el hardware.

Una de las mejores maneras de mantener la seguridad del software, será evitando conectar la plataforma a internet o a la red telefonia. Es decir todo el proceso se hará de manera local, almacenando la información en la aplicación, o en un servidor local que solo tendrá el dispositivo móvil incluido con las MIBoS. Esto evitará el robo de datos personales, tanto de los clientes como de los dependientes.

También y solo como caso de seguridad se buscará que solo el súper-usuario tenga acceso a la información.

Por otro lado, el hardware puede sufrir daños o tener interrupciones. Para arreglar esta problemática también se podrán usar los casilleros de manera mecánica empleando una cerradura común, la cual solo estará disponible para el dueño del local o para el personal autorizado. Esta solución está prevista para los casos en los cuales no se puedan abrir los casilleros por falta de corriente o por errores del hardware, y los productos de las personas no resulten afectados o el procedimiento quede interrumpido.

Como recurso básico para el funcionamiento de las MIBoS, existirá una cámara que monitoreará las MIBoS, guardando la información, para cualquier percance que pueda afectarlas de manera física (que golpeen la maquina o que intenten robarla o forcejearla para abrir los casilleros).

Capítulo 3

Pruebas a componentes críticos/sensibles/importantes del diseño

3.1 COMPONENTES IMPORTANTES

Existen varios componentes que nos pueden ser de utilidad para el desarrollo de las MIBoS. Pero estos mismos componentes pueden traer varios problemas.

En este capítulo se describirán a detalle todos los posibles componentes o herramientas que se pueden usar para que las MIBoS cumplan con los requerimientos. Este análisis se hará con el fin de que las MIBoS puedan ser modulares, escalables y adaptables a los distintos ambientes en los cuales se instalarán, utilizando solo componentes de fácil acceso en el mercado nacional.

Cerradura

Las cerraduras tienen uno de los papeles más importantes en las MIBoS, por que justo a partir de ellas, es por donde empezará la seguridad de todos los casilleros. Para que las cerraduras funcionen de manera correcta con todo el sistema deberán ser operados con dos tipos de cerraduras: una mecánica y una eléctrica.

La cerradura mecánica se encargará de poder abrir los casilleros, aunque el sistema no tenga fuente de alimentación eléctrica, o bien cuando el sistema tenga alguna falla.

La operación de apertura y cierre será controlado por un Arduino con varios métodos de ingreso al sistema de validación y operación del mismo.

Arduino

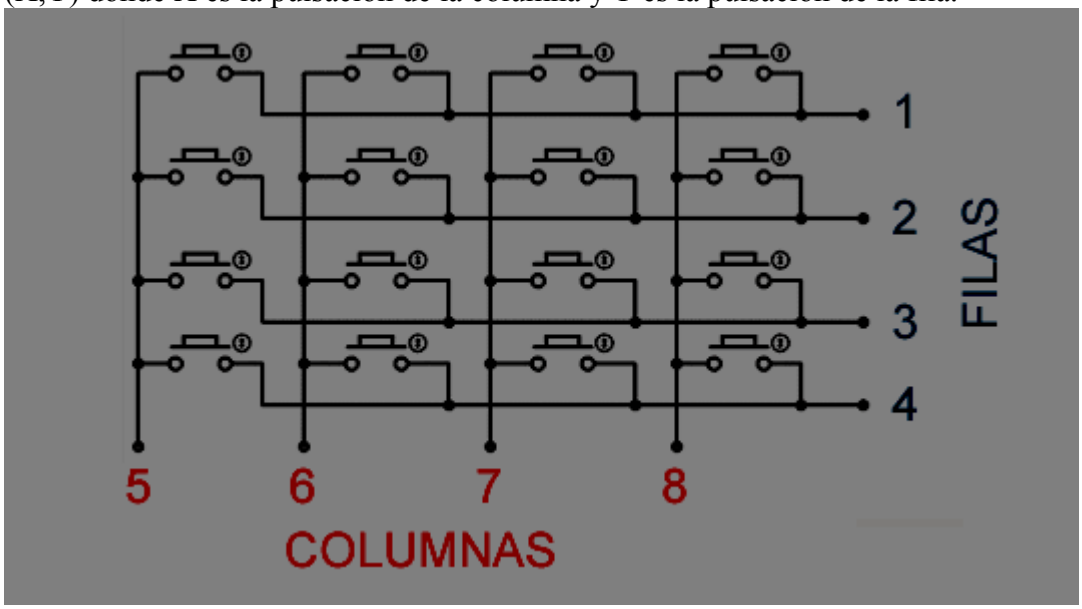
Arduino Mega es una placa del microcontrolador ATmega1280, el cual tiene 54 pines digitales de entrada y salida, de los cuales 14 pines pueden ser usados como salidas analógicas, también tiene un oscilador de 16Mhz, conexión usb, un enchufe de alimentación y un botón de reset.
El control del hardware es una parte muy importante para este proyecto, ya que es el que nos brindará la seguridad de abrir o cerrar los casilleros correctos, manteniendo otros sin apertura.
Crear un cerrojo que se pueda abrir y cerrar a nuestro gusto es una parte muy importante la cual se lograría únicamente con elementos de hardware, elementos que impliquen permitir el paso de corriente a través del cerrojo o que eviten el paso de la misma.
Para realizar lo anterior, se podría poner simplemente un switch, el cual abra y cierre los casilleros, es decir controle el cerrojo, pero por otro lado lo que se busca es que este diseño de casilleros se pueda controlar de manera inalámbrica, y existen muchas maneras de hacerlo, pero por cuestiones de utilidad se usa la placa de desarrollo Arduino Mega que contiene un microcontrolador.
Gracias a lo anterior es bastante útil el empleo de una placa Arduino (no necesariamente el modelo "Arduino Mega"), ya que por sus componentes nos podrá ayudar a controlar el seguro que es alimentado mediante 5 volts, y no sólo puede ser uno, ya que este puede ser empleado en algún caso donde existan más de un casillero, permitiendo abrir o cerrar todos los casilleros.

Otra característica importante es su bajo costo debido a sus volúmenes de producción, esto es de mucha utilidad al momento de realizar pruebas y no gastar demasiado dinero para el desarrollo de las mismas.

Teclado matricial

Emplear un teclado físico nos permitirá controlar el acceso al sistema empleando un menú y después de satisfacer las validaciones implementadas abrirá el casillero empleando un Arduino. Por ello esta es una de las opciones que se deben considerar para desarrollar y verificar su viabilidad.

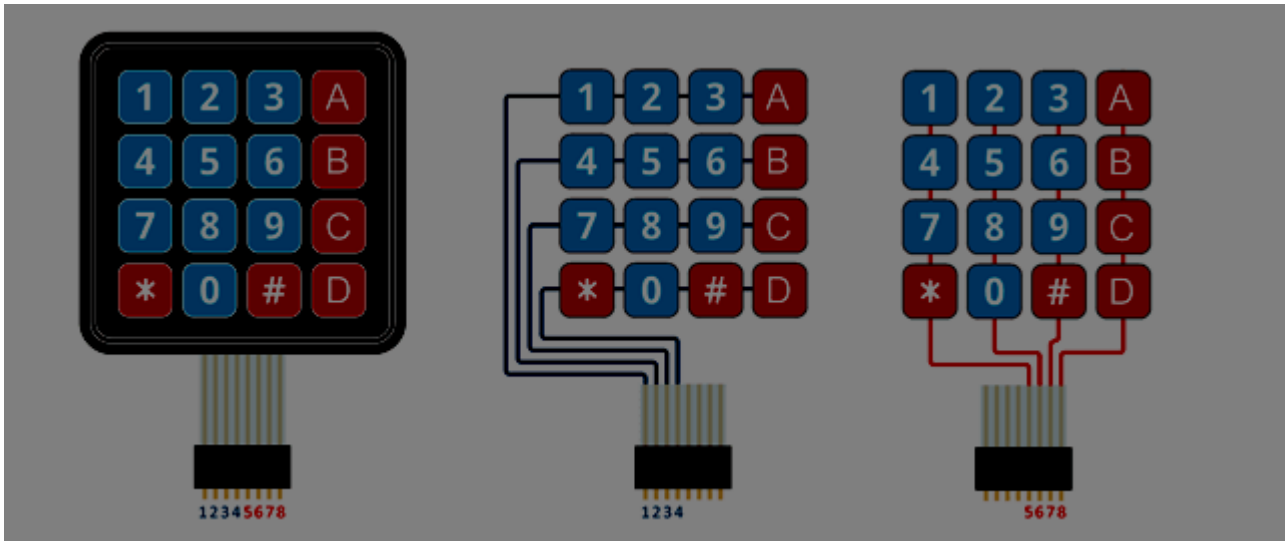
El keypad o teclado matricial es formado mediante un una matriz de pulsadores, en nuestro caso es un arreglo de interruptores de 4x4 que están agrupados mediante filas y columnas.
Funciona mediante la pulsación de alguna coordenada en nuestro teclado, guardándose la forma (X,Y) donde X es la pulsación de la columna y Y es la pulsación de la fila.



Para detectar la pulsación se pone la tierra a un extremo del pulsador y en el otro extremo conectaremos una entrada digital con una resistencia pull-up.

Para leer todas las teclas tendremos que hacer un barrido por filas. En primer lugar ponemos todas las filas a 5V, y definimos todas las columnas como entradas con resistencia de pull-up.
Progresivamente ponemos una fila a 0V, y leemos las entradas de la columna. Una vez realizada la lectura volvemos a ponerla a 5V, pasamos a la siguiente fila, y volvemos a realizar el progreso hasta recorrer todas las filas.
Para detectar NxM pulsadores necesitamos sólo N+M conductores. Por lo tanto, el ahorro de conductores es superior cuanto más grandes sean N y M, y más parecidos entre sí.

Para conectarlo a nuestro Arduino insertamos los pines del teclado a nuestro Arduino y la manera en la cual se registra en el Arduino será imprimiendo los valores que se reciben o detectan del teclado.



Exiten muchas maneras de realizar el código necesario para que el Arduino funcione con el teclado, pero en nuestro caso y para facilitarnos la detección de las pulsaciones empleamos una librería llamada Keypad, la cual nos ayuda a organizar mejor nuestra matriz y en este caso nuestro código. (ver Apéndice 1)

Conectar nuestro teclado matricial y su correcto funcionamiento es una parte muy importante del proyecto, ya que se encargará de la comunicación externa o de manera más correcta tendrá una comunicación directa con el usuario, ya que empleando nuestro teclado matricial escribiremos cual es el usuario y cuál es su contraseña.

Módulo I2C y Display 16x2

El módulo I2C tiene muchas funciones aplicadas mediante el Arduino, pero para utilidad del proyecto únicamente tendrá una función, la cual será disminuir el número de pines utilizados en nuestro Arduino para hacer funcionar nuestro display de 16x2.
El módulo I2C emplea un controlador I2C PCF8574, este es un expansor de I/O digitales y controlado por I2C, este módulo se emplea en su mayoría para controlar LCD alfanuméricos.
El I2C tiene por defecto dos posibles direcciones hexadecimales 0x3F y 0x27. Identificar la dirección de nuestro modulo nos evita problemas con la programación.
Y se conecta a nuestro Arduino empleando únicamente 4 pines: Voltaje, Tierra, SDA SCL
El pin de voltaje es con el cual se alimenta nuestro modulo, será alimentado mediante 5 volts, y con el pin de tierra se cerrará el circuito.

I2C son las siglas que hacen referencia a una norma llamada Inter Integrated Circuit Bus, esta norma es para la comunicación digital entre diferentes componentes de un sistema electrónico, tiene varias especificaciones, pero las principales son:

Es un protocolo controlado por dos hilos, uno hilo es empleado para la trasmisión de datos en este caso llamado SDA, y el otro hilo es un reloj asíncrono que nos avisa cuando leer los datos es decir SCL.
Emplea 7 bits y cada dispositivo que es conectado al bus I2C tendrá su dirección exclusiva.

Quien controla al reloj será un dispositivo y se llamará Master (en nuestro caso quien controla el reloj es el Arduino Mega), y la velocidad del reloj puede variar dependiendo del dispositivo Master.
Puede haber más de un dispositivo Master, pero sólo uno puede estar activo a la vez, y deberá ddar una detección de colisiones y un protocolo de arbitraje.
El bus es activo en un nivel bajo, es decir la señal se activará cuando sea 0.

El pin SDA y el pin SCL se conectan directamente a los pines 20 y 21 de nuestro Arduino Mega. Esta es la parte más importante del empleo de este módulo, ya que en vez de conectar nuestro Arduino al display, y usar 6 pines, usaremos únicamente 2 pines de nuestro Arduino, esto es una ayuda muy notable al momento de construir nuestro hardware, porque esos 4 pines que sobran los podemos usar para otros elementos, e incluso únicamente por limpieza de nuestro circuito.
Para este caso decimos que usamos 2 pines, debido a que no contaremos el pin de alimentación y tierra, ya que esos dos pines serán los que usaremos para alimentar todas nuestras partes externas del Arduino incluidos algunos módulos.
La librería que usamos para poder controlar los pines SDA y SCL de nuestro Arduino Mega se llama Wire, y nos sirve para gestionar el protocolo de comunicación entre nuestro Arduino, nuestro I2C y nuestro display.

El modulo que se empleara junto con el controlador PCF8574 cuenta con un potenciómetro que servirá para regular el contraste de nuestro display, y cuenta con un jumper que activará el LED de nuestro display, esto último también puede ser ajustado mediante código.

El primer paso para comenzar a usar el módulo consiste en encontrar la dirección hexadecimal de nuestro componente conectado al módulo I2C, ya que como se mencionó el módulo I2C detectará algún esclavo en alguna de sus localidades, y está es la localidad que se usará para programar nuestro display. (Ver apéndice 2)

En el caso de nuestro dispositivo nos dice que usa la dirección 0x3Fy es la dirección que usaremos durante todo nuestro proyecto, para poder controlar el display.
Otra parte crucial que nos ayudará a mantener el orden del código y la optimización del mismo es emplear el uso de la librería LiquidCrystalI2C (Ver apéndice 3), que contiene varias instrucciones que nos ayudarán a optimizar nuestro programa. Lo primero que hacemos para probar el funcionamiento de este módulo es imprimir un hola mundo en nuestra pantalla (Ver apéndice 4), de este modo se comprobaría el funcionamiento correcto de nuestro LCD.

Aplicación web

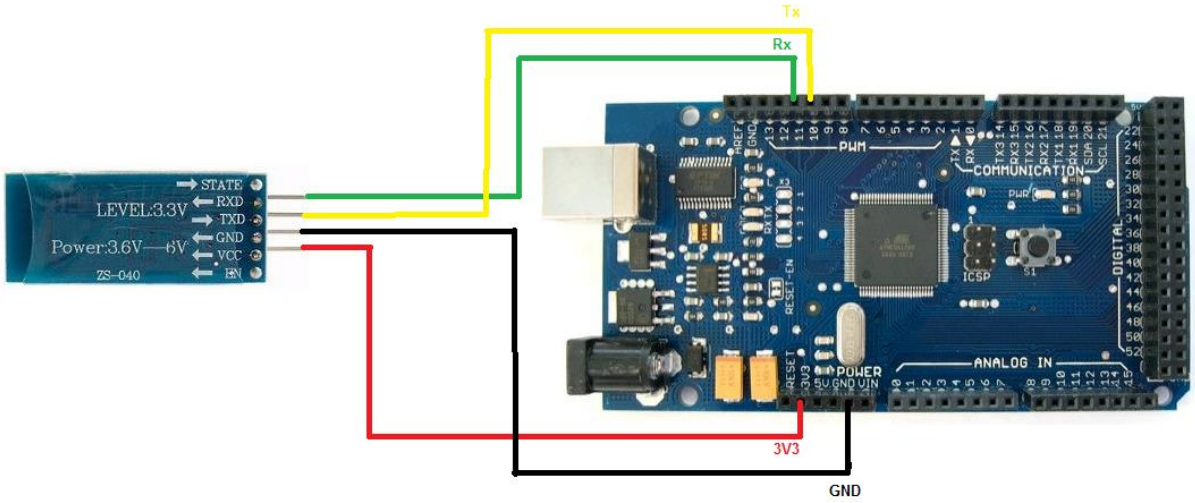
El diseño e implementación del entorno web es una de las partes más importantes para el desarrollo de este proyecto, debido a que se necesita una buena presentación, que sea llamativa y a la vez fácil de usar y con ello atraer más usuarios.
Es uno de los primeros pasos a realizar, sobre todo es importante realizar la aplicación web antes que realizar la aplicación móvil para los sistemas operativos tanto de IOS como para Android. Esto se debe a varias razones, primordialmente evitar descargar una aplicación móvil, es decir, tener la opción de simplemente abrir el navegador de tu celular y no instalar aplicación alguna en el mismo; ya que muchas veces existen complicaciones al realizar la descarga por falta de conocimiento, por falta de espacio en memoria o incluso simplemente porque el cliente no desea agregar más aplicaciones móviles a su dispositivo.
Y está claro que para realizar una aplicación web con estas características (buena presentación, sencilla y llamativa) es necesario usar las herramientas correctas.
El desarrollo de aplicaciones web, involucra varias partes, la primera parte es lo que frecuentemente se llama front e incluye la parte visual de nuestra aplicación web, es decir la parte que verá el usuario final, desde las imágenes videos, estructura y forma, hasta la presentación y el dinamismo de esta. La segunda parte consiste en el background es decir las acciones que realizará nuestra aplicación web pero que el usuario no podrá ver directamente como funcionan, por ejemplo cuando un usuario se registra, al mismo no le interesa que es lo que realiza la base de datos o como es agregado a una tabla.

- **Front**
  Esta parte será realizada principalmente por 3 herramientas, HTML, CSS y JS
  - HTML Es un lenguaje de marcado, el cual es un estándar para la elaboración de páginas web, este estándar es responsabilidad de W3C quien es encargado de estandarizar la mayoría de las tecnologías que están conectadas a la web. Esta herramienta tiene como función principal la de añadir elementos a la página mediante referencias, es decir el archivo HTML solo contiene texto.
  - CSS es un lenguaje de diseño el cual sirve para crear y definir la presentación de un lenguaje marcado (en nuestro caso es HTML). Este lenguaje sirve para darle mejor presentación de algún documento o archivo empleando layouts y modificando los colores y las fuentes. También sirve para mantener un orden de diseño o un estándar de diseño, esto se logra empleando una misma hoja de estilo, de esta manera evita que el documento que contiene el texto sea repetitivo.
  - JavaScript esta es una de las partes más interesantes ya que es un lenguaje de programación orientado a objetos el cual se encarga de entregar dinamismo a la página mejorando la interfaz de usuario.

- **Background**
  Para el desarrollo del background se necesitar usar de un framework que nos conectará con la base de datos y con los servidores web, en este caso, la opción de framework que se podría usar sería Django, y la base de datos será hecha en MySQL.

Modulo Bluetooth HC-05

Una de las mejores maneras para tener comunicación inalámbrica con el Arduino, puede ser empleando un módulo bluetooth, al cual se le podrán enviar distintos datos de manera inalámbrica, para que este módulo a su vez, mande esos mismos datos a Arduino para ser interpretados, y dependiendo de la información mandada, se ejecutará algún tipo de acción.
El módulo que se propone es el modelo HC-05. Uno de los beneficios de este modelo es que puede funcionar como maestro o esclavo, además de que empleando el comando AT se pueden configurar y revisar varias cosas de este módulo, tales como: Nombre, código de acceso, manera de conexión, baudios etc.
Para poder acceder a estos comandos, el HC-05 se conecta a algún arduino de la siguiente manera:



Y ejecutando un simple código en el Arduino (Ver apéndice 5) y empleando la interfaz de Arduino se ejecutan los distintos comandos AT (Ver apéndice 6).

El principal funcionamiento que se le dará al módulo bluetooth será el de recibir datos desde un dispositivo móvil, y que dependiendo del dato que reciba e implementando el Arduino, abrirá o cerrará las distintas cerraduras.

Sensor Biométrico
Una de las mejores maneras de identificar a un usuario de tal manera que nadie pueda suplantarlo, es mediante un sensor biométrico, el cual se encarga de realizar el escaneó de huellas dactilares.
La seguridad que proporciona este tipo de sensor es de las mejores, dado que es prácticamente imposible que dos personas tengan la misma huella digital.
El sensor biométrico que se busca emplear almacena hasta 162 huellas, el cual es un número bastante aceptable para la posible cantidad de usuarios que se pueden tener.
Habrá dos programas distintos para el funcionamiento de este sensor, el primero será encargado de registrar las huellas dactilares (Ver apéndice 7), y el segundo, se encargará de la lectura de las huellas (Ver apéndice 8).

Ambos programas se encuentran guardados en la librería "Adafruit Finger Print Sensor Library" por ello lo único que se tendrá que hacer es cargarlos directamente al Arduino.
El registro de huellas, se basa en capturar distintas imágenes de la huella que se esta colocando sobre el sensor y de esta manera obtener mejores datos que almacenar y con esto tener una mayor seguridad al momento de la lectura de huellas.
El segundo programa, encargado de la lectura de huellas, será el más importante, ya que se adaptará su funcionamiento de tal manera que al comparar la huella que está siendo colocada en el sensor con las diferentes imágenes que están guardadas, y cuando la huella colocada coincida con alguna imagen guardada, confirmará la huella que está siendo colocada en el sensor.
El segundo programa tendrá gran importancia, debido a que será el encargado de verificar la coincidencia de la huella que está siendo colocada en el sensor con las huellas almacenadas. Y si la huella que está siendo leída coincide con alguna almacenada el seguro de los casilleros se abrirá, en caso de que no exista ninguna coincidencia, enviará un mensaje de error.

Cerradura Mecánica hasta aquí llegó la revisión de apéndice y bibliografía

La cerradura mecánica será un seguro común y corriente, el cual estará colocado en todos los casilleros, para que se puedan abrir de manera manual en caso de que ocurra algún inconveniente con el sistema, es decir funcionaran como sistema de emergencia.
La cerradura que se usará será la siguiente:

Esta cerradura es la más adecuada para el diseño de las MIBoS, ya que se adaptara a todo el sistema sin causar una invasión al mismo.

Cerradura Electromecanica

Esta cerradura será activada empleando una fuente externa que alimentara la cerradura para realizar la apertura. Al ser manipulada mediante corriente, solo se deberá de controlar el paso de la misma empleando un relevador.

Aplicación Arduino (Arduino IDE)

Para realizar y subir programas usaremos la plataforma Arduino IDE. El código que se subirá a la placa Arduino, deberá contener dos funciones esenciales, la primera será la función setup y la segunda la función loop.

El funcionamiento principal del programa del Arduino para las MIBoS estará encargado de abrir y cerrar los distintos casilleros colocados en las MIBoS. Pero este programa crecerá conforme se adapte a los distintos módulos y sensores que se anexarán, tales como el sensor biométrico y el módulo bluetooth. De tal modo que empleando la aplicación de Arduino, este será capaz de abrir y cerrar todos los casilleros mediante bluetooth o con la huella dactilar de los usuarios.

Aplicación Móvil

Actualmente existen dos sistemas operativos predominantes en el mercado de los móviles, Android y IOs.
El sistema operativo Android es empleado por muchas marcas de renombre como Samsung, Huawei, Motorola, LG, etc. Incluyendo así gran cantidad de dispositivos móviles. Por otro lado el sistema operativo IOs. Solo puede ser empleado por los dispositivos móviles de la marca Apple. Por ello, Android es la mejor opción para desarrollar la aplicación, ya que abarca mayor número de dispositivos móviles.
Android Studio es el entorno en el cual se desarrollará la aplicación para la plataforma Android.

En términos generales la aplicación móvil servirá para realizar varias funciones. La primer parte de la aplicación consistirá en el registro de los distintos tipos de usuarios, y permitirá el registro de sus huellas digitales de ciertos usuarios al momento de registrarlos. La siguiente parte será la conexión con el módulo bluetooth, es decir la aplicación sincronizará el dispositivo móvil con el Arduino mediante conexión bluetooth. Por último la aplicación tendrá la función de interfaz, y en ella se podrá ver el estatus de todos los casilleros, al igual que la información de todos los usuarios.

Base de Datos

Existen muchos sistemas gestores de bases de datos, pero para este proyecto se usará MySQL, ya que este servidor nos permitirá conectarnos desde una aplicación web o alguna aplicación móvil, permitiendo una escalabilidad aceptable.

La base de datos será la encargada de administrar a todos los usuarios y su información, al igual que la información histórica de los casilleros.

Seguridad

La parte de la seguridad se brindará cifrando la información personal de los usuarios en cuanto se almacenen en las base de datos. De igual manera y para evitar alguna intromisión de terceros, se buscará, en medida de lo posible, que la plataforma de los casilleros no esté conectada a internet.

Por otro lado pueden existir otros problemas de seguridad, como la seguridad física, es decir evitar que los casilleros puedan ser abiertos por personas no autorizadas. Para resolver este problema se usará una cerradura mecánica, la cual aportará seguridad extra en caso de que se intente violar el seguro eléctrico.

Además del registro que se almacenará en la base de datos sobre los usuarios que emplean los casilleros, también se usará una cámara de seguridad, para evitar cualquier mal entendido y poder monitorear el estado general de las MIBoS.

El material con el cual estarán realizadas las MIBoS será lámina de acero al carbón, o lámina galvanizada, por su costo accesible, duración y resistencia, siendo bastante agradable a la vista sin aditamentos extra.

3.2 Conclusión de viabilidad de componentes

Al evaluar los aspectos de todos los elementos propuestos para el sistema. Se escogieron los componentes más viables comparando las funciones parecidas o incluso reemplazables y para reducir la cantidad de elementos de hardware en el sistema, logrando así una mayor accesibilidad al sistema en cuanto a costo, y una mayor adaptabilidad a diferentes ambientes o sitios de colocación para las MIBoS. Para evitar quitar algún elemento del sistema, el elemento deberá de tener un funcionamiento armónico con todo el sistema promoviendo la escalabilidad e intentando usar las tecnologías más modernas.

Por ello los siguientes componentes no serán contemplados para el diseño final del sistema:

Teclado Matricial, módulo I2C, display LCD de 16x2 y aplicación web.

Los tres primeros componentes trabajan juntos para un mismo fin, el cual es desplegar y navegar dentro de un menú. El módulo I2C funciona para disminuir el número de pines que usaría el display LCD de 16x2. Este display a su vez, nos serviría para mostrar un menú empleando el Arduino, y la manera en la cual navegaríamos en este menú sería usando el teclado matricial.

El proceso de estos tres componentes, será sustituido por solo 2 componentes, los cuales serán un módulo bluetooth, y una Tablet o celular. El dispositivo móvil sustituirá la interfaz que se desplegaría en el display de 16x2, y también sustituirá el teclado matricial, todo esto empleando una aplicación móvil. El modulo bluetooth se encargará de la comunicación entre el Arduino y el dispositivo móvil.

El display se encargaría únicamente de mostrar un mensaje, mientras que toda la información y programación se ejecuta o guarda en el Arduino, dando mayor carga de trabajo a la placa. Por otro lado la aplicación móvil tiene más ventajas que solo desplegar un menú, también tiene mayor cantidad de almacenamiento (muy útil para la base de datos), más opciones de diseño y de navegación en la interfaz, por último, como la ejecución de la aplicación es en el dispositivo móvil, quita carga de trabajo al Arduino, de esta manera el trabajo estará más ordenado y el diseño de las MIBoS será más modular.

La aplicación web puede sustituirse parcialmente por la aplicación móvil, pero como de momento el diseño no contempla una conexión a internet, la aplicación web no será contemplada en el diseño final del sistema.

3.3 Detectores indispensables

Gracias a la investigación de varios elementos y a que se descartaron otros elementos, se llegó a la conclusión de que existen elementos que son indispensables para el desarrollo de las MIBoS.

El siguiente listado explica los elementos con los cuales las MIBoS no podrían cumplir su función.

Arduino: Como se ha mencionado, esta placa será la encargada de controlar la apertura de los seguros de cada casillero, y se encargará de interpretar la comunicación entre dispositivos.

Dispositivo Móvil (Tablet o celular): Almacenará la información de los usuarios, y será la parte responsable de autorizar el acceso de los mismos a los casilleros.

Módulo Bluetooth: Encargado de la comunicación entre el dispositivo móvil y el Arduino.

Seguro Mecánico y Electrónico: Encargados de la apertura manual o electrónica de los distintos casilleros, y serán la protección más importantes para los elementos guardados en los casilleros.

Empleando la aplicación en el dispositivo móvil se accederá a una cuenta, para mandar la información de apertura de algún casillero, esta información será recibida por el módulo Bluetooth que a su vez lo transmitirá al Arduino para que active o desactive el relevador del seguro de alguno de los casilleros.

Estos 6 elementos en conjunto con la armadura de las MIBoS, son los elementos básicos para el desarrollo del proyecto.

Capítulo 4

4.1 Diseño formal

El diseño formal se realizó dividiendo el proyecto completo en varias etapas, en las cuales se desarrolló en partes específicas del proyecto y controladas mediante el método kanban, mientras el control de versiones es mediante github.

Kanban

Kanban es un sistema de tarjetas, para controlar el desarrollo de algún producto. Este sistema funciona mediante contenedores, los cuales tienen diferentes nombres, estos nombres sirven para identificar la acción que se está realizando en ese contenedor. Las acciones son representadas mediante tarjetas.
Y los contenedores serán en total 5: Bugs | To do | Doing | Done | Master.
Los contenedores están organizados de izquierda a derecha es decir las tarjetas empezarán desde To Do hasta que lleguen al contenedor llamado Master.
El contenedor llamada To Do, es el primer contenedor. Este contenedor aloja las tarjetas que representan todas las tareas que se tienen que hacer para realizar el proyecto.
El contenedor Doing aloja una sola tarea. Esta tarea es la que se está realizando actualmente, y solo podrá existir una tarjeta en esta sección, la cual al ser concluida pasará al siguiente contenedor.
El contenedor Done aloja todas las tarjetas que ya han sido terminadas que no tienen ningún problema para ejecutarse.
El ultimo contenedor es el contenedor Master, en este contenedor, todas las tarjetas que se encontraban en Done son integradas, es decir, cuando una tarjeta está en Done para poder llevarla a Master deberá de ser integrada al proyecto completo, o a los componentes que estén actualmente en Master y trabajar de manera armónica con este mismo contenedor. Esta rama es la más importante ya que es la rama que se muestra a los usuarios, es decir es una rama publica por lo tanto no debe tener errores.

El contenedor Bugs, es un contenedor cuya prioridad está por sobre todas las tarjetas alojadas en To Do y en Doing, ya que en el ámbito informático, los bugs son un problema que se tiene que resolver en el momento en que surgen. En este contenedor se alojan los bugs que fueron encontrados en la rama Master. Por lo tanto en cuanto se coloque una tarjeta en Bugs, se deberá de colocar en el contenedor Doing dejando las tareas actuales a un lado, y hasta que la tarjeta de Bug sea arreglada, se podrá retomar con la tarjeta en la que actualmente se estaba trabajando en la rama Doing.

Control de versiones en GitLab

GitLab es una plataforma que ayuda a controlar las versiones de un proyecto mediante un sistema de control de versiones de Git. Aunque se utiliza primordialmente para el control de código fuente, también lo podemos emplear para controlar las versiones de diferentes textos.
Esta plataforma me ayuda a mantener un orden en las versiones de código que escribo.

Una de sus cualidades es el hecho de poder actualizar automáticamente los cambios que se hacen al código, y en caso de que hubiese un error regresar a versiones anteriores del código.

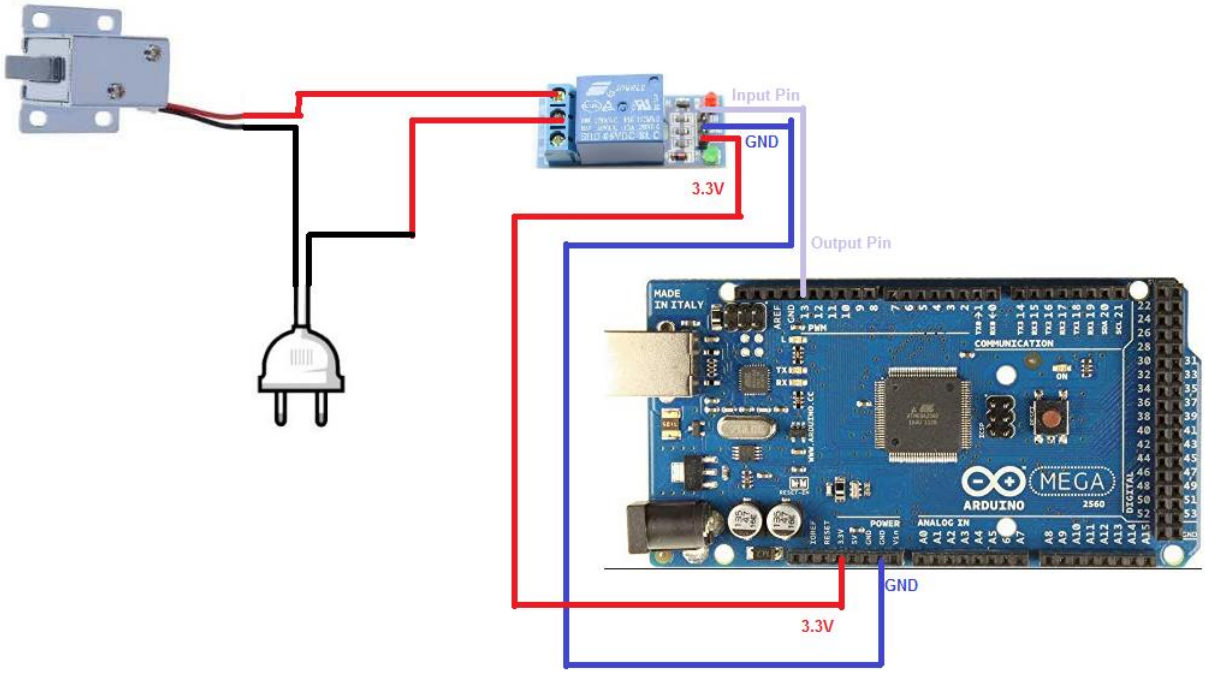
El diseño general del prototipo, se podrá dividirá en 2 partes principalmente, el diseño de hardware y el diseño de software. Ambas partes van de la mano, por lo mismo es difícil separarlas en un apartado a cada una. A lo anterior se le puede agregar el hecho de que muchas veces el software puede sustituir al hardware o viceversa, por ello para saber que partes de hardware usar se debe pensar al mismo tiempo en el código o la parte de software que implementaría, y ocurre lo mismo para el software.
Por ello la parte del diseño se dividirá en los bloques que se han desarrollado explicando cómo convergen las pares de software y hardware.
La forma en la que se divide el proyecto no por ser modular, implica que lleven un orden los módulos, es decir. Esto ayuda a simplificar el trabajo, y que al momento de reunir nuestros módulos todos funcionen por separado, mejorando la integración segura de los mismos, ya que ningún módulo se podrá integrar a la parte final del proyecto si no funciona correctamente y de manera fluida con los otros módulos.

4.2 Descripción y desarrollo de elementos

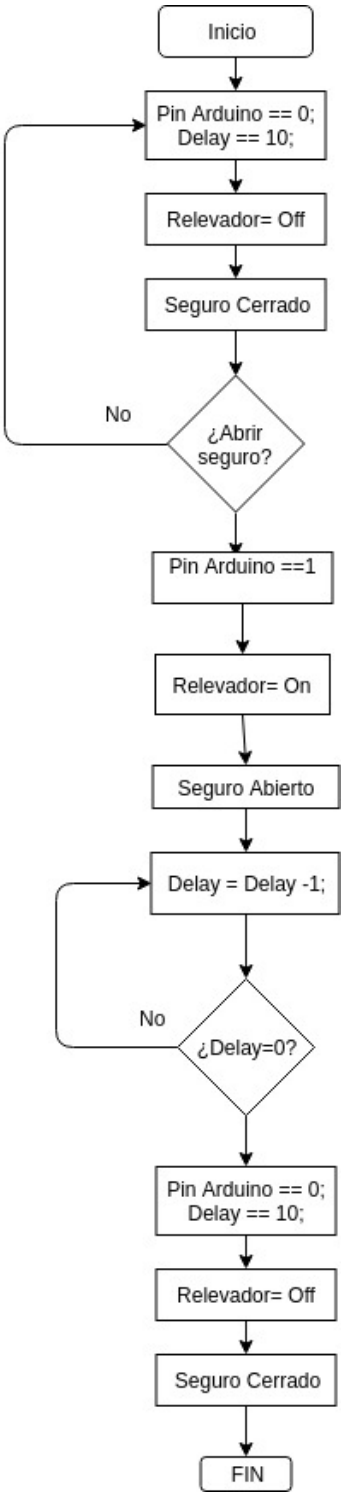
SEGURO ELECTROMAGNETICO

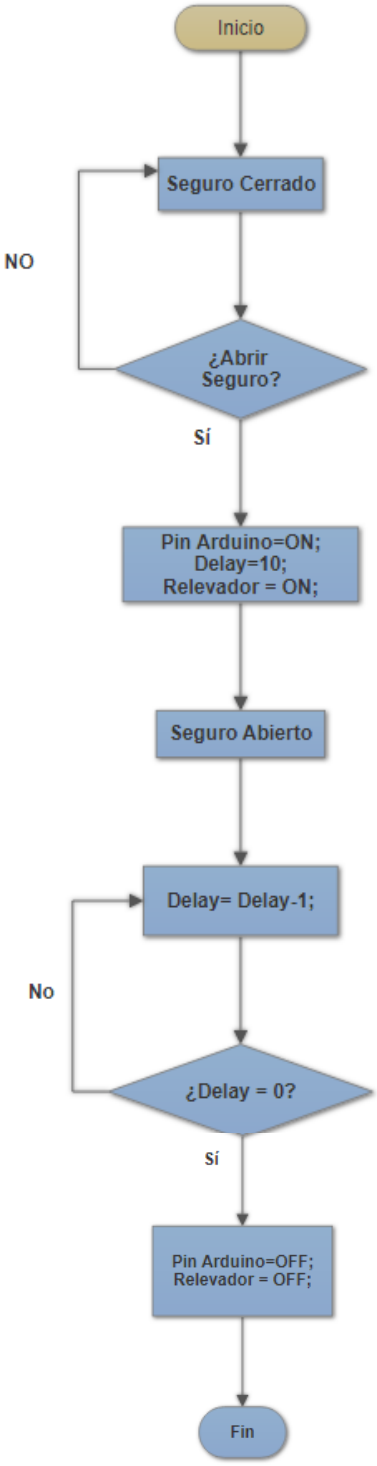
El Seguro que se emplea para cerrar la puerta funciona mediante un electroimán.
Inicialmente el electroimán estará desactivado, es decir no genera ningún campo magnético, lo cual hace que la cerradura permanezca cerrada hasta que sea alimentada con una fuente de 12 volts lo que propicia que el electroimán actué sobre el eje y lo retraiga, es decir se abre el seguro.

La manera de conectar el Arduino al seguro es empleando un relevador y el seguro será activado o desactivado mediante este relevador, mismo que también nos sirve como temporizador.
El tiempo de activación del seguro será controlado mediante un pin de salida del Arduino que a su vez está conectado al relevador. El seguro electromagnético será energizado para brindar acceso al contenido del casillero deseado. Por lo tanto poder abrirla o cerrarla de una manera automatizada sin la necesidad de emplear una llave es uno de los puntos principales del proyecto.



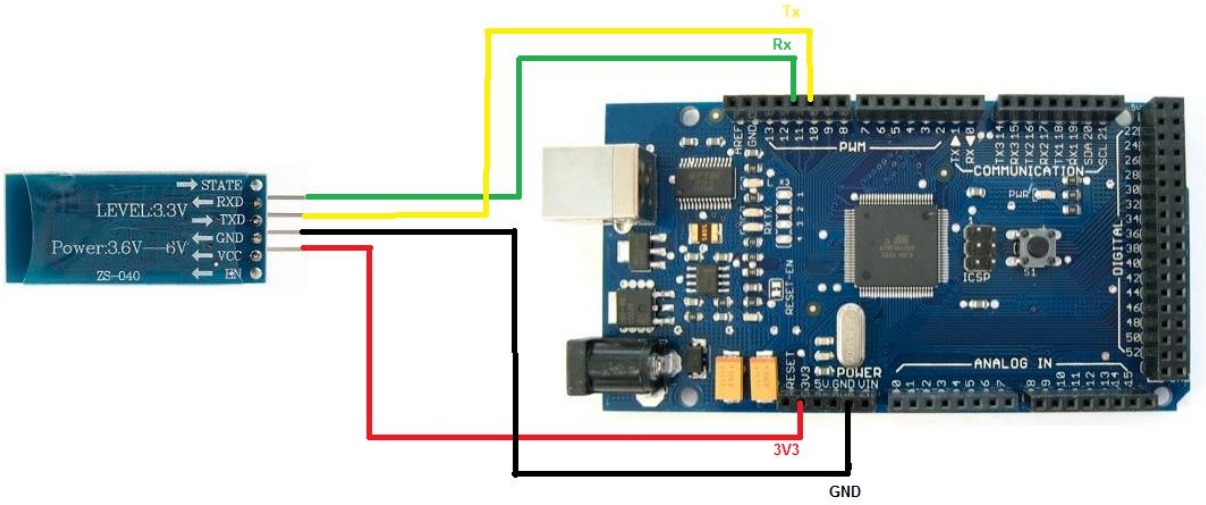
El diagrama de flujo para el programa que controla la apertura del seguro es el siguiente:


Inicio
Pin Arduino == 0;
Delay == 10;
Relevador= Off
Seguro Cerrado
No
¿Abrir seguro?
Pin Arduino ==1
Relevador= On
Seguro Abierto
Delay = Delay -1;
No
¿Delay=0?
Pin Arduino == 0;
Delay == 10;
Relevador= Off
Seguro Cerrado
FIN

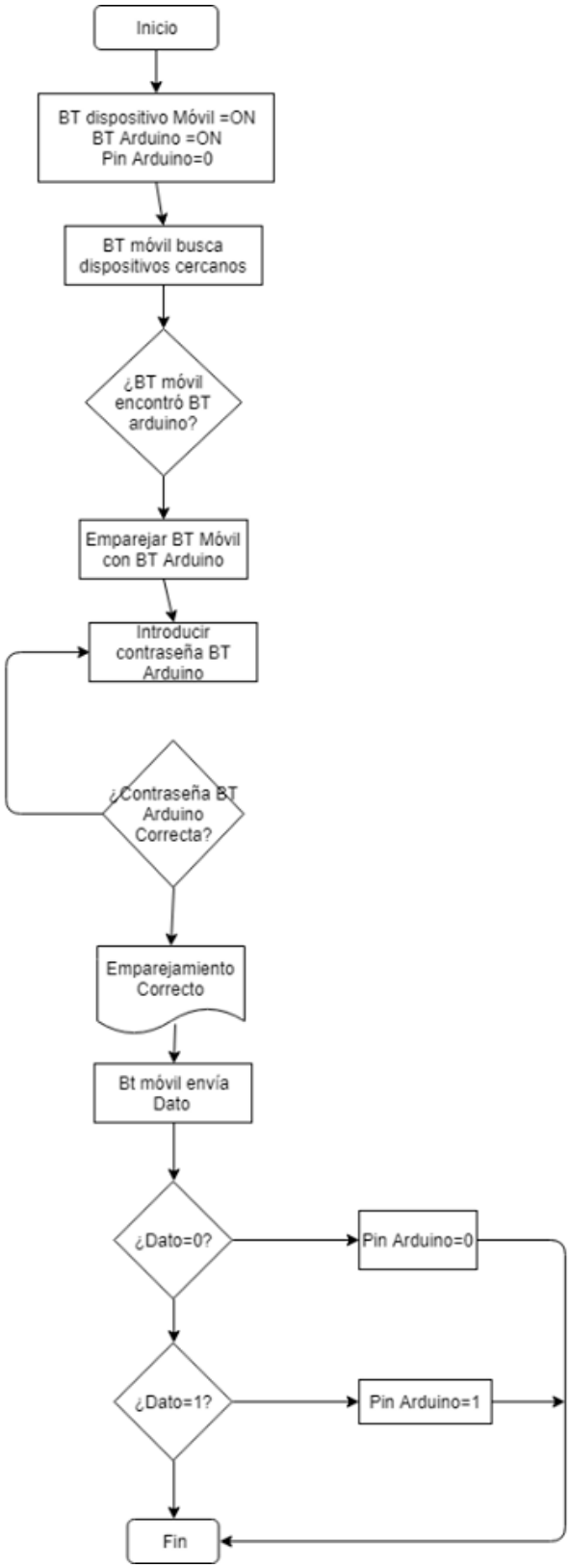

Inicio
Seguro Cerrado
NO
¿Abrir Seguro?
Sí
Pin Arduino=ON;
Delay=10;
Relevador = ON;
Seguro Abierto
Delay= Delay-1;
No
¿Delay = 0?
Sí
Pin Arduino=OFF;
Relevador = OFF;
Fin

BLUETOOTH Y ARDUINO

El módulo HC05 se conectará al dispositivo móvil empleando tecnología Bluetooth. Esta tecnología permite la transmisión y recepción de datos de manera inalámbrica.
Este módulo funcionará como esclavo, y el dispositivo móvil fungirá como maestro, esto quiere decir que el dispositivo móvil será el que busque el módulo y se conecte al mismo.
Después de que el dispositivo móvil haya hecho el emparejamiento, mandará distintos datos mediante Bluetooth al módulo, este los interpretará y usando el puerto serial los enviará a la placa.
La placa a su vez se encargará de realizar distintas tareas, desde activar alguno de los relevadores hasta realizar lectura o registro de huellas dactilares.

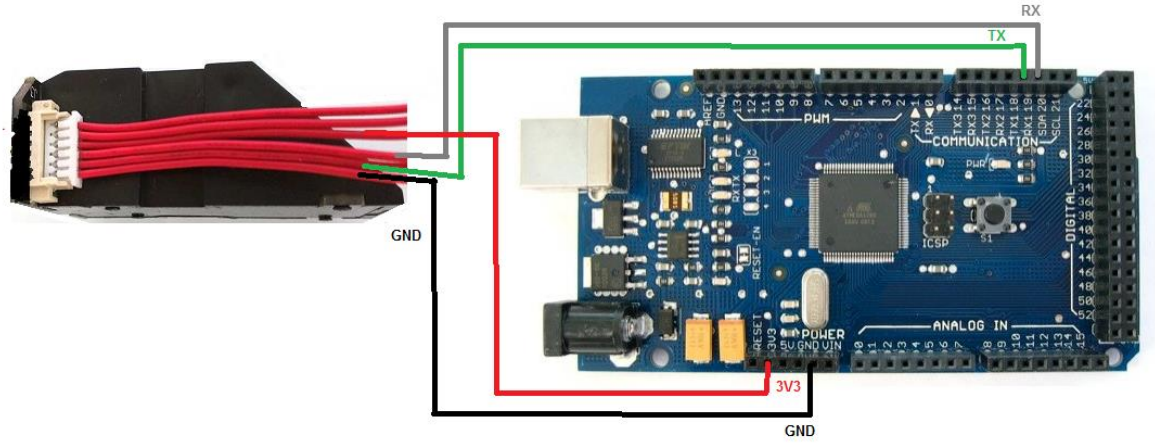


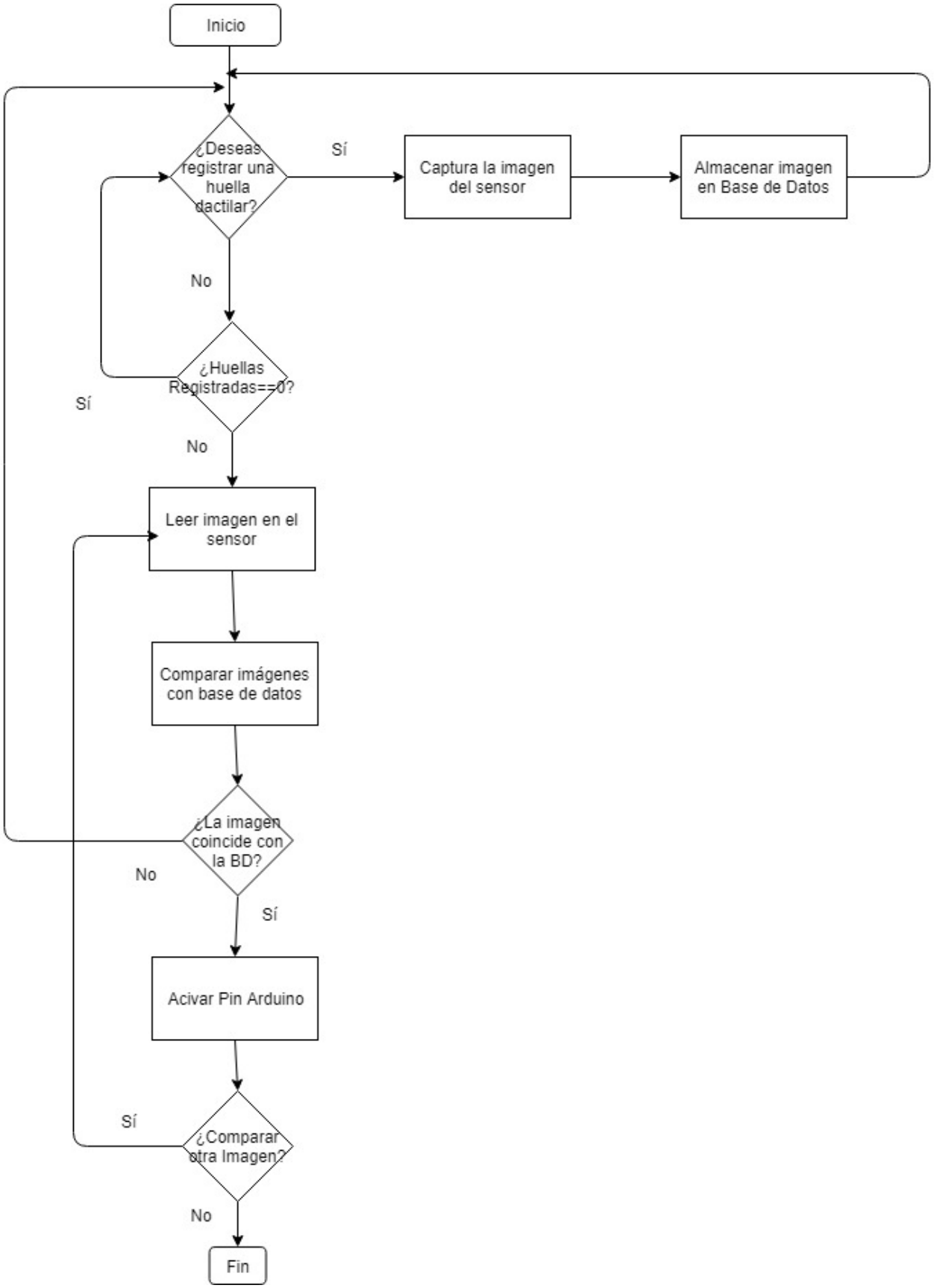
En el siguiente diagrama de flujo, se muestra el emparejamiento, y el ejemplo e interpretación de algunos datos que puede enviar el dispositivo móvil, que en este caso únicamente prenderá o apagará algún Pin del Arduino


Inicio
BT dispositivo Móvil =ON
BT Arduino =ON
Pin Arduino=0
BT móvil busca dispositivos cercanos
¿BT móvil encontró BT arduino?
Emparejar BT Móvil con BT Arduino
Introducir contraseña BT Arduino
¿Contraseña BT Arduino Correcta?
Emparejamiento Correcto
Bt móvil envía Dato
¿Dato=0?
Pin Arduino=0
¿Dato=1?
Pin Arduino=1
Fin

MÓDULO BIOMÉTRICO

El sensor biométrico se encargará de capturar imágenes para realizar el registro de huellas dactilares. Las huellas registradas se almacenarán en una base de datos y cuando el sensor detecte alguna imagen procederá a compararla con la base de datos, si esta coincide pondrá en alto un pin del Arduino, el cual estará conectado al relevador y este a su vez con el seguro.




Inicio
¿Deseas registrar una huella dactilar?
Sí
Captura la imagen del sensor
Almacenar imagen en Base de Datos
No
¿Huellas Registradas==0?
Sí
No
Leer imagen en el sensor
Comparar imágenes con base de datos
¿La imagen coincide con la BD?
No
Sí
Acivar Pin Arduino
¿Comparar otra Imagen?
Sí
No
Fin

SEGURO MECÁNICO

El seguro mecánico será un seguro de emergencia, ya que solo se empleará cuando la corriente eléctrica sea cortada.
Solo el superusuario y los dependientes tendrán acceso a este tipo de seguro, y se usará una cámara de seguridad, para saber quiénes son las personas que usan este método de apertura.
Este seguro mecánico estará conectado al seguro eléctrico de tal manera que podría considerarse un seguro híbrido, ya que se

La apertura mecánica no interferirá con la apertura eléctrica y viceversa. Gracias a la forma en la cual se están construyendo y por ello se puede considerar un seguro híbrido, el cual se puede abrir de manera eléctrica y de manera mecánica, brindando mayor seguridad a los casilleros y a los usuarios.

La siguiente ilustración muestra cómo están conectados ambos seguros y su forma de apertura mecánica:

A pesar de ser una manera muy simple de solucionar el problema, ya que unicamente se conectará mediante algún alambre el seguro mecánico al electrico, cumple completamente con el cometido.

SWITCH PARA DESACTIVAR ON/OFF

Un switch externo podrá controlar todos los dispositivos, es decir alimentar todo el sistema, de tal manera que la única forma de apertura sea usando el seguro mecánico.

APLICACIÓN ANDROID

La aplicación para los sistemas Android será una de las partes cruciales, ya que empleando algún dispositivo que use este sistema operativo (incluyendo celulares y tabletas) será la manera local con la cual podremos abrir nuestro casillero. Esto se realizará mediante el bluetooth de nuestros dispositivos al conectarlo al Arduino empleando este mismo medio, es decir la comunicación que existirá entre Arduino y el celular o la Tablet, será mediante bluetooth.

Lo anterior permitirá que la conexión sea únicamente local, evitando en gran medida los posibles ataques a la plataforma, Ya que solo los aparatos sincronizados y de confianza serán los que tendrán acceso a los diferentes casilleros.
También es importante mencionar que la conexión mediante bluetooth nos ahorrará hardware controlado por el Arduino, específicamente el uso de keypad, el cual puede resultar problemático al momento de programar y de realizar llamadas al mismo, esto debido al tiempo de respuesta que ofrece el keypad.

Por todo lo antes mencionado se propone tener un dispositivo Android que se encuentre conectado al Arduino de manera permanente y cuya única función sea ejecutar la aplicación de conexión de nuestro Arduino.

La aplicación desarrollada para este dispositivo contará con las siguientes actividades:

0.0 Conexión Bluetooth

1.0 Menú Master

- 1.0 Inicio de Sesión Súper Usuario
  - 1.1 Menú Súper Usuario
    - 1.1.1 Abrir Casilleros
    - 1.1.2 Registro de usuario
    - 1.1.3 Borrar usuario
    - 1.1.4 Registrar Huella usuario
    - 1.1.5 Registro de dependiente
    - 1.1.6 Borrar dependiente
    - 1.1.7 Registrar Huella dependiente
    - 1.1.8 Información de los casilleros
- 2.0 Inicio de Sesión Dependiente
  - 2.1 Menú Dependiente
    - 2.1.1 Abrir Casilleros
    - 2.1.2 Registro de usuario
    - 2.1.3 Borrar Usuario
- 3.0 Inicio de Sesión Usuario
  - 3.1 Menú Usuario
    - 3.1.1 Abrir Casilleros

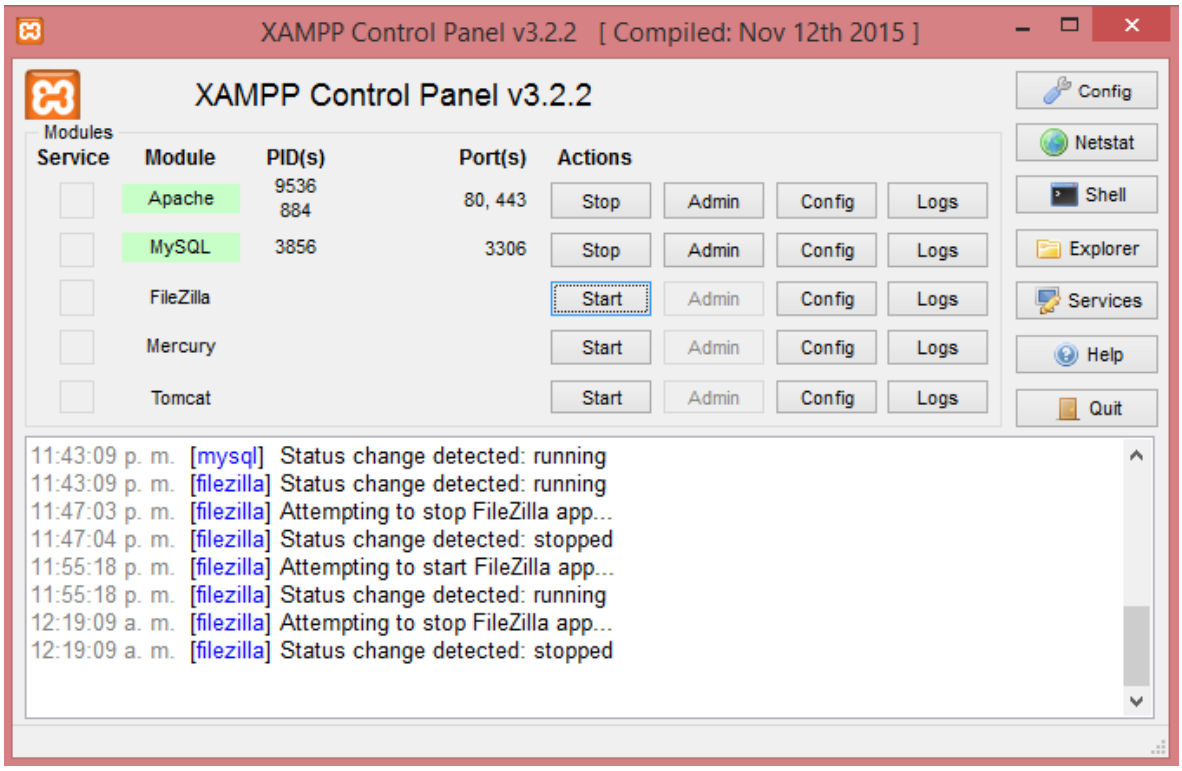
Base de Datos (BD)

La base de datos requerirá un servidor web en el cual se almacenará toda la información, también se necesitará un gestor de BD, y por último necesitaremos un intérprete del lenguaje PHP, ya que este lenguaje nos permitirá conectarnos a la BD desde la aplicación.

Todas las herramientas que se necesitan para el funcionamiento de la BD, las podemos encontrar en distintas multiplataformas, tales como XAMPP, WAMP, LAMP, APPSERVER, etcétera.
Para las MIBoS, se usará XAMPP, por su fácil instalación, y por su simplicidad en la interfaz, ayudándonos a administrar de manera sencilla nuestros servicios.

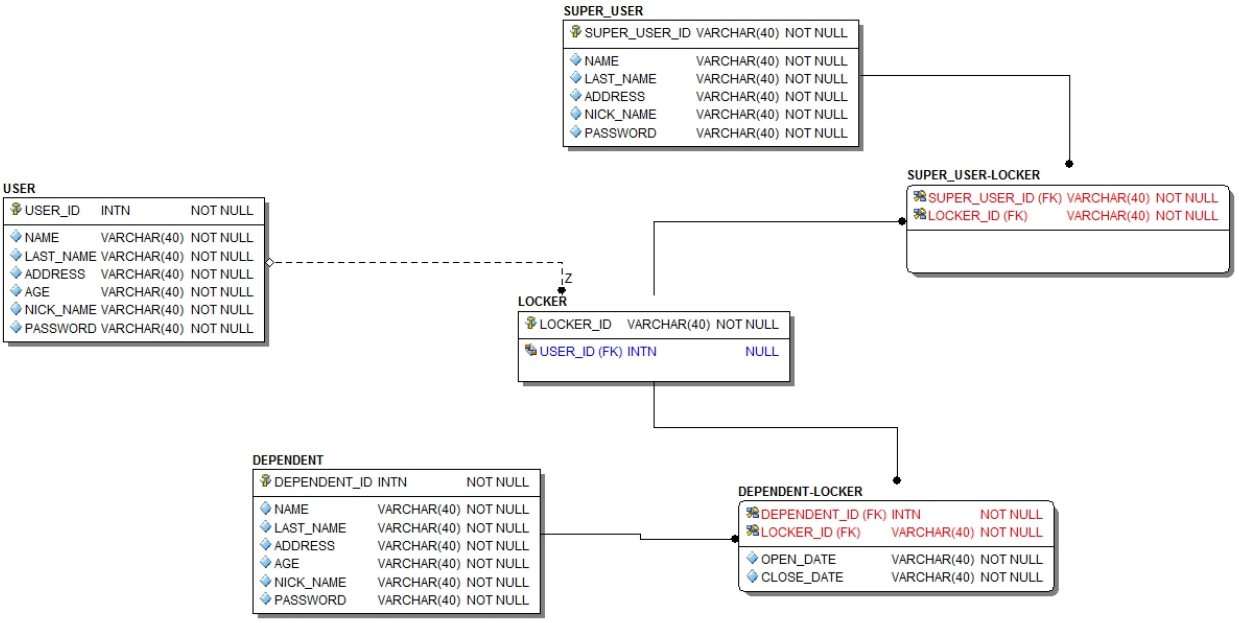
XAMPP Es una multiplataforma de código libre con un conjunto de soluciones, que consiste principalmente en un sistema de gestión de bases de datos MySQL, un servidor web apache, y los interpretes de scripts escritos en los lenguajes PHP y Pearl. Por lo tanto XAMPP será la multiplataforma que emplearemos para nuestro desarrollo de la base de datos.

En la siguiente imagen se muestra como de inicia Apache y MySQL simplemente abriendo la aplicación y dando click al botón start:

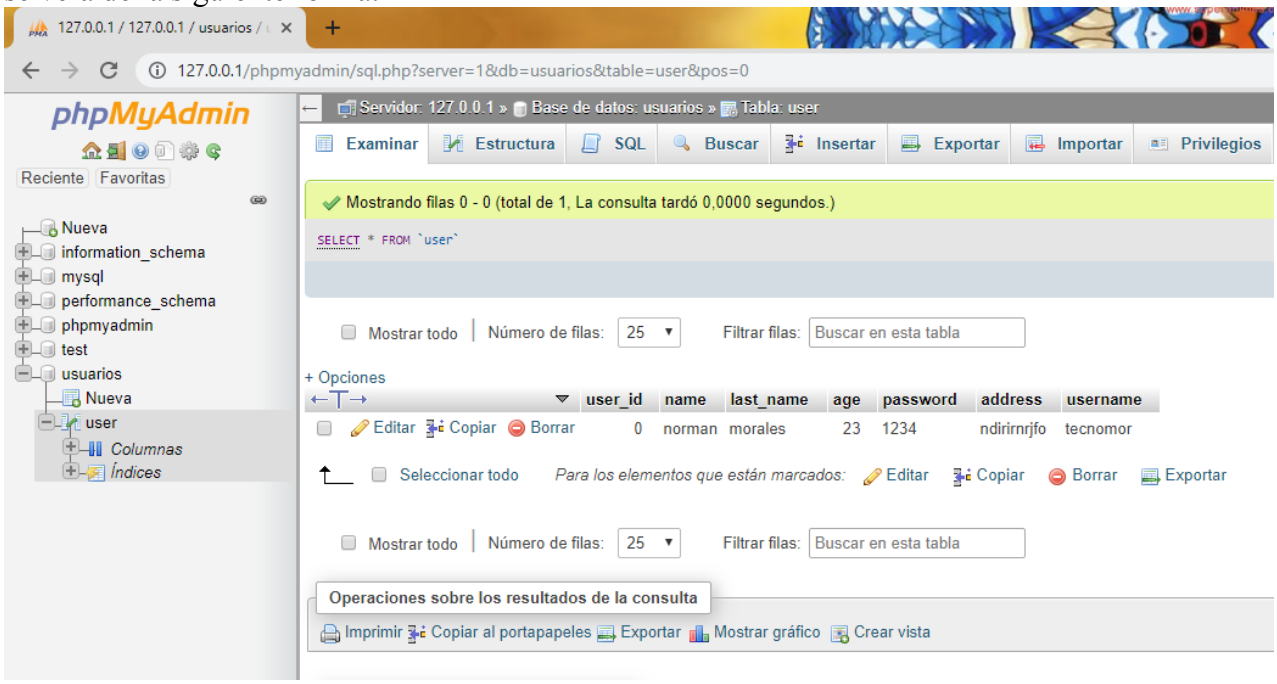


La base de datos será la encargada de almacenar toda la información de los tres tipos de usuarios.
El primer usuario será el Super Usuario, es decir el dueño del local, los datos que se almacenarán de este tipo de usuario serán: Nombre, Nickname y contraseña.
La información de los dependientes y clientes deberá ser más detallada, incluyendo los siguientes datos: Nombre completo, dirección, edad, fecha de nacimiento, nickname y contraseña.
Por otro lado, se deberá almacenar cierta información al momento de la apertura de un casillero. Se deberá almacenar el nombre del usuario que abrió el casillero, junto con la fecha y hora de su apertura. Es decir se deberá guardar un historial de aperturas en la base de datos.

El modelo lógico de la BD que cumple con los requerimientos es el siguiente:



Toda la información será recabada en la aplicación, y esta a su vez será almacenado en un servidor Apache, empelando el gestor MySQL y se agregará la información a las tablas mediante scripts en PHP.
El servidor local iniciado, en el cual se agregarán todas las tablas y la información de las mismas se verá de la siguiente forma:

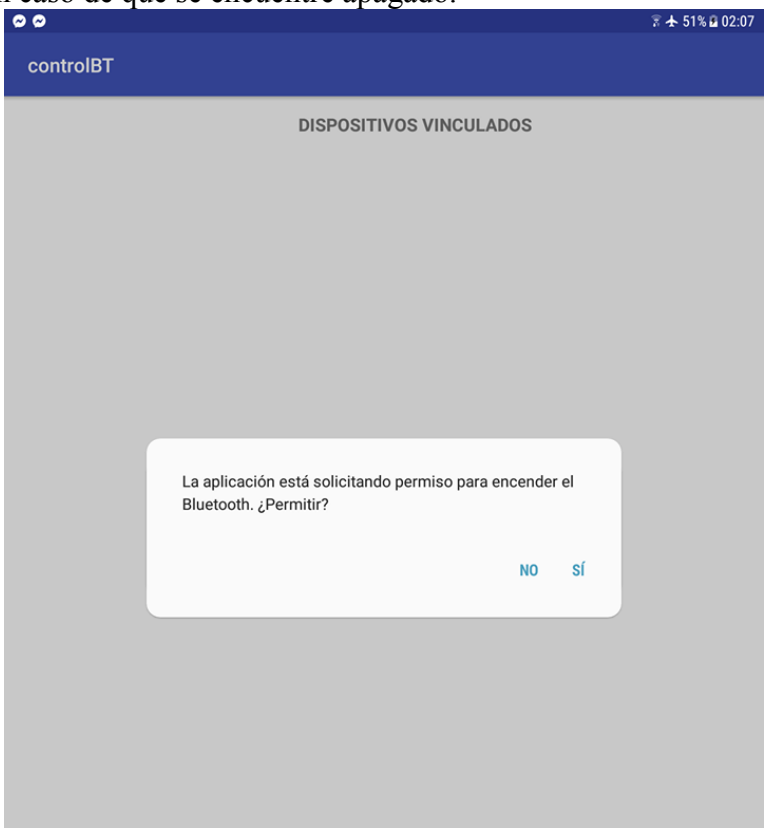


Como se observa en la imagen anterior, crear base de datos, agregar tablas y administrar las mismas tablas en un servidor local se realiza de manera sencilla empleando XAMPP.

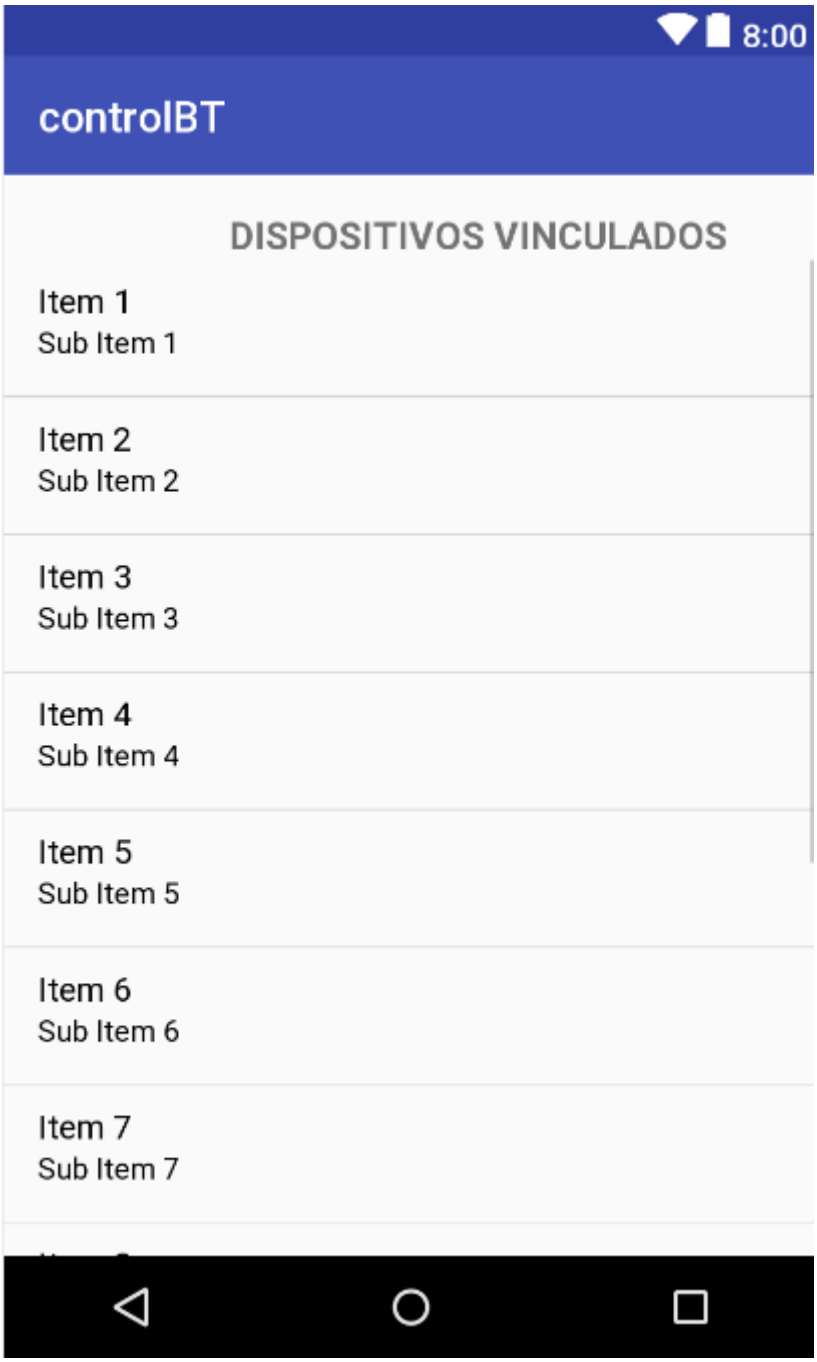
4.3 Integración de partes

Las actividades de la aplicación móvil serán las encargadas de controlar varias partes de las MIBoS y de integrar las mismas.
La aplicación móvil se realiza mediante actividades, las cuales nos sirven para modular el desarrollo de la misma.
El desarrollo de las actividades incluyen dos partes, un archivo .xml encargado de la parte del diseño, y un archivo .java en donde se encuentran los algoritmos que empleará cada actividad.
A continuación se describen las distintas actividades y los componentes que emplearán dichas actividades para el desarrollo del proyecto.

ACTIVIDAD 0.0 CONEXIÓN
La actividad 0.0, se considera como una actividad pre-inicial, la cual nos permitirá conectar el dispositivo móvil con el Arduino empleando la tecnología Bluetooth del dispositivo y del módulo HC05.
El Hardware y las herramientas necesarias para esta conexión son las siguientes:
-Android Studio: Emplearemos esta herramienta para crear una vista que active el bluetooth del dispositivo móvil en caso de que se encuentre apagado.



Después desplegará una lista de los dispositivos vinculados mediante Bluetooth y seleccionar el dispositivo al cual se deberá conectar la aplicación móvil (Ver apéndice 9).

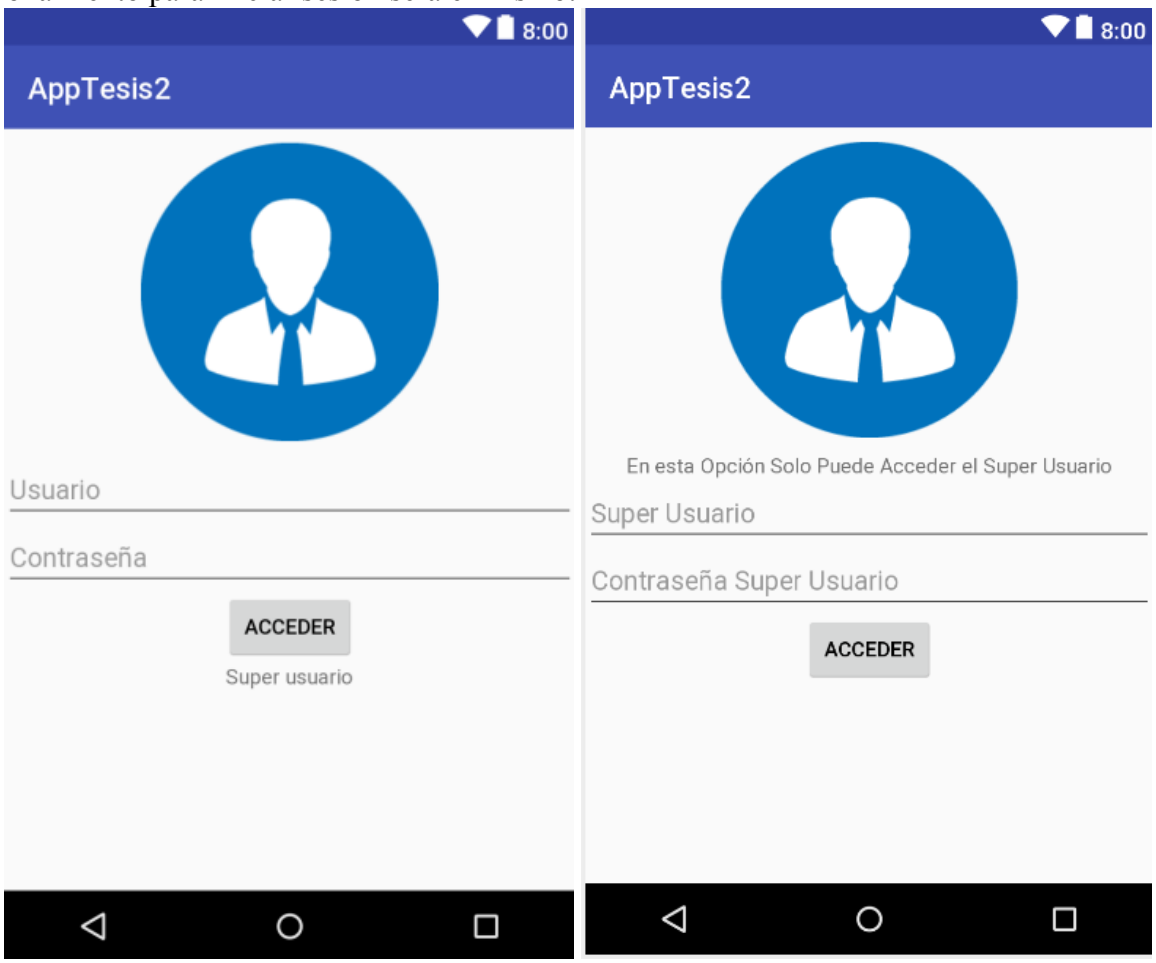


-Arduino: Funcionará como fuente de alimentación para el módulo Bluetooth, y como intérprete de los datos que envié y reciba de este mismo módulo.
-Modulo Hc05: Este módulo será conectado al Arduino y a la aplicación móvil, ya que empelando la tecnología Bluetooth permitirá la transferencia de información entre estas dos herramientas,

ACTIVIDAD 1.0, 2.0 Y 3.0 INICIO SESIÓN

La actividad 1.0, 2.0 y 3.0 son actividades dedicadas al inicio de sesión para los distintos usuarios y deberán estar conectadas a la base de datos montada en un servidor local. Estas actividades

realizará una lectura de los datos introducidos (usuario y contraseña) para compararlos con los datos almacenados en la BD, y en caso de que coincidan pasar a la actividad siguiente (Actividad), en caso contrario se mostrará un mensaje de error.
Las herramientas necesarias para realizar el inicio de sesión de super usuario son:
-Android Studio: Se usará Android Studio para crear una actividad que pueda recibir dos datos diferentes, el usuario y la contraseña. Y un botón de acceso el cual nos abrirá la siguiente actividad en caso de que los datos introducidos sean correctos, o un mensaje de error en caso contrario (Ver apéndice 10).
El inicio de sesión cambiará de diseño según el tipo de usuario, de esta manera se podrá acceder al inicio de sesión de super usuario en desde el menú de inicio de sesión de usuario normal, pero el funcionamiento para iniciar sesión será el mismo:



- XAMPP: Mediante MySQL creará las tablas donde se almacenará la información de los 3 tipos de usuarios, también servirá para levantar el servidor web local empleando apache. Y con su intérprete de PHP modificará la información en las tablas de usuarios.
- Script PHP: Los scripts son códigos que estarán almacenados en el servidor Apache, y su función principal será obtener los datos que se introduzcan en la aplicación móvil, mandar la información a la base de datos, y después mandar la respuesta del servidor a la aplicación (Ver apéndice 11).

ACTIVIDAD 1.1, 2.1 y 3.1
La actividad que sigue dependerá del tipo de usuario que haya iniciado sesión, ya que será un menú personalizado para el tipo de usuario. Algunos menús comparten
La actividad para el menú del super usuario (Actividad 1.1) será el menú con mayor número de opciones, y cada opción nos redireccionará a una actividad nueva (Ver apéndice 12).Las opciones que nos brinda esta actividad son las siguientes:

-Registro de dependiente
-Registro de usuario
-Registrar Huella dependiente
-Registrar Huella Usuario
-Borrar Dependiente
-Borrar Usuario
-Abrir Casilleros

El menú para Dependientes (Actividad 2.1), contendrá algunas de las opciones que tiene el super usuario en su menú (Ver apéndice 13):

-Registro de usuarios
-Abrir Casilleros
-Borrar Usuario

Y el último menú, será el menú al cual podrán acceder los usuarios normales (Actividad 3.1). Este menú sólo tendrá una opción de todas las que tiene el super usuario (Ver apéndice 14):

-Abrir Casilleros

Como se describe anteriormente, todos los menús son muy similares, lo único que se cambia entre cada uno es la disminución de opciones según el tipo de usuario que haya accedido a la aplicación.

Para realizar estas actividades de menú, la única herramienta necesaria será Android Studio, ya que se toma en cuenta que el dispositivo móvil está conectado al módulo HC05 y por ser solo un menú, no realizará ningún tipo de modificación a la base de datos.

ACTIVIDADES 1.1.1, 2.1.1, 3.1.1 ABRIR CASILLEROS

Desde el menú de cualquier tipo de usuario se podrá seleccionar la opción de abrir casilleros, opción que abrirá otra actividad dependiendo del tipo de usuario que haya accedido.
La actividad para abrir casilleros, será la misma para el super usuario, que para los dependientes, ya que ellos tendrán la oportunidad de abrir cualquiera de los casilleros, así se encuentren ocupados, o libres. En cambio los usuarios solo podrán acceder a los casilleros que se encuentren libres.
Sin importar esto, el diseño de la actividad será el mismo para los 3 tipos de usuarios.
Las herramientas que empleará esta actividad son las siguientes:

-Android Studio: La actividad que se desarrollará deberá mostrar el estatus de todos los casilleros, es decir, mostrará si los casilleros se encuentran ocupados o desocupados, y para el caso del super usuario o dependientes, podrán abrir el casillero que deseen sin importar su estatus, a diferencia del usuario normal, que solo podrá abrir un solo casillero que se encuentre libre. También se deberá de encargar en mandar información mediante bluetooth al arduino en cuanto algún usuario seleccione un casillero, esta información deberá de cambiar dependiendo del casillero que se desee abrir.

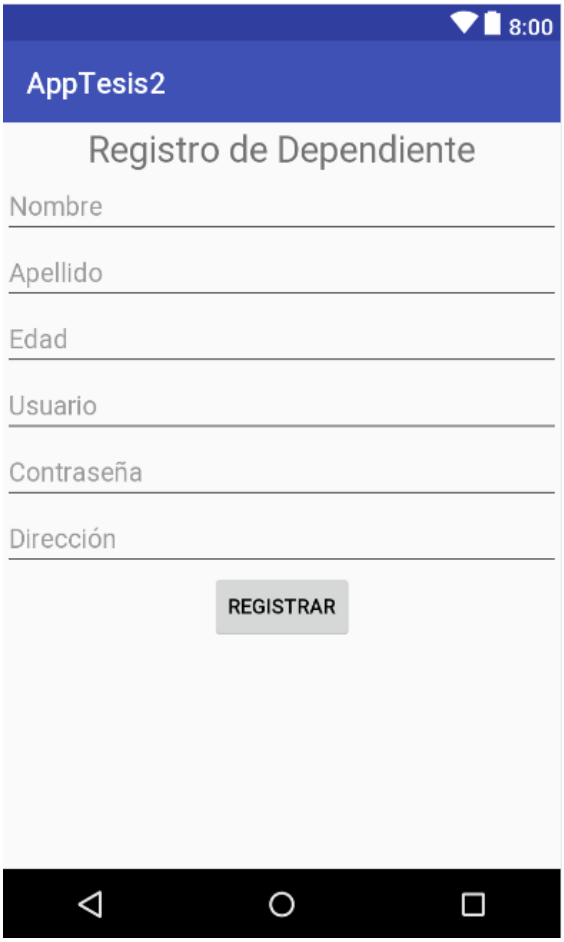
- XAMPP: Empleando MySQL crearemos las tablas para almacenar a la información de los casilleros, y las tablas que almacenarán la información entre los usuarios y los casilleros que hayan tenido interacción. Y como en otras actividades se necesitará esta herramienta para tener activo el servidor web local y como intérprete de PHP.
- Script PHP: El Script para esta actividad consistirá en informar el estatus de todos los casilleros, revisando la base de datos, para que después esta información sea mandada a la actividad y despliegue el estatus. El script deberá cambiar este mismo estatus (ocupado a desocupado y viceversa), y de actualizar la tabla donde se almacena la información sobre que usuarios abren que casilleros.
-Sensor biométrico: Las huellas que haya almacenado el sensor biométrico permitirán el cambio de estado de los pines de los casilleros que se encuentren en el estatus de desocupados
-HC05: El módulo servirá de conexión e interprete entre el dispositivo móvil y el Arduino.
-Arduino: Al recibir la información de la actividad mediante el HC05 o mediante el sensor biométrico, el Arduino deberá de interpretarla, identificando cual es el casillero que se desea abrir, y activará el pin de salida correspondiente a ese casillero. La activación del pin de salida solo durará 30 segundos aproximadamente, regresando el pin a su estatus original pasado este tiempo.
-Relevador: Se activará o se cerrará dependiendo del estatus del pin de salida del arduino.
-Seguro electromagnético: Se abrirá o cerrará dependiendo del estatus del relevador, permitiendo el acceso al casillero.

Al finalizar la apertura de casilleros, se deberá cerrar sesión para que otro usuario pueda acceder a la aplicación.

ACTIVIDAD 1.1.2, 2.1.2 Y 1.1.5 REGISTRO DE USUARIO/DEPENDIENTE

La actividad de registro de usuario obtendrá los datos completos de los usuarios, para almacenarlos en la BD, mismos datos que serán consultados para iniciar sesión en la aplicación. Los datos que se solicitan serán los mismos para los dependientes como para los usuarios, por lo tanto el funcionamiento de las actividades 1.1.2, 2.1.2 y 1.1.4 será prácticamente el mismo, salvo que en el diseño se mostrará a quien se está registrando y en la base de datos se almacenará en su respectivas tablas (Si se registra a un usuario se almacenará en la tabla de usuarios, y si se registra a un dependiente se almacenará en la tabla de dependientes).

La comparación del diseño entre el registro de usuario y el registro de dependiente se muestra a continuación:



Sólo los super usuarios podrán acceder a la actividad para registrar a dependientes (Actividad 1.1.4), en cambio para registrar a los usuarios normales, el super usuario y los dependientes podrán realizar esta acción (Actividades 1.1.2, 2.1.2).
Esta actividad necesitará unicamente herramientas de software.

-Android Studio: Se diseñara una actividad en la cual mostrará los datos que deberá llenar el solicitante para convertirse en usuario o dependiente, los datos incluyen nombre, apellidos, fecha de nacimiento, dirección, teléfono, nick name, y contraseña (Ver apéndice 15).

-XAMPP: Aquí se crearan las tablas en las cuales se almacenará la información de los usuarios y de los dependientes. También se levantará el servidor local (Ver apéndice 16).

-Script PHP: Obtendrá la información que se coloque en la actividad para después enviarla a la base de datos y almacenarla en la tabla correspondiente.

Después de crear el registro de usuario o dependiente, la aplicación podrá admitir que se realice otro registro o que se regrese a la actividad inicial para que otro usuario acceda a la aplicación.

ACTIVIDAD 1.1.3, 2.1.3, 1.1.6 BORRAR USUARIO/DEPENDIENTE

El diseño de la actividad para borrar usuarios será el mismo que el diseño para borrar a los dependientes pero solo los super usuarios podrán borrar a los dependientes de la base de datos.

En esta actividad se emplearán las mismas herramientas que se emplean en la actividad para realizar registros de usuarios o dependientes.

-Android Studio: En esta actividad solo se solicitará el nick name del usuario o dependiente que se desee eliminar (Ver apéndice 17)

-XAMPP: En este caso solo se utilizará esta herramienta para levantar el servidor local, y como intérprete de PHP, para tener acceso a las tablas de la base de datos.

-Script PHP: El script de esta actividad consistirá en localizar la información del nick name que haya sido introducido, para eliminar su registro en la tabla de usuarios manteniendo la información del usuario o dependiente eliminados en las demás tablas.

Al finalizar la eliminación de algún usuario o dependiente se permitirá eliminar a otros usuarios o dependientes, en caso contrario se puede regresar al inicio de la aplicación para permitir el acceso a otros usuarios.

La comparación del diseño entre las actividades para borrar a los usuarios o a los dependientes se muestra a continuación:

ACTIVIDAD 1.1.4, 1.1.7 REGISTRAR HUELLA DE USUARIO/DEPENDIENTE

El registro de huellas será una actividad en la cual solo el super usuario podrá acceder. Y se accederá a esta actividad a través del menú de super usuario.

Las herramientas para el desarrollo de esta actividad serán las siguientes:

-Android Studio: Se creará la actividad con un diseño en el cual se escribirá el nombre de usuario o dependiente del cual se agregarán su huella dactilar, el número que tendrá en la base de datos para el registro y mostrará los pasos necesarios para realizar el registro de la huella empleando Bluetooth. También en caso de que no exista el usuario, se desplegará un mensaje de error.

-XAMPP: Se usará esta herramienta para levantar el servidor local, y como intérprete de PHP.

-Script PHP: El Script nos ayudará a buscar en la base de datos si el nombre de usuario o del dependiente que haya sido introducido en la actividad se localiza en la base de datos.

-HC05: El módulo servirá de conexión e interprete entre el dispositivo móvil y el Arduino.

-Arduino: El arduino mandará y recibirá las señales mediante Bluetooth para realizar el procedimiento necesario para almacenar las imágenes que capture el sensor biométrico.

- Sensor Biometrico: Tomara capturas de la huella dactilar que sea colocada y las almacenará en la memoria del arduino.

ACTIVIDAD 1.8 INFORMACIÓN DE LOS CASILLEROS

Solo el super usuario podrá acceder a esta información, en la cual se encontrará toda la información respecto a la interacción con los casilleros. La información que podrá ver de los casilleros es la siguiente:

Numero de casilleros
Estatus de casilleros
Historial de aperturas

En el historial de aperturas, se podrá saber que usuarios abrieron ese casillero, en que día y en que horario.

-Android Studio: El diseño de la actividad consistirá en un diseño parecido al de la actividad para abrir los casilleros, en el cual se mostrarán todos los casilleros y la información e historial de los casilleros.

-XAMPP: Se crearán las tablas necesarias para acceder a la información e historial de los | casilleros. Y también nos servirá como intérprete de PHP y para levantar el servidor local donde se almacenarán las tablas nuevas.

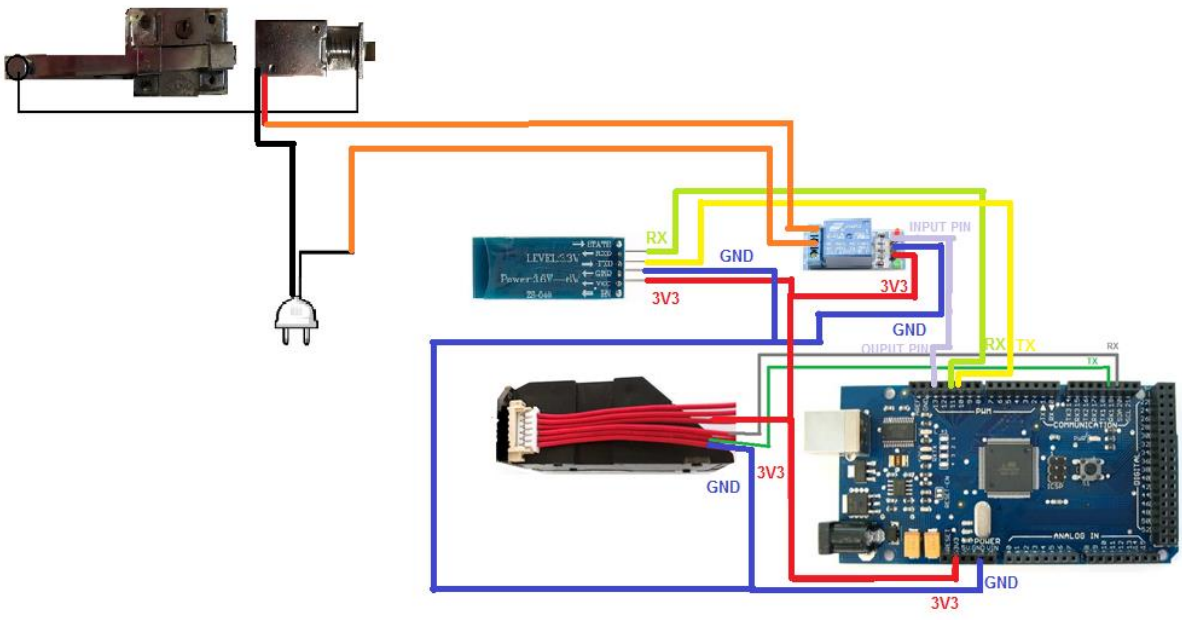
- Script PHP: Accederá a la información almacenada en la base de datos de todos los casilleros, para mandar la información requerida a la actividad.

4.4 Integración final
La primer parte de la integración consiste en conectar todas las herramientas disponibles al Arduino (HC05, sensor biométrico, relevadores y seguros electromagnéticos) y cargar la aplicación móvil (con todas las actividades antes mencionadas) al dispositivo móvil.
Después se levantará el servidor local desde una computadora, se crearán las tablas correspondientes y por último se guardarán los Scripts de PHP que requiere el proyecto.
Al iniciar la aplicación móvil y alimentar el Arduino, la aplicación se sincronizará con el módulo HC05 y pasaremos al menú de las posibles actividades a ejecutar.
Y desde ese menú se accederá a las distintas actividades.

Pruebas realizadas

Las pruebas realizadas para la integración final se encuentran mencionadas en cada una de las actividades anteriores que tienen referencia hacia el apéndice, donde se encuentra el código que se empleó para realizar dichas actividades descritas.
Las actividades que no estén referenciadas hacia el apéndice es por qué no se pudieron hacer las pruebas necesarias para integrarlas, por lo tanto solo incluye su diseño.

Diseño final de Hardware
El diseño final consiste en la integración de todos los elementos antes descritos. La primer parte de Hardware puede ser variable, ya que dependerá únicamente del dispositivo móvil empleado. Por otro lado el resto de hardware empleado estará conectado al Arduino de la siguiente manera:



Este diseño solo involucra un solo seguro (Ver apéndice 18), pero puede ser modificado para que use todos los pines de salida disponibles del Arduino.

SELECCIÓN DE CASILLEROS

EL diseño, tamaño, cantidad y materiales de los anaqueles o casilleros que se emplearán, podrán variar dependiendo de las necesidades del dueño. Por lo tanto se podrá ofrecer a los usuarios distintas marcas de casilleros para que ellos mismos decidan cuales son los casilleros que más se podrán adaptar a su negocio.

Cabe destacar que el diseño final de los seguros es adaptable a muchos modelos de casilleros que se encuentran en el mercado actual.

CAPITULO 5 CONCLUSIONES

5.1 Logros del proyecto

El objetivo establecido del proyecto se cumplió de manera exitosa, ya que se diseñó un dispositivo que ayudará con la recepción y entrega de productos mejorando el tiempo de las transacciones, esto gracias a que se logró realizar la base de datos en la cual se almacena la información de los distintos tipos de usuarios y de los estados de los casilleros o anaqueles; empleando una interfaz con buenos rangos de seguridad y de validación para cada usuario. Por otro lado se diseñaron medidas de seguridad física para el resguardo de los equipos y bienes. Adicionalmente el diseño cuenta con un procedimiento de instalación donde se especifican los elementos empleados y los tipos de conexiones, todo esto para facilitar tanto el uso como la instalación y/o mantenimiento y reparación del sistema.

El diseño del sistema se realizó con éxito, gracias a la verificación de ciertos elementos críticos que comprueban la funcionalidad del diseño, y de igual manera estas pruebas permitieron arrojar un estimado de costos.

Concluyendo, se logró cumplir con los propósitos de la tesis, encontrando muchas posibles mejoras a lo largo del diseño y pruebas realizadas.

5.2 Tareas pendientes

Posibles Mejoras

Existen muchas mejoras que se pueden agregar al proyecto para que este se convierta en un proyecto aún más adaptable al mercado y más autónomo.

En la aplicación, una de las primeras mejoras consiste en el almacenamiento de la información. Donde se podría buscar que la base de datos se encuentre almacenada en el dispositivo móvil y no en un servidor local que tenga que levantarse desde una computadora. De igual manera se podría buscar que la aplicación se conecte a un servidor remoto, para acceder a la información desde cualquier punto, cuidando que esta información esté encriptada para la seguridad de los usuarios.

La mejora en los casilleros podría consistir en la construcción de casilleros personalizados y no depender de adaptar el producto a los casilleros que se encuentran actualmente en los mercados. Claro que esto conllevaría un costo extra, pero de igual manera generará mayor valor sobre el producto.

El diseño del hardware puede mejorar y reducir su precio sustancialmente cambiando la placa completa del Arduino Mega por solo el Microcontrolador y diseñar nuestra propia placa de desarrollo, empleando solo los pines que requiere el software, sin desperdiciar energía y dinero en componentes que ocupan espacio.

5.2 Materias Empleadas

El desarrollo de esta tesis necesitó del conocimiento que obtuve en muchas materias cursadas durante la carrera, pero algunas partes resultaron imperativas para el desarrollo del proyecto:

-Computación para ingenieros

-Bases de datos:

-Ingeniería de software

-Administración de proyectos de software

-Redes de datos

-Circuitos eléctricos

-Seminario de titulación

Apéndice

Apéndice 1: Código de implementación del teclado matricial

```
#include <Keypad.h>

const byte ROWS = 4; //four rows
const byte COLS = 4; //three columns
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'#','0','*','D'}
};
byte rowPins[ROWS] = {13, 12, 11, 10}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {9, 8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

void setup(){
  Serial.begin(9600);
}

void loop(){
  char pulsacion = keypad.getKey();

  if (pulsacion != NO_KEY){
    Serial.println(pulsacion);

  }
}
```

Este código primero guarda el número de filas y de columnas para después asignarle un valor según la pulsación. La siguiente parte consiste en asignar las columnas y las filas a pines específicos, en nuestro caso emplearemos desde el pin 6 hasta el pin 13 de nuestro Arduino. En la parte de loop cuando detecta una pulsación imprime su valor.

Apéndice 2: Escaneo de localidad para módulo I2C
El código que se muestra a continuación nos permite saber en que localidad se encuntra el módulo I2C

```
#include <Wire.h>
void setup()
{
  Wire.begin();

  Serial.begin(9600);
  while (!Serial);
  Serial.println("\n Escaner I2C ");
}
void loop()
{
  byte error, address;
  int nDevices;

  Serial.println("Escaneando...");

  nDevices = 0;
  for(address = 1; address < 127; address++ )
  {
    Wire.beginTransmission(address);
    error = Wire.endTransmission();

    if (error == 0)
    {
      Serial.print("Dispositivo I2C encontrado en la dirección 0x");
      if (address<16)
        Serial.print("0");
      Serial.print(address,HEX);
      Serial.println("  !");

      nDevices++;
    }
    else if (error==4)
    {
      Serial.print("Error desconocido en la dirección 0x");
      if (address<16)
        Serial.print("0");
      Serial.println(address,HEX);
    }
  }
  if (nDevices == 0)
    Serial.println("Dispositivos I2C no encontrados\n");
  else
    Serial.println("done\n");
  delay(5000);
}
```

Lo que realiza este código es pasar por cada una de las 128 localidades, y en la localidad que esté conectado un dispositivo nos arrojará el valor hexadecimal en el cual está conectado, y en caso de que no esté conectado ningún dispositivo I2C nos desplegará un mensaje de error.

Apéndice 3: Funciones importantes de librería LiquidCrystal_I2C

Después de escanear la dirección se necesitará una librería que nos ayudé a programar nuestro display, esta librería se llama LiquidCrystal_I2C.

Las principales funciones que usaremos de esta librería son las siguientes:

init()

Inicializa el modulo adaptador LCD a I2C, esta función internamente configura e inicializa el I2C y el LCD.

clear()

Borra la pantalla LCD y posiciona el cursor en la esquina superior izquierda (posición (0,0)).

setCursor(col, row)

Posiciona el cursor del LCD en la posición indicada por col y row(x,y); es decir, establecer la ubicación en la que se mostrará posteriormente texto escrito para la pantalla LCD.

print()

Escribe un texto o mensaje en el LCD, su uso es similar a un Serial.print

scrollDisplayLeft()

Se desplaza el contenido de la pantalla (texto y el cursor) un espacio hacia la izquierda.

scrollDisplayRight()

Se desplaza el contenido de la pantalla (texto y el cursor) un espacio a la derecha.

backlight();

Enciende la Luz del Fondo del LCD

noBacklight();

Apaga la Luz del Fondo del LCD

createChar (num, datos)

Crea un carácter personalizado para su uso en la pantalla LCD. Se admiten hasta ocho caracteres de 5x8 píxeles (numeradas del 0 al 7). Dónde: num es el número de carácter y datos es una matriz que contienen los pixeles del carácter. Se verá un ejemplo de esto más adelante.

Apéndice 4: Hola mundo librería I2C

```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x3F,20,4);  // set the LCD address to 0x3F for a 16 chars and 2 line display

void setup()
{
  lcd.init();                      // inicializa LCD
  // Imprime mensaje en LCD
  lcd.backlight();
  lcd.setCursor(3,0);
  lcd.print("Hello, world!");
}


void loop()
{
}
```

Apéndice 5: COMANDOS AT1

```
#include <SoftwareSerial.h>   // Incluimos la librería  SoftwareSerial
SoftwareSerial BT(10,11);    // Definimos los pines RX y TX del Arduino conectados al Bluetooth

void setup()
{
  BT.begin(9600);       // Inicializamos el puerto serie BT (Para Modo AT 2)
  Serial.begin(9600);   // Inicializamos  el puerto serie
}

void loop()
{
  if(BT.available())    // Si llega un dato por el puerto BT se envía al monitor serial
  {
    Serial.write(BT.read());
  }

  if(Serial.available())  // Si llega un dato por el monitor serial se envía al puerto BT
  {
     BT.write(Serial.read());
  }
}
```

Apéndice 6: COMANDOS AT2

AT COMMAND LISTING	
COMMAND	FUNCTION
AT	Test UART Connection
AT+RESET	Reset Device
AT+VERSION	Query firmware version
AT+ORGL	Restore settings to Factory Defaults
AT+ADDR	Query Device Bluetooth Address
AT+NAME	Query/Set Device Name
AT+RNAME	Query Remote Bluetooth Device's
AT+ROLE	Query/Set Device Role
AT+CLASS	Query/Set Class of Device CoD
AT+IAC	Query/Set Inquire Access Code
AT+INQM	Query/Set Inquire Access Mode
AT+PSWDAT+PIN	Query/Set Pairing Passkey
AT+UART	Query/Set UART parameter
AT+CMODE	Query/Set Connection Mode
AT+BIND	Query/Set Binding Bluetooth Address
AT+POLAR	Query/Set LED Output Polarity
AT+PIO	Set/Reset a User I/O pin

Apéndice 7: Registro de Huellas

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

uint8_t id;

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit Fingerprint sensor enrollment");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }
}

uint8_t readnumber(void) {
  uint8_t num = 0;

  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}
```

```
void loop()                     // run over and over again
{
  Serial.println("Ready to enroll a fingerprint!");
  Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...");
  id = readnumber();
  if (id == 0) {// ID #0 not allowed, try again!
     return;
  }
  Serial.print("Enrolling ID #");
  Serial.println(id);

  while (!  getFingerprintEnroll() );
}

uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }

  // OK success!
```

```
p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
  default:
```

```
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}
```

```
  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
```

Apéndice 8: Registro de Huellas

```
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3);

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);

void setup()
{
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }

  finger.getTemplateCount();
  Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
  Serial.println("Waiting for valid finger...");
}

void loop()                     // run over and over again
{
  getFingerprintIDez();
  delay(50);            //don't ned to run this at full speed.
}

uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
```

```
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK success!

p = finger.image2Tz();
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
p = finger.fingerFastSearch();
if (p == FINGERPRINT_OK) {
  Serial.println("Found a print match!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_NOTFOUND) {
  Serial.println("Did not find a match");
  return p;
} else {
```

```
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);

  return finger.fingerID;
}

// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);
  return finger.fingerID;
}
```

Apéndice 9: Conexión Bluethoot

Archivo .xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tecnomor.apptesis2.Dispos_bt">

    <TextView
        android:id="@+id/idTitulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="16dp"
        android:text="              DISPOSITIVOS VINCULADOS "
        android:textAlignment="center"
        android:textSize="18sp"
        android:textStyle="bold" />

    <ListView
        android:id="@+id/IdLista"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@id/idTitulo" />
</RelativeLayout>
```

Archivo .java

```java
package com.tecnomor.controlbt;

import ...
public class Dispos_bt extends AppCompatActivity {
    //1)
    // Depuración de LOGCAT
    private static final String TAG = "Dispos_bt"; //<-<- PARTE A MODIFICAR >->->
    // Declaracion de ListView
    ListView IdLista;
    // String que se enviara a la actividad principal, mainactivity
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Declaracion de campos
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter mPairedDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dispos_bt);
    }
    @Override
    public void onResume()
    {
        super.onResume();
        //----------------------------------
        VerificarEstadoBT();

        // Inicializa la array que contendra la lista de los dispositivos bluetooth vinculados
        mPairedDevicesArrayAdapter = new ArrayAdapter( context: this, R.layout.nombre_dispositivos);
        // Presenta los disposisitivos vinculados en el ListView
        IdLista = (ListView) findViewById(R.id.IdLista);
        IdLista.setAdapter(mPairedDevicesArrayAdapter);
        IdLista.setOnItemClickListener(mDeviceClickListener);
        // Obtiene el adaptador local Bluetooth adapter
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();

        //------------------- EN CASO DE ERROR -----------------------------------
        //SI OBTIENES UN ERROR EN LA LINEA (BluetoothDevice device : pairedDevices)
        //CAMBIA LA SIGUIENTE LINEA POR
        //Set <BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();
        //------------------------------------------------------------------------

        // Obtiene un conjunto de dispositivos actualmente emparejados y agregua a 'pairedDevices'
        Set <BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

        // Adiciona un dispositivos previo emparejado al array
        if (pairedDevices.size() > 0)
        {
```

```
        {
            for (BluetoothDevice device : pairedDevices) { //EN CASO DE ERROR LEER LA ANTERIOR EXPLICACION
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        }
    }

    // Configura un (on-click) para la lista
    private AdapterView.OnItemClickListener mDeviceClickListener = (av, v, arg2, arg3) -> {

            // Obtener la dirección MAC del dispositivo, que son los últimos 17 caracteres en la vista
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);

            // Realiza un intent para iniciar la siguiente actividad
            // mientras toma un EXTRA_DEVICE_ADDRESS que es la dirección MAC.
            Intent i = new Intent( packageContext: Dispos_bt.this, User_Interface.class);//<-<- PARTE A MODIFICAR >->->
            i.putExtra(EXTRA_DEVICE_ADDRESS, address);
            startActivity(i);
    };

    private void VerificarEstadoBT() {
        // Comprueba que el dispositivo tiene Bluetooth y que está encendido.
        mBtAdapter= BluetoothAdapter.getDefaultAdapter();
        if(mBtAdapter==null) {
            Toast.makeText(getBaseContext(), text: "El dispositivo no soporta Bluetooth", Toast.LENGTH_SHORT).show();
        } else {
            if (mBtAdapter.isEnabled()) {
                Log.d(TAG, msg: "...Bluetooth Activado...");
            } else {
                //Solicita al usuario que active Bluetooth
                Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, requestCode: 1);

            }
        }
    }

}
```

Apéndice 10: Inicio de Sesión

Archivo .xml Inicio de sesión

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.tecnomor.apptesis2.InicioSesion">

    <ImageView
        android:layout_width="200dp"
        android:layout_height="219dp"
        android:layout_gravity="center"
        android:src="@drawable/usuario" />

    <EditText
        android:id="@+id/TV_user"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Usuario"
        android:inputType="text" />

    <EditText
        android:id="@+id/TV_password"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Contraseña"
        android:inputType="numberPassword"

        />

    <Button
        android:id="@+id/IdBotonAcceder"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Acceder" />
    <TextView
        android:id="@+id/ID_tv_registrar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Super usuario" />

</LinearLayout>
```

Archivo .java Inicio de Sesión

```java
package com.tecnomor.apptesis2;

import ...

public class InicioSesion extends AppCompatActivity {
    Button IdBotonAcceder;
    TextView ID_tv_registrar; //declaramos variable para llamar al textview registrar y mandarlo a la actividad registro
    EditText et_user, et_password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_inicio_sesion);
        IdBotonAcceder= (Button) findViewById(R.id.IdBotonAcceder);
        ID_tv_registrar=(TextView) findViewById(R.id.ID_tv_registrar);//llamamos al textview en activity registro
        et_user = (EditText) findViewById(R.id.TV_user);
        et_password= (EditText) findViewById(R.id.TV_password);

        IdBotonAcceder.setOnClickListener((view) -> {

                final String username= et_user.getText().toString();
                final String password= et_password.getText().toString();


                Response.Listener<String> responseListener = (response) -> {
                        try {
                            JSONObject jsonResponse = new JSONObject(response);
                            boolean success = jsonResponse.getBoolean( name: "success");
                            if (success){
                                Intent intentAcceso= new Intent( packageContext: InicioSesion.this, Dispos_bt.class);
                                InicioSesion.this.startActivity(intentAcceso);



                            }else {
                                AlertDialog.Builder builder = new AlertDialog.Builder( context: InicioSesion.this);
                                builder.setMessage("Error login")
                                        .setNegativeButton( text: "Retry", listener: null)
                                        .create().show();
                            }
                        } catch (JSONException e){
                            e.printStackTrace();
                        }
                };

                  LoginRequest loginRequest= new LoginRequest(username, password, responseListener);
                  RequestQueue queue= Volley.newRequestQueue( context: InicioSesion.this);
                  queue.add(loginRequest);


        }); //declaramos el evento al darle click al textview

        ID_tv_registrar.setOnClickListener((view) -> {
                Intent intentRegistro = new Intent( packageContext: InicioSesion.this, SuperUserInicio.class);
                InicioSesion.this.startActivity(intentRegistro);
        }); //declaramos el evento al darle click al textview

    }
}
```

Archivo .xml Super Usuario inicio de sesion

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.tecnomor.apptesis2.SuperUserInicio">


    <ImageView
        android:layout_width="200dp"
        android:layout_height="219dp"
        android:layout_gravity="center"
        android:src="@drawable/usuario" />



    <TextView
        android:id="@+id/ID_tv_registrar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="En esta Opción Solo Puede Acceder el Super Usuario" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Super Usuario"
        android:inputType="text" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Contraseña Super Usuario"
        android:inputType="textPassword"

        />
    <Button
        android:id="@+id/IdBotonAccederSupUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Acceder" />

</LinearLayout>
```

Archivo .java Super Usuario Inicio de sesión

```java
package com.tecnomor.apptesis2;

import ...

public class SuperUserInicio extends AppCompatActivity {
    Button IdBotonAccederSupUsuario;
    EditText et_user, et_password;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_inicio_sesion);
        IdBotonAccederSupUsuario= (Button) findViewById(R.id.IdBotonAcceder);
        et_user = (EditText) findViewById(R.id.TV_user);
        et_password= (EditText) findViewById(R.id.TV_password);

        IdBotonAccederSupUsuario.setOnClickListener((view) -> {

                final String username= et_user.getText().toString();
                final String password= et_password.getText().toString();


                Response.Listener<String> responseListener = onResponse(response) -> {
                        try {
                            JSONObject jsonResponse = new JSONObject(response);
                            boolean success = jsonResponse.getBoolean( name: "success");
                            if (success){
                                Intent intentAcceso= new Intent( packageContext: SuperUserInicio.this, SuperusuarioMenu.class);
                                SuperUserInicio.this.startActivity(intentAcceso);



                            }else {
                                AlertDialog.Builder builder = new AlertDialog.Builder( context: SuperUserInicio.this);
                                builder.setMessage("Error login")
                                        .setNegativeButton( text: "Retry", listener: null)
                                        .create().show();
                            }
                        } catch (JSONException e){
                            e.printStackTrace();
                        }
                };

                LoginRequest loginRequest= new LoginRequest(username, password, responseListener);
                RequestQueue queue= Volley.newRequestQueue( context: SuperUserInicio.this);
                queue.add(loginRequest);


        }); //declaramos el evento al darle click al textview


    }
}
```

Archivo Inicio de sesión Request Java

```java
package com.tecnomor.apptesis2;

import ...

public class LoginRequest extends StringRequest {
    private static final String LOGIN_REQUEST_URL="http://192.168.1.75/Login.php"; //AQUÍ VA LA URL DE NUESTRO ARCHIVO DE REGISTRO
    private Map<String,String> params;
    public LoginRequest(String username,  String password, Response.Listener<String>listener){
        super(Request.Method.POST, LOGIN_REQUEST_URL,listener, null);
        params=new HashMap<>();
        params.put("username",username);
        params.put("password",password);

    }//constructor de la clase

    @Override
    public Map<String, String> getParams() { return params; }
}
```

Apéndice 11: Archivo PHP incio de sesión

```php
<?php
    $con = mysqli_connect("localhost", "root", "", "user");

    $username = $_POST["username"];
    $password = $_POST["password"];

    $statement = mysqli_prepare($con, "SELECT * FROM user WHERE username = ? AND password = ?");
    mysqli_stmt_bind_param($statement, "ss", $username, $password);
    mysqli_stmt_execute($statement);

    mysqli_stmt_store_result($statement);
    mysqli_stmt_bind_result($statement, $userID, $name, $age, $username, $password);

    $response = array();
    $response["success"] = false;

    while(mysqli_stmt_fetch($statement)){
        $response["success"] = true;
        $response["name"] = $name;
        $response["age"] = $age;
        $response["username"] = $username;
        $response["password"] = $password;
    }

    echo json_encode($response);
?>
```

Apéndice 12: Menú super usuario

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tecnomor.apptesis2.SuperusuarioMenu">
    <TextView
        android:id="@+id/IdOpSuUser"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Opciones de Super Usuario"
        android:textAlignment="center"
        android:layout_marginTop="15sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/IdRegistrarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/IdOpSuUser"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="34dp"
        android:text="Registrar Usuario"
        tools:ignore="UnknownId" />

    <Button
        android:id="@+id/IdRegistrarDependiente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/IdRegistrarUsuario"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="21dp"
        android:text="Registrar Dependiente"
        tools:ignore="UnknownId" />
```

```
<Button
    android:id="@+id/IdRegistrarHuellaDependiente"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/IdRegistrarDependiente"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:text="Registrar Huella Dependiente"
    tools:ignore="UnknownId" />

<Button
    android:id="@+id/IdRegistrarHuellaUsuario"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/IdRegistrarHuellaDependiente"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="14dp"
    android:text="Registrar Huella Usuario"
    tools:ignore="UnknownId" />

<Button
    android:id="@+id/IdBorrarUsuario"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/IdBorrarDependiente"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="65dp"
    android:text="Borrar Usuario"
    tools:ignore="UnknownId" />

<Button
    android:id="@+id/IdBorrarDependiente"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/IdRegistrarHuellaUsuario"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="16dp"
    android:text="Borrar Dependiente "
    tools:ignore="UnknownId" />
```

```
    <Button
        android:id="@+id/IdAbrirLocker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_marginBottom="11dp"
        android:text="Abrir Locker"
        tools:ignore="UnknownId" />

</RelativeLayout>
```

Apendicce 13: Menú Dependiente

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tecnomor.apptesis2.SuperusuarioMenu">
    <TextView
        android:id="@+id/IdOpSuUser"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Opciones de Dependiente"
        android:textAlignment="center"
        android:layout_marginTop="15sp"
        android:textStyle="bold" />

    <Button
        android:id="@+id/IdRegistrarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/IdOpSuUser"
        android:layout_alignStart="@+id/IdOpSuUser"
        android:layout_below="@+id/IdBorrarDependiente"
        android:layout_marginTop="59dp"
        android:text="Registrar Usuario"
        tools:ignore="UnknownId" />


    <Button
        android:id="@+id/IdBorrarDependiente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/IdOpSuUser"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="54dp"
        android:text="Borrar Usuario"
        tools:ignore="UnknownId" />

    <Button
        android:id="@+id/IdAbrirLocker"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:layout_below="@+id/IdRegistrarUsuario"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="77dp"
        android:text="Abrir Locker"
        tools:ignore="UnknownId" />

</RelativeLayout>
```

Apéndice 14: Menú usuario

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.tecnomor.apptesis2.User_Interface">
    <TextView
        android:id="@+id/IdControl"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:text="Usuario"
        android:textAlignment="center"
        android:textStyle="bold" />

    <Button
        android:id="@+id/IdEncender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/IdControl"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="26dp"
        android:text="ABRIR PUERTA"
        tools:ignore="UnknownId" />


</RelativeLayout>
```

Apéndice 15: Registro

Archivo .xml Registro

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.tecnomor.apptesis2.Registro">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Registro de Dependiente"
        android:layout_gravity="center"
        android:textSize="25sp"/>

    <EditText
        android:id="@+id/EditT_nombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Nombre"
        android:inputType="text" />

    <EditText
        android:id="@+id/EditT_apellido"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Apellido"
        android:inputType="text" />

    <EditText
        android:id="@+id/EditT_edad"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Edad"
        android:inputType="text" />
```

```
    <EditText
        android:id="@+id/EditT_usuario"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Usuario"
        android:inputType="text" />

    <EditText
        android:id="@+id/EditT_contraseña"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Contraseña"
        android:inputType="textPassword" />

    <EditText
        android:id="@+id/EditT_direccion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Dirección"
        android:inputType="text" />

    <Button
        android:id="@+id/Btn_registrar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="REGISTRAR" />

</LinearLayout>
```

Archivo RegistroRequest.java

```
package com.tecnomor.apptesis2;
//esta clase permite hacer toda la clase de registro

import com.android.volley.Response;
import com.android.volley.toolbox.StringRequest;

import java.util.HashMap;
import java.util.Map;


public class RegisterRequest extends StringRequest {
    private static final String REGISTER_REQUEST_URL="http://192.168.1.75/Register.php"; //AQUÍ VA LA URL DE NUESTRO ARCHIVO DE REGISTRO
    private Map<String,String> params;
    public RegisterRequest(String name, String last_name, String username, int age, String address, String password, Response.Listener<String>listener){
        super(Method.POST, REGISTER_REQUEST_URL,listener, null);
        params=new HashMap<>();
        params.put("name",name);
        params.put("last_name",last_name);
        params.put("username",username);
        params.put("age",age+"");
        params.put("address",address);
        params.put("password",password);

    }//constructor de la clase

    @Override
    public Map<String, String> getParams() { return params; }
}
```

Archivo Registro.java

```
package com.tecnomor.apptesis2;

import ...

public class Registro extends AppCompatActivity implements View.OnClickListener {
    EditText etnombre, etapellido, etedad, etusuario, etcontraseña, etdireccion;
    Button btn_registrar;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro);

        etnombre= findViewById(R.id.EditT_nombre);
        etapellido= findViewById(R.id.EditT_apellido);
        etedad= findViewById(R.id.EditT_edad);
        etusuario= findViewById(R.id.EditT_usuario);
        etcontraseña= findViewById(R.id.EditT_contraseña);
        etdireccion= findViewById(R.id.EditT_direccion);

        btn_registrar=findViewById(R.id.Btn_registrar);

        //evento al dar click en el botton
        btn_registrar.setOnClickListener(this);

    }
    //este es el evento onclick del boton
    @Override
    public void onClick(View v) {
        final String name=etnombre.getText().toString();
        final String last_name=etapellido.getText().toString();
        final int age= Integer.parseInt(etedad.getText().toString());
        final String username=etusuario.getText().toString();
        final String password=etcontraseña.getText().toString();
        final String address=etdireccion.getText().toString();

       Response.Listener<String> respoListener = (response) → {
              try {
                  JSONObject jsonReponse = new JSONObject(response);
                  boolean success= jsonReponse.getBoolean( name: "success");
                  if(success){
                      Intent intent=new Intent( packageContext: Registro.this,InicioSesion.class);
                      Registro.this.startActivity(intent);
                  }else {
                      AlertDialog.Builder builder = new AlertDialog.Builder( context: Registro.this);
                      builder.setMessage("Error registro")
                              .setNegativeButton( text: "Retry", listener: null)
                              .create().show();
                  }
              } catch (JSONException e){

                  e.printStackTrace();
              }

       };

    RegisterRequest registerRequest= new RegisterRequest(name, last_name, username, age, address, password, respoListener);
        RequestQueue queue = Volley.newRequestQueue( context: Registro.this);
        queue.add(registerRequest);
    }
}
```

Apéndice 16: Archivo PHP Registro

```php
<?php
    $con = mysqli_connect("localhost", "root", "", "user");

    $name = $_POST["name"];
    $last_name = $_POST["last_name"];
    $address = $_POST["address"];
    $age = $_POST["age"];
    $username = $_POST["username"];
    $password = $_POST["password"];
    $statement = mysqli_prepare($con, "INSERT INTO user (name, last_name, address. username, age, password) VALUES (?, ?, ?, ?, ?, ?)");
    mysqli_stmt_bind_param($statement, "ssis", $name, $last_name, $address, $username, $age, $password);
    mysqli_stmt_execute($statement);

    $response = array();
    $response["success"] = true;

    echo json_encode($response);
?>
```

Apéndice 17:

Borrar Dependiente

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.tecnomor.apptesis2.BorrarDependiente">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Escribe el nombre del dependiente que deseas borrar" />

    <EditText

        android:id="@+id/IdUsuarioEliminar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Dependiente"
        android:inputType=""

        />

    <Button
        android:id="@+id/IdBotonBorrarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Borrar Dependiente" />

</LinearLayout>
```

Borrar Usuario

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="com.tecnomor.apptesis2.BorrarUsuario">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Escribe el nombre del Usuario que deseas borrar" />

    <EditText

        android:id="@+id/IdUsuarioEliminar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Usuario"
        android:inputType=""

        />

    <Button
        android:id="@+id/IdBotonBorrarUsuario"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Borrar Usuario" />

</LinearLayout>
```

Apéndice 18: Código implamentación de hardware para Arduino

```
#include <Adafruit_Fingerprint.h>
#include <SoftwareSerial.h>
#define mySerial Serial1 //seleccionamos el serial dellector de huellas

Adafruit_Fingerprint finger = Adafruit_Fingerprint(&mySerial);
SoftwareSerial ModBluetooth(10, 11);//serial para rx y tx de bt

#define RELAY_ON 0
#define RELAY_OFF 1

uint8_t id;
void setup() {
  ////////////////////////////////inicializacion FINGERPRINT
  Serial.begin(9600);
  while (!Serial);  // For Yun/Leo/Micro/Zero/...
  delay(100);
  Serial.println("\n\nAdafruit finger detect test");

  // set the data rate for the sensor serial port
  finger.begin(57600);

  if (finger.verifyPassword()) {
    Serial.println("Found fingerprint sensor!");
  } else {
    Serial.println("Did not find fingerprint sensor :(");
    while (1) { delay(1); }
  }


////////////////////////////////FIN FINGERPRINT

////////////////////////////////inicializacion BT
   ModBluetooth.begin(9600);
    Serial.begin(9600);
    ModBluetooth.println("MODULO CONECTADO");
    ModBluetooth.print("#");
////////////////////////////////Fin BT
  ///////RELAY
    //Asegurar nivel ALTO en cada entrada de rele
  digitalWrite (13, RELAY_ON);
  //Definir los pines como salida PARA EL RELAY
  pinMode (13, OUTPUT);
  ////////////////////TERMINA RELAY
}
```

```
uint8_t readnumber(void) {
  uint8_t num = 0;

  while (num == 0) {
    while (! Serial.available());
    num = Serial.parseInt();
  }
  return num;
}


void loop()
{
  if (ModBluetooth.available())
    {
        Serial.print("BT conectado");
        char VarChar;

        VarChar = ModBluetooth.read();

       if(VarChar == '3') //prender led o abrir puerta
        {
        digitalWrite(13, HIGH);
        delay(5000);
        ModBluetooth.print("LED ENCENDIDO\n");
        Serial.print("LED ENCENDIDO\n");
        delay(2000);
        ModBluetooth.print("DONE\n");
        ModBluetooth.print("#");
        digitalWrite(13, LOW);
        }
        if(VarChar == '4')
        {
        digitalWrite(13, LOW);
        ModBluetooth.print("LED APAGADO\n");
        Serial.print("LED APAGADO\n");
        ModBluetooth.print("#");
        }
        if(VarChar == '2')
        {
          Serial.println("Ready to enroll a fingerprint!\n");
          Serial.println("Please type in the ID # (from 1 to 127) you want to save this finger as...\n");
          id = readnumber();
          if (id == 0) {// ID #0 not allowed, try again!
             return;
          }
          Serial.print("Enrolling ID #\n");
```

```
          Serial.println(id);

          while (!  getFingerprintEnroll() );
        }


    }
  else
  {//////LEE SI HAY HUELLA
        finger.getTemplateCount();
        Serial.print("Sensor contains "); Serial.print(finger.templateCount); Serial.println(" templates");
        Serial.println("Waiting for valid finger...");
        Serial.print("Leyendo huella");
        getFingerprintIDez();
        delay(50);
      /////LEE LA ENTRADA DE DATOS
  }
}



////////////////////////////////código registrar huella
//////////////////////////////
uint8_t getFingerprintEnroll() {

  int p = -1;
  Serial.print("Waiting for valid finger to enroll as #"); Serial.println(id);
  while (p != FINGERPRINT_OK) {
    p = finger.getImage();
    switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println(".");
      break;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      break;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      break;
    default:
      Serial.println("Unknown error");
      break;
    }
  }
```

```
p = finger.image2Tz(1);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

Serial.println("Remove finger");
delay(2000);
p = 0;
while (p != FINGERPRINT_NOFINGER) {
  p = finger.getImage();
}
Serial.print("ID "); Serial.println(id);
p = -1;
Serial.println("Place same finger again");
while (p != FINGERPRINT_OK) {
  p = finger.getImage();
  switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image taken");
    break;
  case FINGERPRINT_NOFINGER:
    Serial.print(".");
    break;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    break;
  case FINGERPRINT_IMAGEFAIL:
    Serial.println("Imaging error");
    break;
```

```
  default:
    Serial.println("Unknown error");
    break;
  }
}

// OK success!

p = finger.image2Tz(2);
switch (p) {
  case FINGERPRINT_OK:
    Serial.println("Image converted");
    break;
  case FINGERPRINT_IMAGEMESS:
    Serial.println("Image too messy");
    return p;
  case FINGERPRINT_PACKETRECIEVEERR:
    Serial.println("Communication error");
    return p;
  case FINGERPRINT_FEATUREFAIL:
    Serial.println("Could not find fingerprint features");
    return p;
  case FINGERPRINT_INVALIDIMAGE:
    Serial.println("Could not find fingerprint features");
    return p;
  default:
    Serial.println("Unknown error");
    return p;
}

// OK converted!
Serial.print("Creating model for #");  Serial.println(id);

p = finger.createModel();
if (p == FINGERPRINT_OK) {
  Serial.println("Prints matched!");
} else if (p == FINGERPRINT_PACKETRECIEVEERR) {
  Serial.println("Communication error");
  return p;
} else if (p == FINGERPRINT_ENROLLMISMATCH) {
  Serial.println("Fingerprints did not match");
  return p;
} else {
  Serial.println("Unknown error");
  return p;
}
```

```
  Serial.print("ID "); Serial.println(id);
  p = finger.storeModel(id);
  if (p == FINGERPRINT_OK) {
    Serial.println("Stored!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_BADLOCATION) {
    Serial.println("Could not store in that location");
    return p;
  } else if (p == FINGERPRINT_FLASHERR) {
    Serial.println("Error writing to flash");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }
}
////////////////////////////////código leer huella
////////////////////////////
uint8_t getFingerprintID() {
  uint8_t p = finger.getImage();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image taken");
      break;
    case FINGERPRINT_NOFINGER:
      Serial.println("No finger detected");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_IMAGEFAIL:
      Serial.println("Imaging error");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK success!

  p = finger.image2Tz();
  switch (p) {
    case FINGERPRINT_OK:
      Serial.println("Image converted");
      break;
```

```
    case FINGERPRINT_IMAGEMESS:
      Serial.println("Image too messy");
      return p;
    case FINGERPRINT_PACKETRECIEVEERR:
      Serial.println("Communication error");
      return p;
    case FINGERPRINT_FEATUREFAIL:
      Serial.println("Could not find fingerprint features");
      return p;
    case FINGERPRINT_INVALIDIMAGE:
      Serial.println("Could not find fingerprint features");
      return p;
    default:
      Serial.println("Unknown error");
      return p;
  }

  // OK converted!
  p = finger.fingerFastSearch();
  if (p == FINGERPRINT_OK) {
    Serial.println("Found a print match!");
  } else if (p == FINGERPRINT_PACKETRECIEVEERR) {
    Serial.println("Communication error");
    return p;
  } else if (p == FINGERPRINT_NOTFOUND) {
    Serial.println("Did not find a match");
    return p;
  } else {
    Serial.println("Unknown error");
    return p;
  }

  // found a match!
  Serial.print("Found ID ###"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);

  return finger.fingerID;

  //////////ACTIVA RELE
    digitalWrite (13, RELAY_ON);     //Activa relé 1
    Serial.print(" PUERTA ABIERTA ");
    delay (5000);
    digitalWrite (13, RELAY_OFF);
  //////////TERMINA DE ACTIVAR RELE


}
```

```
// returns -1 if failed, otherwise returns ID #
int getFingerprintIDez() {
  uint8_t p = finger.getImage();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.image2Tz();
  if (p != FINGERPRINT_OK)  return -1;

  p = finger.fingerFastSearch();
  if (p != FINGERPRINT_OK)  return -1;

  // found a match!
  Serial.print("Found ID #"); Serial.print(finger.fingerID);
  Serial.print(" with confidence of "); Serial.println(finger.confidence);


    //////////ACTIVA RELE
    digitalWrite (13, RELAY_OFF);     //Activa relé 1
    Serial.print(" PUERTA ABIERTA ");
    delay (5000);
    digitalWrite (13, RELAY_ON);
  //////////TERMINA DE ACTIVAR RELE
  return finger.fingerID;


}
```

El códgio que mostrado realiza las siguientes acciones:

-Realiza la conexión con el módulo HC-05 para la recepción y envió de información empleando este medio, para despues aprovechar esta información y permitir la activación del relevador conectado.

-Registrar una huella dactilar en caso de que se seleccione la opción (en este caso se selecciona desde la aplicación móvil)

-Detectar una huella dactilar al momento que sea colocada en el sensor biométrico y buscar si se encuentra registrada, en caso afirmativo activará el relevador.

Bibliografía;

Tema 2

LINKS DE REFERENCIA:

https://drive.google.com/file/d/10hHjMEH--irvJhjHDUaHXEF_m83IO2Q6/view
https://drive.google.com/file/d/1sHyPWQ1OpKY-k2Vwb5i6sTB4gbJYE7rG/view
https://drive.google.com/file/d/1r2FyauQiFxf7NAz63GoVGH2K09qP9hgk/view
https://drive.google.com/file/d/1z9e2A_ic8ZFt5rff87ZrJzisgc08KP4G/view
https://drive.google.com/file/d/1uQVwVA8uBP-zQGKnjimK1H8XcAHJ2jLP/view
https://drive.google.com/file/d/1ODSIWpR2hRVESx76Gr4eFlZwjuX8LNK3/view
https://drive.google.com/file/d/1tmLJYZ0MjYbuGhqSdXwK2CzD_k2y_VgV/view
https://www.youtube.com/watch?v=ew56YHMtZw8&t=309s
https://www.bestbuy.com.mx/c/buscar-best-buy/buscar?query=cerradura+inteligente&autoFacet=true

Tema 3

CAIRÓ, Osvaldo
Metodología de la Programación Algoritmos, Diagramas de Flujo y Programas 2a. edición
México Alfaomega, 2003 Tomo I

SOLÓRZANO, J. Fernando
Introducción a la Programación estructurada y al lenguaje C
México Facultad de Ingeniería - UNAM, 2003 Tomo II

SOMMERVILLE, Ian
Software Engineering 7th Edition Reading,
Massachussets, U.S.A Addison Wesley, 2004

LINKS DE REFERENCIA:

https://www.omnicell.com/mts/Products_and_Solutions_For_Pharmacy/Automated_Dispensing_Cabinets.aspx
https://www.pppmag.com/documents/V3N6/BuyersGuideP24.pdf
https://www.youtube.com/watch?v=ZZmUuRG87sU
https://www.bizjournals.com/bizjournals/how-to/technology/2015/04/the-internet-of-things-is-transforming-vending.html
http://www.coincitymexico.com/coincity/proyectos
http://www.made-in-china.com/products-search/hot-china-products/Vending_Machine.html?gclid=CjwKCAiAy4bTBRAvEiwAFtatHMB22oPtaVxz1UR-wAHTdXCG0DDXA3P-KVvRLeJP1HsceWtsGYwfshoC1HAQAvD_BwE
https://naylampmechatronics.com/blog/35_Tutorial--LCD-con-I2C-controla-un-LCD-con-so.html
https://www.prometec.net/bus-i2c/

http://playground.arduino.cc/Code/LCDi2c
https://www.luisllamas.es/arduino-teclado-matricial
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html
https://www.carrod.mx/products/modulo-bluetooth-zs-040-hc-05
https://www.youtube.com/watch?v=vraj-c1jSfc
https://www.prometec.net/bt-hc05/
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html

Tema 4

ARELLANO M., Lucila P. y Hernandez Hdez. Luciralia
Manual de prácticas de la asignatura de Bases de Datos
UNAM, Fac. De Ingeniería., DIE

DE MIGUEL MARTÍNEZ, Adoración, PIATTINI , Mario, ESPERANZA, Marcos
Diseño de bases de datos relacionales
México Alfaomega, 2000

TANENBAUM, Andrew S.
Redes de Computadoras 4a. edición
México Pearson Educación, 2003

COMER, Douglas E.
Interconectividad de Redes con TCP/IP Diseño e Implementación Vol II 3a. edición
México Prentice Hall, 2000

LINKS DE REFERENCIA:

https://www.youtube.com/watch?v=dDUd-V425OU
http://www.innovadomotics.com/mn-tuto/mn-android/proyectos/23-and-cnm-bt.html?start=5
http://blog.antoniohorrillo.com/iotard01/
https://aprendiendoarduino.wordpress.com/tag/hc-05/
http://cursoarduinomega.blogspot.com/2015/10/conexion-bluetooth-android-con-arduino.html
https://naylampmechatronics.com/blog/24_configuracion-del-modulo-bluetooth-hc-05-usa.html
https://www.youtube.com/watch?v=sRD-tsACiVs&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h
https://www.youtube.com/watch?v=I4CUje6bfjM&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h&index=2
https://www.youtube.com/watch?v=QxItUFFEJFo&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h&index=3
https://www.youtube.com/watch?v=PbSMPgL5_KY&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h&index=4
https://www.youtube.com/watch?v=-usS3_-

zWVg&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h&index=5
https://www.youtube.com/watch?v=LyL6wWCST9Y&list=PLQsKNPJaFPHosQtcBkdHcwj2FVF7fTJ1h&index=6