Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Desarrollo de una aplicación de identificación y control de acceso para dispositivos móviles

Autor: Danel Castro García

Tutor(es): Ing. Daniel Meriño Tamayo

La Habana, 16 de noviembre del 2023

“Año 64 de la Revolución”

DECLARACIÓN DE AUTORÍA

El autor del trabajo de diploma con título “Desarrollo de una aplicación de identificación y control de acceso para dispositivos móviles.”, concede a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la investigación, con carácter exclusivo. De forma similar se declara como único autor de su contenido. Para que así conste firma(n) la presente a los 16 días del mes de noviembre del año 2023.

Danel Castro García

_____

Firma del Autor

Ing. Daniel Meriño Tamayo

_____

Firma del Tutor

DATOS DE CONTACTO

<Curriculum e información de contacto del tutor: nombre y apellidos, títulos académicos, formación de postgrado recibida, lugar de trabajo, responsabilidades laborales asumidas, experiencia profesional, líneas de trabajo y/o investigación, correo electrónico, perfiles en redes profesionales>

<Curriculum e información de contacto del asesor: nombre y apellidos, títulos académicos, formación de postgrado recibida, lugar de trabajo, responsabilidades laborales asumidas, experiencia profesional, líneas de trabajo y/o investigación, correo electrónico, perfiles en redes profesionales>

<Curriculum e información de contacto del consultante: nombre y apellidos, títulos académicos, formación de postgrado recibida, lugar de trabajo, responsabilidades laborales asumidas, experiencia profesional, líneas de trabajo y/o investigación, correo electrónico, perfiles en redes profesionales>

AGRADECIMIENTOS

Agradecer a todas mis amistades que estuvimos los 5 años de la carrera, juntos, en todo momento, a mi tutor por ayudarme en el proceso de culminación de estudios y a mi familia por apoyarme en todo.

DEDICATORIA

Esta tesis se la dedico a mis queridos padres, por inculcarme una buena educación y valores desde niño, por estar presentes cuando siempre los necesité y por hacerme la persona que soy en estos momentos

Resumen

En la era digital actual, los dispositivos móviles desempeñan un papel crucial en nuestras vidas diarias, ofreciendo una amplia gama de funcionalidades y servicios. La autenticación y el control de acceso son aspectos vitales en diversos contextos. La falta de una autenticación adecuada puede tener consecuencias graves, como el robo de identidad, el acceso no autorizado a información confidencial y la violación de la privacidad.

El objetivo principal de este proyecto es desarrollar una solución segura para verificar la identidad de credencial, esta aplicación permitirá identificar a las personas, gestionar el acceso y registrar las entradas y salidas. Además, se buscará detectar posibles violaciones los usuarios y controlar su acceso a la entidad XETiD. Mediante el uso de un código QR/barras o un número de seguridad y la entrada de personas no autorizadas a la institución.

Palabras Claves: acceso autorizado, autenticación, control de acceso, seguridad

ABSTRACT

In today's digital age, mobile devices play a crucial role in our daily lives, offering a wide range of functionalities and services. Authentication and access control are vital aspects in various contexts. Failure to have proper authentication can lead to serious consequences such as identity theft, unauthorized access to sensitive information, and breach of privacy.

The main objective of this project is to develop an efficient and secure solution to verify credential identity, this application will allow people to be identified, manage access and record entries and exits. In addition, it will detect possible violations by users and control their access to the XETiD search entity. Through the use of a QR/bar code or security number and the entry of unauthorized persons into the institution.

Keywords: authorized access, authentication, access control, security

Índice

Tabla de contenido

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

AVAL DEL CLIENTE

xetid

Habana, 1 de noviembre de 2023

Estimado Danel Castro García.

Nos complace informarle que su solicitud de aceptación de software ha sido aprobada. En nombre de XETID nos complace aceptar su software Aplicación de identificación y control de acceso para dispositivos móviles para su uso en nuestra organización, expresamos nuestra satisfacción con respecto al software que se ha implementado recientemente en nuestra empresa.

Atentamente, Daniel Meriño Tamayo

Cargo: Jefe de Proyecto

Firma:

XETID

Calle 206A, e/ 217 y 203,
municipio Boyeros, La Habana
(+537) 837 3660 / (+537) 837 36n
xetid@xetid.cu
www.xetid.cu

OPINIÓN DEL(OS) TUTOR(ES)

<Contenido de la opinión de los tutores>

INTRODUCCIÓN

En la actualidad, el desarrollo tecnológico ha generado una transformación social significativa, impulsando la sociedad de la información. Es importante destacar que la información se ha convertido en un recurso clave en esta nueva sociedad, dando lugar a la aparición de nuevas profesiones y adaptando las existentes. Las tecnologías de la información y la comunicación (TIC) desempeñan un papel fundamental en esta dimensión social, influyendo en diversos ámbitos y dando forma a nuevas estructuras sociales. Existe una interacción constante y bidireccional entre la tecnología y la sociedad. A medida que aumenta el uso de las TIC, también crece la necesidad de implementar mecanismos de seguridad de la información de manera efectiva en las organizaciones. Es fundamental contar con dispositivos de autenticación que faciliten el control del acceso lógico de los usuarios en los sistemas informáticos. Es importante destacar que estos mecanismos y dispositivos de seguridad se implementan con el objetivo de proteger los activos, cumpliendo con las regulaciones y leyes vigentes. Un control de accesos es un sistema electrónico que restringe o permite el acceso de un usuario o grupo de usuarios a un área específica validando la identificación por medio de diferentes tipos de lectura (clave por teclado, lector de tarjetas, biometría, etc,) y a su vez controlando el recurso (puerta, armario, etc,) por medio de un dispositivo eléctrico como un electroimán, pestillo o motor.

Un control de accesos requiere flexibilidad para que no haya limitaciones en la movilidad por cambios que se producen en los permisos. Necesita precisión para que se le asigne el permiso correcto a cada persona. Y también es necesario que tenga suficiente capacidad para almacenamiento y registro de un mínimo de datos (Whitman, 2018). Hasta hace poco, el acceso a áreas restringidas se realizaba mediante métodos mecánicos, como cerrojos y llaves. Sin embargo, en la actualidad, estos métodos están siendo reemplazados por sistemas electrónicos más seguros y avanzados. El control de acceso tiene como objetivo principal regular la entrada y salida de personas en áreas protegidas. Este sistema proporciona información sobre quién entra, cuándo y hacia dónde se dirige cada individuo.

Existen diferentes tipos de sistemas de control de acceso. Por un lado, están los modelos estándar que se instalan en el pomo exterior de la puerta. Por otro lado, existen modelos duales que permiten la lectura desde ambos lados de la puerta. Estos modelos son ideales para zonas donde se requiere acceso autorizado desde ambos lados, como pasillos o áreas de paso, estos sistemas de control de acceso se implementan con el fin de garantizar la seguridad y protección de las áreas restringidas. Su objetivo es evitar fraudes y asegurar que solo las personas autorizadas tengan acceso a dichas zonas. (Whitman, 2018).

En los últimos años la informática y las telecomunicaciones han dado un salto vertiginoso y con ello han llevado a la sociedad lo que se conoce como “Era de la Información”. La aplicación de esta ciencia en el campo de la seguridad ha revolucionado, a su vez, la forma de llevar el registro de las personas que acceden a una entidad, gracias a estos avances actualmente existen sistemas informáticos que automatizan este proceso. En paralelo se han desarrollado diversos mecanismos de autenticación tales como: tarjetas inteligentes, sistemas basados en biometría, los cuales incluyen, huellas digitales, voz, rostro, iris, escritura a mano y otros métodos automatizados utilizados para reconocer individuos.

Para toda organización donde existe una gran aglomeración de personas y se maneja información sensible es fundamental asegurar el control de los accesos a sus instalaciones, con el objetivo de mantener un control estricto de la entrada y salida del personal autorizado. Esto en ocasiones se convierte en una tarea sumamente complicada por lo que se hace necesaria la utilización de Sistemas de Control de Acceso (SCA) para limitar el acceso a lugares restringidos y llevar el registro de los mismos.

Además, en no pocas ocasiones, se ven limitados los recursos, y escasean los dispositivos necesarios para la identificación y reconocimiento del personal por cualquiera de los métodos existentes. Actualmente, los terminales destinados para identificación, pueden encarecerse en dependencia de la tecnología y sensores que incorporen para su cometido, y teniendo en cuenta las condiciones económicas del

país, la importación de estos equipos puede resultar bastante costoso en tiempo y presupuesto.

Es por ello que se propone el desarrollo de una aplicación de identificación y control de acceso para dispositivos móviles mediante lecturas de diferentes métodos, permitiendo o denegando el acceso a la entidad.

Luego de analizar la problemática anterior, se plantea la siguiente interrogante como **Problema de investigación:** ¿Cómo usar un dispositivo móvil como terminal de identificación, que se integre con el Sistema de Control de Acceso XETiD?

Para dar solución al problema antes expuesto se formula el siguiente **objetivo general:** Desarrollar una aplicación para dispositivos móviles, que permita la identificación de personas por diferentes métodos, y se integre al Sistema de Control de Acceso XETiD.

Para dar cumplimiento al objetivo general, se definen como

objetivos específicos:

1. Elaborar el marco teórico de la investigación
2. Realizar análisis y diseño de software según los requerimientos identificados.
3. Implementar el software para dispositivos móviles.
4. Realizar las pruebas que permitan evaluar y ensayar las funcionalidades del software.

Teniendo como **objeto de estudio:** aplicación de identificación y control de acceso para dispositivos terminales de identificación y control de acceso.

Enmarcado en el **campo de acción:** Aplicaciones de identificación y control de acceso para dispositivos móviles.

Con el propósito de cumplir con el objetivo general, se plantean las siguientes **tareas de la investigación:**

1. Describir el estado del arte de las aplicaciones de identificación y control de acceso para dispositivos móviles.

2. Definición y estudio de metodología, tecnología y herramientas a emplear para el desarrollo del componente.

3. Elaboración de los artefactos correspondientes al negocio mediante la modelación por procesos.

4. Identificar las funcionalidades principales en términos de requisitos a partir del estudio de software similares y la consulta a especialistas.

5. Validación de los requisitos funcionales identificados.

Métodos científicos de investigación.

En esta investigación se emplearon varios métodos teóricos para el estudio del avance de los controladores de acceso automatizados, con el objetivo de respaldar y facilitar el proceso de modelado. Se utilizó el método histórico-lógico para comprender la evolución de estos controladores y realizar procedimientos que ayuden a su comprensión, así como para deducir situaciones que puedan complicar el problema, elementos cruciales para obtener una solución efectiva. También se aplicó el método empírico cuantitativo a través de la observación, con el propósito de recopilar datos a medida que se desarrolla el software. A continuación, se presenta la estructura del presente trabajo de diploma:

Capítulo 1

En este capítulo se exploran los aspectos generales del estudio del estado del arte en el control de acceso de entidades. Se explicará el proceso de ejecución del control de acceso y se identificarán los principales desafíos que enfrentan las entidades en la actualidad. Con base en estos desafíos, se buscará una solución óptima para garantizar su cumplimiento. Se realizará una descripción y selección de las herramientas, tecnologías y metodología que se utilizarán en el desarrollo de la aplicación.

Capítulo 2

En este capítulo se proporciona una descripción general de los procesos involucrados en el campo de acción del control de acceso. Se presenta una visión general de la propuesta de solución y cómo funciona. Se detalla la solución propuesta, se especifican los requisitos funcionales y no funcionales necesarios, se identifican los actores del sistema y se presentan los diagramas de caso de uso utilizados.

Capítulo 3

En este capítulo se representan los elementos físicos requeridos para implementar la aplicación, utilizando un diagrama de despliegue. Se presentan los aspectos fundamentales de la fase de construcción, que incluyen los procesos de implementación y prueba de software. Se definen los tipos de pruebas que se realizarán al software y se muestran los resultados obtenido

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se presentan los fundamentos teóricos que respaldan la investigación para el desarrollo de la aplicación de control de acceso destinada a la entidad XETiD. Además, se ofrece una breve descripción de las diversas herramientas, lenguajes y tecnologías que se emplearán en su desarrollo, así como la metodología de desarrollo de software que se utilizará.

1.1. Conceptos fundamentales

El control de acceso se refiere al proceso de otorgar permisos a usuarios o grupos para acceder a objetos como archivos o impresoras en una red. Este proceso se basa en tres conceptos fundamentales: identificación, autenticación y autorización. El control de acceso se centra en proteger los objetos de valor y en las decisiones tomadas por las personas para determinar quién tiene acceso a ellos. Este concepto se aplica tanto al control de acceso a espacios físicos como al acceso a información dentro de un sistema., (Stutzman, 2017)

El propósito de estos sistemas, es controlar y gestionar el acceso a un determinado espacio físico. Estos sistemas se utilizan para garantizar la seguridad y proteger los activos y las personas que se encuentran en esos espacios. Los sistemas de control en función de su grado de automatización se clasifican en:

Controles manuales: estos sistemas se basan en que el personal encargado, como vigilantes, guardias de seguridad, personal administrativo o recepcionistas, otorga o niega el permiso de acceso. Para que este sistema sea efectivo, se requiere un esfuerzo y una planificación significativos por parte de las personas responsables. Por lo tanto, es importante que el personal esté familiarizado con todas las personas autorizadas para acceder al lugar. Sin embargo, este tipo de sistema presenta limitaciones cuando el grupo autorizado es muy amplio o cuando hay cambios frecuentes en el personal. (Martínez-Ortega, 2016).

Controles semimanuales: este enfoque combina el uso de equipos o dispositivos electromecánicos para respaldar al personal en la evaluación de las solicitudes de

acceso y la toma de decisiones para permitir o denegar la entrada. Entre los dispositivos más comunes se encuentran las botoneras digitales. Estos dispositivos permiten una mayor eficiencia y precisión en el proceso de control de acceso, al tiempo que brindan un apoyo adicional al personal encargado de tomar decisiones. (Toledano, 2019).

Controles automáticos: se refieren a aquellos sistemas en los cuales las etapas de verificación y acceso son llevadas a cabo completamente por equipos o sistemas electrónicos. Estos sistemas están programados para tomar decisiones de forma automatizada cuando se requiere. Este enfoque elimina la necesidad de intervención humana en el proceso de control de acceso, lo que puede resultar en una mayor eficiencia y precisión en la gestión de la seguridad. (Whitman, 2018)

Los controles desempeñan un papel crucial en la mitigación del riesgo y la reducción de la posibilidad de pérdidas. Para lograr una defensa en profundidad, es necesario implementar una combinación de controles. Una forma de clasificar los controles de acceso es según la forma en que se implementan. Existen tres tipos diferentes de implementaciones: administrativos, físicos y técnicos/lógicos. Estos tipos de controles abarcan diferentes aspectos de la seguridad y trabajan en conjunto para garantizar un entorno protegido. (Harris, 2016)

1.1.1. Tipos de Controles

Controles administrativos

Los controles administrativos desempeñan un papel importante en la gestión de las amenazas internas, como el robo de información privilegiada o las violaciones de bases de datos. Los controles administrativos son fundamentales para establecer una cultura de seguridad sólida dentro de la organización y garantizar que se tomen las medidas adecuadas para prevenir incidentes de seguridad. (Harris, 2016)

Controles físicos

Los controles físicos desempeñan un papel crucial en la prevención y disuasión de eventos desastrosos dentro de un entorno físico. Estos controles incluyen medidas como la presencia de guardias de seguridad, la instalación de cámaras de seguridad, el aseguramiento de salas de servidores y el bloqueo de los ordenadores portátiles. Estas medidas físicas ayudan a proteger los activos y recursos de la organización al controlar el acceso físico a áreas restringidas y garantizar la integridad de los equipos y dispositivos. Los controles físicos son una parte integral de la seguridad global de una organización y complementan otros controles para garantizar una protección efectiva. (Harris, 2016)

Controles técnicos o lógicos

Los controles técnicos o lógicos desempeñan un papel clave en restringir el acceso a los sistemas de información y proteger la información que contienen. Estos controles incluyen medidas como el cifrado de datos, el uso de tarjetas inteligentes para autenticación, listas de control de acceso y protocolos de transmisión seguros. El cifrado de datos garantiza que la información esté protegida mientras se almacena o se transmite, evitando que personas no autorizadas puedan acceder a ella. Las tarjetas inteligentes proporcionan una forma segura de autenticación, asegurando que solo las personas autorizadas puedan acceder a los sistemas. Las listas de control de acceso establecen los permisos y restricciones de acceso a diferentes recursos y datos, garantizando que solo las personas adecuadas tengan acceso a la información relevante. Los protocolos de transmisión seguros, como HTTPS, protegen la integridad

y confidencialidad de la información durante la transmisión. Estos controles técnicos o lógicos son esenciales para proteger los sistemas de información y la información confidencial de una organización. (Harris, 2016)

1.1.2. Conceptos asociados al dominio del problema

Para crear un sistema de control de acceso físico confiable y seguro, es imprescindible que los equipos de desarrollo comprendan a fondo los requisitos y el entorno en el que se implementará. El conocimiento detallado de los conceptos clave, los flujos de información y la filosofía de trabajo es esencial para garantizar la efectividad y la robustez.

1. Identificación y autenticación: Implica el uso de métodos como tarjetas de acceso, sistemas de reconocimiento biométrico (como huellas dactilares o reconocimiento facial) o contraseñas para verificar la identidad de las personas y permitirles acceder a áreas restringidas.
2. Barreras físicas: Incluyen puertas con cerraduras electrónicas, torniquetes, portones o cualquier otro mecanismo físico que limite o controle el acceso a áreas específicas.
3. Vigilancia y monitoreo: Consiste en el uso de cámaras de seguridad, sistemas de detección de intrusos y otros dispositivos de vigilancia para supervisar y registrar actividades en tiempo real, con el fin de detectar cualquier intento de acceso no autorizado.
4. Control de visitantes: Se refiere a los procedimientos y políticas establecidos para gestionar y controlar el acceso de visitantes a una instalación, como el registro de visitantes, la emisión de tarjetas temporales de acceso y la supervisión de su actividad mientras estén presentes.
5. Registro de acceso: Implica la recopilación y el almacenamiento de información sobre quién accede a áreas restringidas, cuándo lo hace y por cuánto tiempo. Esto permite llevar un registro preciso de los movimientos y puede resultar útil para investigaciones posteriores en caso de incidentes.

. Al comprender y aplicar estos conceptos de manera efectiva, los equipos de desarrollo pueden diseñar soluciones sólidas que aseguren un control de acceso físico confiable y prevengan cualquier intento de acceso no autorizado.

1.1.3. Seguridad informática

La seguridad física se refiere a las medidas y controles implementados para proteger los activos físicos y las personas dentro de una organización. El objetivo principal es garantizar la integridad, confidencialidad y disponibilidad de los recursos físicos y prevenir el acceso no autorizado a las instalaciones o áreas restringidas. (Mattord, 2020).

1.1.4. Características de la seguridad informática

➢ **Integridad:** Los activos físicos deben mantenerse en su estado original y protegerse contra cualquier tipo de alteración no autorizada.

➢ **Confidencialidad:** La información y los activos en las instalaciones deben protegerse contra el acceso no autorizado por personas no autorizadas.

➢ **Disponibilidad:** Los recursos físicos necesarios deben estar disponibles para las personas autorizadas en el momento requerido.

➢ **Irrefutable:** Las acciones que ocurren dentro de las instalaciones y el acceso a los activos físicos deben poder ser rastreados y no negados por las personas responsables.

A partir de estas definiciones se puede concluir que la seguridad informática es el estado de cualquier tipo de información que indica que está libre de peligro, daño o riesgo. Implica la preservación de la: integridad, confidencialidad, disponibilidad, así como la condición de ser irrefutable y se logra implementando un conjunto adecuado de controles que abarcan políticas, prácticas, procedimientos, estructuras organizacionales y funciones de software. (Whitman, "Management of Information Security", 2020)

Por lo antes planteado se puede decir que es esencial tener en cuenta los pilares fundamentales de la seguridad Informática conceptualmente expuestos, tanto en la problemática que se pretende solucionar como en el desarrollo de un software, especialmente en un sistema para el control de acceso.

1.2. Modelo de control de acceso

Según el tipo de política de autorización, los modelos de control de acceso pueden ser divididos en: acceso discrecional (DAC), acceso mandatorio (MAC) y acceso basado en roles.

- Control de acceso discrecional (DAC):

En este modelo, los propietarios o administradores de los recursos físicos tienen el poder de decidir quién tiene acceso a estos recursos y cómo se otorga dicho acceso.

- Control de acceso mandatorio (MAC):

Este modelo utiliza políticas de seguridad predefinidas y niveles de clasificación para determinar el acceso a los recursos físicos. Las decisiones de acceso se basan en la autoridad y la necesidad de conocer.

- Control de acceso basado en roles (RBAC):

Este modelo asigna roles a diferentes personas dentro de una organización y les otorga permisos de acceso basados en ese rol, el rol de hecho, es asociado con un conjunto de opciones de permisos en particular. Cuando los usuarios cambian, los roles solo necesitan ser retirados y reasignados. (Ferraiolo, 2001).

Del mismo modo, el modelo de control de acceso basado en roles, es considerado como una forma natural de controlar el acceso a los recursos en las organizaciones y empresas. La motivación detrás de modelo de control de acceso basado en roles parte de considerar que la responsabilidad de un sujeto es más importante que el sujeto en sí. En el modelo, un sujeto puede tener más de un rol o ser miembro de varios grupos. Finalmente, el modelo de control de acceso basado en roles tiene muchas ventajas en comparación con los otros modelos, sin embargo, tiene sus propias dificultades cuando se despliega en el mundo real. En primer lugar, la selección de los roles correctos que representan a un sistema no es una tarea fácil, y la división de los sujetos en

categorías basadas en los roles podría empeorar las cosas. Los roles en el modelo, clasifican a los sujetos en una serie de categorías; así, cada sujeto tiene que tener un rol con el fin de acceder al sistema. A pesar de eso, los roles pueden dar a un sujeto más derechos que los que necesita necesariamente tener, con la posibilidad de tener otro rol que podría conducir a la violación de una política de acceso (Sandhu, 1996).

1.2.1. Consideraciones para un sistema de controles de acceso

Un sistema de control de acceso debe ser cuidadosamente planificado de acuerdo a los requisitos de seguridad del espacio restringido y consideraciones prácticas. Para lograrlo, es importante tener en cuenta siete variables clave en el diseño:

- Tiempo de Ingreso: Es el tiempo que una persona tarda en atravesar todo el sistema de seguridad para ingresar al establecimiento. Este tiempo depende de la respuesta de los dispositivos del sistema.
- Aislamiento: Se refiere al lugar donde se instalará el sistema de control de acceso, asegurando que sea el punto más vulnerable del perímetro defensivo.
- Efectividad del Sistema: Se evalúa mediante variables como el tiempo medio entre fallas, tasas de falsas aceptaciones y rechazos, y las acciones a tomar en caso de fallos.
- Método de Cuarentena: Se enfoca en el procedimiento para detener a una persona mientras atraviesa el sistema de control de acceso, ya sea para entrada o salida del perímetro protegido.
- Incomodidad Causada: Es importante considerar que el diseño del sistema no cause incomodidad que afecte la capacidad operativa de los elementos protegidos.
- Tráfico: Se debe tener en cuenta el flujo de personas que afectará al sistema, no solo el promedio, sino también el tráfico en horas pico.
- Costo: El sistema de control de acceso debe contar con la tecnología necesaria para proteger los objetos valiosos, y su costo debe ser acorde al valor de dichos objetos. (Sandhu, 1996)

1.2.2. Últimas tendencias en los sistemas de control de acceso.

Los sistemas de control de acceso desempeñan un papel fundamental en la protección de zonas restringidas al permitir o denegar la entrada de personas. En la actualidad, estos sistemas se han vuelto cada vez más importantes y son utilizados por muchas organizaciones. Los dispositivos utilizados en el control de acceso automatizado emplean tarjetas, tecnología biométrica o una combinación de ambas. Entre las técnicas biométricas, la más comúnmente utilizada es el reconocimiento de huellas dactilares. Sin embargo, han surgido nuevos y avanzados terminales en el mercado, como el reconocimiento facial, que ofrecen tasas de rechazo falsas más bajas, mayor confiabilidad y comodidad en comparación con los sistemas existentes hace algunos años. (Jain, 2011)

Otras tecnologías biométricas, como el reconocimiento de iris o retina, aún no se han consolidado completamente debido principalmente a su costo. Por otro lado, tecnologías de vanguardia, como los terminales que combinan el reconocimiento de huella dactilar y venas, están ganando terreno. Esta innovadora tecnología combina los mecanismos de protección de cada biometría, lo que la hace extremadamente resistente al fraude. En el caso de los dispositivos que requieren tarjetas, la tecnología RFID (Identificación por Radiofrecuencia) ha reemplazado a la banda magnética y al código de barras. Se trata de un sistema de almacenamiento y recuperación de datos remoto que transmite la identidad de un objeto, similar a un número de serie único, mediante ondas de radio. (Jain, 2011)

1.3. Estudio del estado del arte

Sistemas de control de acceso estudiados a nivel internacional

El sistema de control de acceso Arquero ofrece una solución de seguridad integrada y completa para controlar el acceso a diferentes áreas de una empresa. Se trata de una plataforma unificada desde la cual se pueden operar y supervisar todos los subsistemas de seguridad de una instalación. Es una solución todo en uno que proporciona un entorno gráfico y de gestión único. Permite la integración de sistemas de control de acceso y presencia, intrusión, detección de incendios, video-vigilancia, audio, control de horarios y automatización de edificios.

Este sistema es reconocido como una de las soluciones más completas y potentes disponibles en el mercado internacional. Ofrece una amplia gama de características y funcionalidades para garantizar la seguridad y protección de las instalaciones de una empresa. Con Arquero, las organizaciones pueden tener un control total sobre quién tiene acceso a sus dependencias, asegurando la protección de activos, la prevención de intrusiones y la gestión de la seguridad en general. (Pérez, 2020)

Políticas de Acceso

En Arquero, puedes definir políticas de acceso tanto para usuarios como para rutas de seguridad. Esto significa que puedes establecer los derechos de acceso de una persona o grupo a áreas protegidas específicas. Estas políticas permiten determinar qué usuarios o grupos tienen autorización para acceder a qué rutas y en qué horarios. Además, ofrece la posibilidad de exigir una autenticación adicional en determinados horarios o en accesos críticos. Por ejemplo, puedes requerir que los usuarios presenten una tarjeta de acceso o realicen una autenticación biométrica, como una huella dactilar o reconocimiento facial. Además, se puede solicitar la introducción de un PIN para fortalecer aún más la autenticación. (García, 2019)

Estas medidas adicionales de seguridad ayudan a garantizar que solo las personas autorizadas puedan acceder a áreas sensibles o restringidas en momentos específicos. Con la combinación de tarjetas/biometría y PIN, se refuerza la autenticación y se reduce el riesgo de acceso no autorizado.

Sistema de control de acceso *Easy way*

El sistema de control de accesos *Easy Way* es una solución segura diseñada para controlar el ingreso y salida de personas en todas las áreas de una empresa, lo que se conoce como control de personal. El software de control de acceso permite configurar el hardware desde una computadora, lo que facilita la gestión de la inclusión de planos del edificio, la generación de informes y la elaboración de estadísticas. Se destaca por su facilidad de implementación y operación, lo que lo convierte en una opción conveniente para empresas que necesitan un control rápido pero efectivo de las entradas y salidas. Para validar la identidad del usuario, este sistema utiliza la huella dactilar, que es única para cada persona y proporciona un nivel de seguridad completo. Al combinar esta característica con tecnología sofisticada, el proceso de identificación se agiliza y se reduce la posibilidad de fallos al reconocer la huella, ya que estas se actualizan constantemente.

Este software de control de personal cuenta con controles de acceso totalmente integrados y modulares. Es ampliamente utilizado en empresas para el control de acceso de personal y para el control de acceso a instalaciones o edificios. Además, es adaptable y puede satisfacer requerimientos particulares sofisticados que puedan ser solicitados. Esto permite establecer una relación personalizada con el cliente y su software de control de accesos. *Easy Way* ofrece una solución completa y segura para el control de accesos, brindando a las empresas la capacidad de gestionar el ingreso y salida de personas en sus instalaciones. (Smith, 2020)

Sistema de control de acceso con Tarjetas de Código de Barras

Los sistemas de control de acceso con tarjetas de código de barras son una forma óptica de almacenar información reconocible por los sistemas informáticos. Estas tarjetas tienen una apariencia similar a las tarjetas magnéticas, pero en lugar de una banda magnética, llevan impreso un código de barras. Además, pueden contar con una banda protectora que oculta el código y ayuda a prevenir la duplicación de la tarjeta por medio de fotocopias. (Zhang, 2020)

La principal ventaja de estas tarjetas es que, al pasarlas por el lector, no se requiere un cabezal para el razonamiento de la información. En su lugar, un haz de luz ilumina el

código de barras y el lector lo lee de manera rápida y precisa. Esto agiliza el proceso de autenticación y reduce la posibilidad de fallas.

El uso de tarjetas de código de barras en los sistemas de control de acceso ofrece una solución confiable para la identificación de personas y el control de entrada y salida en diferentes áreas. Estas tarjetas son ampliamente utilizadas en entornos empresariales, instituciones educativas y otros lugares donde se requiere un control de acceso seguro y ágil. (Zhang, 2020)

Desventajas

- Invariabilidad de la información: Una vez que se imprime el código de barras en la tarjeta, la información contenida en él no puede modificarse fácilmente. Esto puede ser un inconveniente si se necesita actualizar o cambiar la información almacenada en la tarjeta, ya que se requeriría emitir una nueva tarjeta con un código de barras actualizado.
- Etiquetas dañadas: Si una tarjeta con código de barras se daña o se desgasta, puede dificultar su lectura por parte del lector. Esto puede generar problemas al intentar escanear la tarjeta y puede requerir la emisión de una nueva tarjeta, lo que implica tiempo y costos adicionales.
- Legibilidad del código: En algunos casos, el número de 12 dígitos en la etiqueta puede estar dañado o desgastado hasta el punto en que no sea legible para el lector. Esto puede ocurrir debido al uso frecuente de la tarjeta o a condiciones adversas. En tales casos, la tarjeta puede volverse inutilizable y requerir una sustitución.

Sistema de control de acceso con Tarjetas de RFID (Identificación por Radio Frecuencia)

Los sistemas de identificación por radiofrecuencia (RFID) son ampliamente utilizados en diversos campos, especialmente en el ámbito de la seguridad. Algunas de las aplicaciones más comunes incluyen:

- Cruces fronterizos: Los sistemas RFID se utilizan para el control de acceso en cruces fronterizos, permitiendo una identificación rápida y segura de vehículos y personas.
- Credenciales de identidad: Las tarjetas de identificación con tecnología RFID se utilizan para autenticar y controlar el acceso a edificios, oficinas o áreas restringidas.
- Control vehicular: Los sistemas de peaje electrónico y los sistemas de control de acceso a estacionamientos utilizan la tecnología RFID para identificar y registrar vehículos.
- Identificación de ganado: Los dispositivos RFID se utilizan para marcar y rastrear animales en la industria ganadera, permitiendo un seguimiento preciso de su ubicación y condiciones de salud.
- Envío de paquetes: Las etiquetas RFID se utilizan en la logística y el transporte para rastrear y gestionar el flujo de paquetes, mejorando la eficiencia y la precisión en el proceso de entrega.
- Control de equipaje en aeropuertos: Los sistemas RFID se utilizan para identificar y rastrear el equipaje de los pasajeros, mejorando la seguridad y facilitando su manejo en los aeropuertos.
- Control de artículos para renta o préstamo: Los sistemas RFID se utilizan en bibliotecas, tiendas de alquiler y otros lugares donde se requiere un control de los artículos prestados, permitiendo un seguimiento automatizado y una gestión más precisa.

En general, los sistemas RFID constan de dos componentes principales: una etiqueta o transpondedor que se coloca en el objeto o se lleva encima de la persona, y un lector que emite señales de radiofrecuencia para leer y comunicarse con la etiqueta. Esta tecnología ofrece una forma rápida, sin contacto y precisa de identificación y control de acceso en diversas aplicaciones. (Jones, 2018)

Desventajas

- Información variable y reutilizable: A diferencia de los códigos de barras, la información almacenada en las etiquetas RFID puede ser modificada y actualizada. Esto puede ser una desventaja si no se controla adecuadamente, ya que podría haber inconsistencias en la información almacenada en las etiquetas.
- Identificación simultánea de productos: En entornos donde hay múltiples productos o etiquetas RFID cercanas entre sí, puede haber interferencias y dificultades para identificar y leer las etiquetas de manera individual. Esto puede generar errores y confusiones en el proceso de identificación y seguimiento.
- Problemas de confiabilidad de lecturas: Dado que la tecnología RFID es relativamente nueva, puede haber problemas de confiabilidad en las lecturas. Esto puede deberse a interferencias electromagnéticas, obstáculos físicos o problemas de calibración del lector. Estos problemas pueden afectar la precisión y la fiabilidad de los datos recopilados.
- Falta de operario de lectura: A diferencia de los sistemas de código de barras, que requieren que un operario escanee manualmente los códigos, los sistemas RFID no requieren una acción directa por parte de un operario. Esto puede generar problemas si no se implementan mecanismos de control adecuados, ya que las etiquetas podrían no ser leídas correctamente o podrían pasar desapercibidas.

Sistema de control de acceso *BioStar VideoPhone*

BioStar es un sistema de control de acceso con conectividad IP y seguridad biométrica desarrollado por Suprema Inc., un fabricante reconocido en esta área. El sistema BioStar ofrece varias características y funcionalidades, incluyendo *BioStar VideoPhone*. Es una aplicación para PC que permite al operador utilizar la PC y un dispositivo conectado como un sistema de interfono. Esta aplicación permite al operador visualizar quién está en la puerta y otorgar acceso si se aprueba. Es una solución única y útil para crear un sistema de intercomunicación por video sin la necesidad de utilizar equipos de videoteléfono adicionales.

Además de *BioStar VideoPhone*, *BioStar* es un software de administración de control de acceso completo. Cuenta con una arquitectura de sistema basada en TCP/IP y soporta lectores IP inteligentes. También ofrece integraciones con una amplia variedad de dispositivos de terceros y otros sistemas de seguridad. Esto brinda una reducción de costos y flexibilidad en el diseño del sistema de control de acceso. (Kim, 2019)

Sistema de control de acceso *ExClouds*

ExClouds es una aplicación basada en web del Sistema *ExpansE.* Esta aplicación ofrece una configuración rápida y sencilla, sin necesidad de instalar software ni realizar configuraciones en la computadora. Permite controlar todas las opciones de configuración del sistema, así como acceder a servicios adicionales, como la vigilancia en tiempo real desde cualquier panel y computadora en la red. La aplicación de servidor de *ExClouds* complementa el sistema con capacidades avanzadas de generación de informes. La interfaz web de *ExpansE* es fácil de usar y facilita la gestión de control de acceso, video y monitoreo de alarmas desde cualquier panel o navegador de Internet. Además, no se requiere que los operadores sean expertos, lo que simplifica aún más el uso de la aplicación. (Johnson, 2020)

La arquitectura completa de cliente-servidor de *ExpansE* permite el uso de múltiples clientes a diferentes niveles de usuario. Esto significa que varios usuarios pueden acceder al sistema y utilizar sus funciones de acuerdo con sus permisos y niveles de acceso designados.

Sistemas de control de acceso desarrollados en la Universidad de las ciencias Informáticas (UCI).

Sistema de Identificación

Este sistema brinda un servicio de certificación de identidad a otros sistemas informáticos, como los destinados al control del acceso. La estructura del sistema se compone de varios módulos: administración, configuración, identificación, detección de rostros y seguridad. Estos módulos desempeñan funciones específicas y trabajan en conjunto para garantizar el correcto funcionamiento del sistema. (Pérez A. , 2018)

Para su desarrollo, se utilizan *frameworks* conocidos como *Spring Framework* y *.Net.* Además, el sistema sigue una arquitectura por capas, lo que permite una mayor reutilización de sus componentes. Esto significa que los módulos de seguridad y configuración, por ejemplo, pueden ser utilizados en otras aplicaciones sin necesidad de volver a desarrollarlos desde cero.

Sistema de Control de Acceso a Comedores

El sistema de Control de Acceso a Comedores es una herramienta diseñada para gestionar y controlar el acceso de estudiantes, profesores y trabajadores a los comedores de las diferentes edificaciones donde se brinda el servicio de alimentación en la UCI.

El sistema se compone de dos partes principales: el control de acceso y la gestión de comensales. En el control de acceso, se utiliza un lector de códigos de barras ubicado en las puertas de los comedores. Cada persona cuenta con un código de barras en su identificación, el cual es registrado al momento de ingresar al comedor. Esto permite llevar un registro preciso de quiénes acceden y en qué momento lo hacen.

Por otro lado, la gestión de comensales permite a los directivos asignar comedores y puertas específicas a los diferentes grupos de personas. Esto facilita la organización y distribución de los recursos disponibles. Además, el sistema ofrece reportes detallados, como la cantidad de comensales desglosada por puerta o tipo de comida (desayuno, almuerzo, cena), lo que brinda información valiosa para la toma de decisiones.

Sistema de control de acceso a los laboratorios de producción (UCILAB)

El Sistema de Control de Acceso a los Laboratorios de Producción (UCILAB) es una herramienta diseñada para gestionar y controlar el acceso a los laboratorios destinados a los procesos productivos en la UCI. Su principal función es verificar y autorizar el acceso de las personas a los laboratorios, basándose en la información almacenada en la base de datos correspondiente, utilizando el número de identificación como referencia. Existen varias implementaciones de este sistema en la UCI, cada una específica para un área productiva en particular. Esto implica que no existe una base de datos centralizada que contenga todos los datos relacionados con los laboratorios de producción. Sin embargo, la aplicación a desarrollar debe ser capaz de gestionar y centralizar toda la información manejada por estas soluciones.

La aplicación a desarrollar debe permitir la gestión centralizada de los proyectos y las personas que tienen acceso a los laboratorios de producción. Esto implica tener un registro actualizado de los proyectos en curso, así como de las personas autorizadas para acceder a cada laboratorio. Además, debe proporcionar funcionalidades para agregar, modificar y eliminar proyectos y personas. Aunque la base de datos no sea centralizada, la aplicación debe ser capaz de recopilar y sincronizar la información relevante de las diferentes implementaciones del sistema en los distintos laboratorios de producción. Esto permitirá tener una visión completa y actualizada de todos los datos relacionados con los laboratorios. (Gómez, 2019)

Sistema de control de acceso para el centro CISED en la UCI

El Sistema de Control de Acceso para el Centro CISED en la UCI es una herramienta implementada en el centro de desarrollo de software CISED con el propósito de gestionar y controlar el acceso del personal a los laboratorios asignados a la producción. La funcionalidad principal del controlador de acceso es asegurar que solo las personas registradas puedan acceder a los servicios de red dentro de los laboratorios. Para lograr esto, se utiliza una aplicación web que recibe el número de solapín del usuario como identificación al momento de ingresar. Si el número de solapín coincide con los registros del controlador de entrada, se permite el acceso a los servicios de red. En caso contrario, se impide el acceso. (Rodríguez, 2020)

El sistema no tiene la capacidad de controlar las aplicaciones iniciadas por el usuario en las estaciones de trabajo dentro de los laboratorios. Su función principal se limita al control de acceso a los servicios de red.

1.3.1. Valoración de los sistemas estudiados

Después del estudio realizado a los sistemas de control de acceso descritos anteriormente, se han identificado varias funcionalidades y características comunes:

- Control de horarios y permisos concedidos: Estos sistemas permiten gestionar los horarios de acceso y los permisos otorgados a los usuarios.
- Generación de informes: Los sistemas son capaces de generar informes que registran eventos de identificación, administración y estaciones de trabajo. Estos informes son útiles para obtener un historial de los eventos ocurridos en el lugar de trabajo.
- Arquitectura Cliente-Servidor: Los sistemas utilizan una arquitectura Cliente-Servidor, lo que permite el uso de múltiples clientes a diferentes niveles de usuarios.
- Dentro de los sistemas analizados, los que se asemejan más a una posible solución para la problemática planteada, según la perspectiva del autor, son:
- Sistema de control de acceso a los laboratorios de producción (UCILAB): Existen varias implementaciones de este sistema en la UCI, específicas para cada área productiva. Esto implica que no existe una base de datos centralizada con todos los datos de los laboratorios de producción, y el acceso a la base de datos está restringido al personal del área correspondiente.

- Sistema de control de acceso a comedores: El estudio de este sistema permitió observar cómo se ejecuta el consumo de servicios de LDAP de la Universidad y modelar las funciones asociadas a visualizar el local correspondiente a cada especialista.

Las ideas de funcionalidades obtenidas de los sistemas controladores de acceso internacionales son:

- Uso de la arquitectura Cliente-Servidor: Esta arquitectura, utilizada por el sistema "*ExClouds*", implica que hay un cliente que solicita servicios a un servidor. Esta estructura permite una comunicación entre los dispositivos y el servidor central.
- Generación de informes de eventos de identificación: Los sistemas "*Easy Way*" y "Arquero" implementan la capacidad de generar informes detallados sobre los eventos de identificación. Estos informes históricos pueden ser útiles para el análisis de eventos ocurridos y el seguimiento de actividades.
- Seguridad en el acceso a datos, estaciones de trabajo y redes: El sistema "Digital Persona" se enfoca en garantizar la seguridad en el acceso de los usuarios a los datos, estaciones de trabajo y redes. Esto implica implementar medidas de autenticación y control de acceso para proteger la información sensible y evitar accesos no autorizados.

1.3.2. Tendencias y tecnologías actuales

Son diversas las tecnologías que pueden ser usadas para la elaboración de un producto de software, como:

- Sistemas de control de acceso con tarjetas inteligentes: Estos sistemas utilizan tarjetas inteligentes que contienen información de identificación y autenticación encriptada. Los lectores de tarjetas permiten el acceso solo a aquellos usuarios autorizados con tarjetas válidas.
- Cerraduras electrónicas: Las cerraduras electrónicas reemplazan las cerraduras tradicionales y pueden ser controladas de forma remota o mediante sistemas de autenticación, como códigos PIN o huellas dactilares.
- Sistemas de videovigilancia y reconocimiento facial: Los sistemas de videovigilancia se han vuelto más sofisticados, y algunos incluyen tecnología de reconocimiento facial para verificar la identidad de las personas que intentan acceder a una determinada área.

- Control de acceso biométrico: Los sistemas de control de acceso biométrico utilizan características físicas únicas, como huellas dactilares, iris o reconocimiento de voz, para autenticar y permitir el acceso a áreas restringidas.
- Sistemas de alarmas y detección de intrusiones: Estos sistemas monitorean y detectan cualquier intento de acceso no autorizado a través de sensores, como detectores de movimiento o sensores de puertas y ventanas.

Conclusión del estudio del arte.

Luego de investigar acerca de los sistemas de control de acceso, se identificaron las funcionalidades comunes, como el control de horarios y permisos, la generación de informes de eventos de identificación, la arquitectura Cliente-Servidor y el diseño centrado en la conveniencia del usuario. Con base en estos hallazgos, se decidió desarrollar una nueva solución utilizando el modelo de control de acceso

1.4. Herramienta y tecnologías a utilizar en el desarrollo de la aplicación móvil.

Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML) es un lenguaje gráfico ampliamente utilizado en la industria del software. Su propósito principal es visualizar, especificar, construir y documentar sistemas de software. UML proporciona un estándar para describir el diseño y la arquitectura de un sistema, cubriendo tanto aspectos conceptuales como procesos de negocios y funciones del sistema. Además, UML incluye herramientas y notaciones gráficas para representar visualmente diferentes aspectos del sistema, como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables.

Este lenguaje se utiliza para especificar un sistema de software en detalle, documentar los artefactos del sistema y facilitar su construcción. Sin embargo, es importante tener en cuenta que UML no prescribe un método o proceso específico para el desarrollo de software. En cambio, proporciona un conjunto de herramientas y notaciones que se pueden aplicar de diferentes maneras para respaldar diferentes metodologías de desarrollo, como el Proceso Unificado Racional (RUP). (Fowler, 2019 (3ª edición)).

Herramienta de modelado ***Visual Paradigm*)**

Visual Paradigm es una herramienta de modelado UML que ofrece una amplia gama de funcionalidades, incluyendo la capacidad de crear diversos tipos de diagramas de clases, realizar el análisis inverso de código, generar código a partir de los diagramas y crear documentación detallada. Estas características resultan especialmente útiles al diseñar nuevas estrategias para los sistemas de información y cuando los métodos y sistemas existentes no satisfacen las necesidades de la organización. Su uso ayuda a agilizar el proceso de construcción de aplicaciones de calidad, al tiempo que proporciona herramientas valiosas para el diseño de estrategias y la mejora de sistemas de información existentes. (Whitten, 2018 (8ª edición)).

En la elaboración de los diferentes diagramas y modelos empleados en la investigación se utilizará *Visual Paradigm*, para el modelado se utiliza el leguaje Unificado de Modelado (UML) el mismo abstrae y visualiza sistemas de la programación orientada a objetos. El lenguaje de modelado es, por lo tanto, una herramienta práctica para los desarrolladores de programas y sistemas. Por un lado, permite crear fotocalcos claros para proyectos de software, y por otro, presentar sistemas de programación complejos de forma comprensible para las personas que no están familiarizadas con la temática. Por ejemplo, si deseas presentar un proyecto de software de la aplicación de la empresa al responsable de marketing, por medio de un diagrama UML puedes ofrecer una visión general de las características más importantes de la aplicación (Ambler, 2004).

Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente altamente popular desarrollado por Microsoft. Es un software libre y multiplataforma que se puede utilizar en sistemas operativos como Windows, GNU/Linux y macOS. Una de las características destacadas de *VS Code* es su sólida integración con Git, lo que facilita el control de versiones y el trabajo colaborativo en proyectos de desarrollo de software. Además, cuenta con un potente soporte para depuración de código, lo que permite identificar y corregir errores. Otra ventaja significativa de *VS Code* es su amplia gama de extensiones, que amplían las funcionalidades del editor y permiten escribir y ejecutar código en diversos lenguajes de programación. Estas extensiones proporcionan herramientas adicionales, como resaltado de sintaxis, completado automático, integración con frameworks y librerías populares, entre otros. (Seacord, 2013).

Flutter

Flutter es un framework de código abierto desarrollado por Google que permite crear aplicaciones para diversos dispositivos utilizando un único código base. Con *Flutter*, es posible desarrollar aplicaciones compatibles con dispositivos Android e iOS, así como aplicaciones web y de escritorio. El kit de desarrollo de software (SDK) de *Flutter* se basa en el lenguaje de programación *Dart*, el cual fue creado por Google como un

sucesor potencial de *JavaScript. Dart* comparte similitudes con lenguajes como Java o C y puede ejecutarse directamente en el navegador. Además, los programas *Dart* también pueden ejecutarse en un servidor mediante el compilador Dart2js. La versatilidad y potencialidad de *Dart* lo convierten en una elección popular para el desarrollo de aplicaciones web y móviles. Su sintaxis es similar a la de JavaScript, lo que facilita la transición para aquellos familiarizados con este último. (Anderson, 2020)

Sistemas Gestores de Base de Datos

Un Sistema Gestor de Base de Datos (SGBD) o *Database* Management *System* (DBMS) es un conjunto de programas diseñados para administrar y gestionar la información contenida en una base de datos. Estos sistemas actúan como una interfaz entre la base de datos, el usuario y las aplicaciones, permitiendo controlar el acceso y manipulación de los datos. Las funciones de un SGBD incluyen la capacidad de almacenar y modificar información, así como facilitar el acceso a los activos de conocimiento de una organización. Los usuarios pueden realizar consultas, análisis y generar informes a partir de los datos almacenados en la base de datos. (Connolly, 2014 (6ª edición)).

PostgreSQL

La alta disponibilidad, la robustez, la consistencia y la tolerancia a fallos son características claves en un sistema de base de datos relacional. Estas cualidades proporcionan confiabilidad y estabilidad en el manejo de los datos, lo que es esencial para el funcionamiento de las operaciones empresariales. Esto convierte convierten a PostgreSQL en una elección ideal para la mayoría de proyectos, en los que su funcionalidad, la seguridad o la integridad referencial resulta de gran importancia. (Momjian, 2001).

Algunas de sus principales características son:

- **Alta concurrencia:** El sistema puede atender a múltiples clientes simultáneamente sin bloqueos, lo que permite un acceso a la información de las tablas.
- **Soporte para múltiples tipos de datos de manera nativa:** Además de los tipos de datos habituales, el sistema ofrece una amplia gama de tipos adicionales, como direcciones IP, figuras geométricas, entre otros.
- **Soporte a *triggers*:** Permite definir eventos y acciones asociadas a ellos, lo que facilita la automatización de tareas en respuesta a ciertos eventos.
- **Trabajo con vistas:** El sistema permite consultar los datos de manera diferente a cómo se almacenan, lo que brinda flexibilidad en la forma de acceder y presentar la información.
- **Objeto-relacional:** El sistema permite trabajar con datos como si fueran objetos y ofrece características propias de la orientación a objetos, como la herencia de tablas.
- **Soporte para bases de datos distribuidas:** El sistema es capaz de trabajar con bases de datos distribuidas, asegurando el éxito de las transacciones en todos los sistemas involucrados.
- **Soporte para gran cantidad de lenguajes:** PostgreSQL es capaz de trabajar con funciones internas, que se ejecutan en el servidor, escritas en diversos lenguajes. Además, ofrece interfaces para ODBC y JDBC, así como interfaces de programación para infinidad de lenguajes de programación.

1.5 Metodología de desarrollo

Una metodología de desarrollo de software se refiere al conjunto de prácticas y enfoques utilizados para estructurar, planificar y controlar el proceso de desarrollo de un sistema informático, suelen ser documentadas y respaldadas por organizaciones, tanto públicas como privadas, con el objetivo principal de mejorar la calidad del software en todas las etapas de desarrollo. Estas metodologías imponen un enfoque

disciplinado al desarrollo de software, con el propósito de hacerlo más predecible. Es importante tener en cuenta que no existe una metodología universal para el desarrollo de software, ya que cada proyecto tiene sus propias características y requiere adaptaciones específicas. Por lo tanto, es necesario configurar el proceso de acuerdo a las necesidades de cada proyecto. (Sommerville, 2015 (10ª edición)).

Para el desarrollo de la solución se utilizará la metodología Scrum. La cual permite el desarrollo rápido y flexible, especialmente en proyectos donde los requisitos pueden cambiar o evolucionar con el tiempo.

Metodología de desarrollo de software Scrum

Permite la entrega temprana de valor, se adapta a los cambios, fomenta la colaboración y proporciona una mayor visibilidad del progreso del proyecto. Su uso fundamental radica en dividir el trabajo en *sprints*, priorizar el *backlog,* tener reuniones estructuradas y establecer roles y responsabilidades claros. Estas características y enfoques hacen que Scrum sea una metodología ágil popular y efectiva para el desarrollo de proyectos. (Sutherland, 2014)

1. **Adaptabilidad al cambio:** Scrum se basa en la premisa de que los requisitos y prioridades pueden cambiar durante el proyecto. Esta metodología ofrece flexibilidad para ajustar las prioridades en cada sprint y responder rápidamente a las necesidades cambiantes del cliente o del mercado.
2. **Entrega temprana de valor:** Scrum se enfoca en la entrega continua de incrementos de producto funcionales después de cada sprint. Esta práctica permite que los clientes o usuarios prueben y brinden retroalimentación temprana, lo cual se traduce en una mayor satisfacción y adaptación del producto a sus necesidades.
3. **Colaboración efectiva:** Scrum fomenta la comunicación y la colaboración activa entre los miembros del equipo y los stakeholders. Este enfoque ayuda a alinear las expectativas y facilita una toma de decisiones más rápida y efectiva.
4. **Mayor visibilidad y transparencia:** Scrum proporciona una visión clara del progreso del proyecto a través de artefactos y reuniones específicas.

Esto promueve una mayor transparencia en el equipo y facilita la identificación temprana de problemas o riesgos.

5. **Mejora continua:** Scrum incluye revisiones de sprint y retrospectivas, donde el equipo puede evaluar su trabajo y encontrar formas de mejorar constantemente su rendimiento y eficiencia.

1.6 Conclusiones del capítulo

En este capítulo se examinaron las condiciones y problemáticas relacionadas con el objeto de estudio. A través de los conceptos y definiciones planteados, se identificaron las condiciones específicas que contribuyen al problema, lo que permitió establecer los objetivos generales y específicos para este trabajo. Además, se seleccionaron las herramientas y la metodología de desarrollo que resultaron ser la opción más adecuada para abordar esta aplicación.

- La descripción de los conceptos clave proporcionó una comprensión más profunda de los temas relacionados con el objeto de estudio de esta investigación.
- El análisis de las aplicaciones descritas permitió abordar aspectos relacionados con las tecnologías y su uso para mejorar el control de acceso a la entidad XETiD.
- El estudio del estado del arte reveló la necesidad de implementar una aplicación que contribuya al control de entrada y salida de todo el personal, con el fin de tener un mejor control de acceso a la entidad.
- El análisis de diversas herramientas, lenguajes y tecnologías permitió adquirir los conocimientos necesarios para seleccionar las más adecuadas en el desarrollo de la solución.

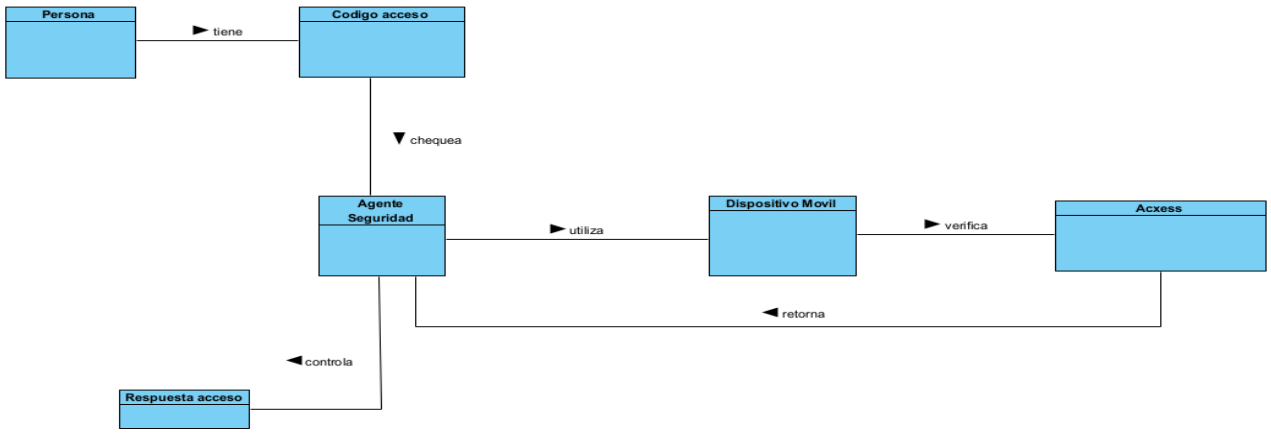
CAPÍTULO 2 DISEÑO DE LA APLICACIÓN PARA SISTEMA DE CONTROL DE ACCESO

En este capítulo se establece el modelo de dominio y se propone una solución al problema planteado. Se identifican, especifican y describen los requisitos funcionales y no funcionales. Se presenta una aproximación a la implementación a través de la etapa de diseño, donde se crea el mapa de navegación, los prototipos de interfaz de usuario y el diagrama de clases. Además, se selecciona el estilo arquitectónico y se modela la arquitectura del sistema utilizando el patrón modelo-vista-plantilla y el patrón modelo-vista-modelo de vista

2.1 Modelo de Dominio

Un modelo del dominio es una representación visual de las clases conceptuales u objetos del mundo real en un dominio específico de interés. Se utiliza cuando no se puede determinar claramente el proceso del negocio con fronteras definidas y cuando los flujos de información son difusos, con múltiples orígenes y eventos. También se aplica en casos de solapamiento de responsabilidades y múltiples responsabilidades. Este modelo se representa mediante un conjunto de diagramas de clases UML en los que no se definen operaciones. (Fowler M. , 2019 (4ª edición)).

Figura 1Modelo de Dominio (Elaboración propia)



2.1.2 Descripción de Conceptos

A continuación, se presenta la descripción de los conceptos modelados en el modelo conceptual.

- ✓ Persona: Personal laboral que requiere el acceso a la entidad
- ✓ Código de acceso: Código QR o código de barras como recurso de identificación
- ✓ Guardia de seguridad: Persona asignada para controlar los eventos de ingreso / egreso del personal
- ✓ Dispositivo Móvil: Medio por el cual se va a realizar las lecturas de identificación mediante el uso de cámara
- ✓ Acxess: Sistema de Control de acceso
- ✓ Permitir/ denegar: Respuesta del punto de control sobre el intento de ingreso por el punto de control.

2.2 *Product Backlog* (Requisitos funcionales y no funcionales):

La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema. (Sommerville, "Requirements Engineering: A Good Practice Guide", 1997)

2.2.1 Requisitos Funcionales:

Un requisito funcional es una capacidad o condición que el sistema debe cumplir.

ID	Nombre del Requisito	Descripción	Prioridad
RF1	Escanear código QR	El sistema permite escanear código QR	Media
RF2	Escanear código de barra.	El sistema permite escanear código de barra.	Media
RF3	Controlar acceso a la empresa.	El sistema permite controlar acceso a la empresa.	Media
RF4	Registrar evento de acceso.	El sistema permite registrar evento de acceso.	Media
RF5	Enviar notificación en tiempo real.	El sistema permite enviar notificación en tiempo real.	Media
RF6	Adaptar interfaz de usuario a diferentes tamaños de pantalla y orientaciones.	El sistema permite adaptar interfaz de usuario a diferentes tamaños de pantalla y orientaciones.	Alta
RF7	Integrarse con servicio nativo de cámara	El sistema permite integrarse con servicio nativo de cámara	Alta
RF8	Mostrar punto de control.	El sistema permite mostrar punto de control.	Alta
RF9	Modificar punto de control.	El sistema permite modificar punto de control.	Alta
RF10	Acceder privilegios admin.	El sistema permite acceder privilegios.	Alta
RF11	Modificar privilegios admin.	El sistema permite modificar privilegios.	Alta
RF12	Cambiar clave de acceso a privilegios admin	El sistema permite cambiar la clave de acceso	Alta
RF13	Mostrar estadísticas de acceso.	El sistema permite mostrar estadísticas de acceso.	Alta
RF14	Consultar el sistema Acxess y reconocer la persona	El sistema permite consultar el sistema Acxess y reconocer la persona	Alta
RF15	Registrar horario/fecha de entrada y salida	El sistema permite registrar horario/fecha de entrada y salida	Alta
RF16	Sincronizar acceso.	El sistema permite sincronizar acceso.	Alta
RF17	Detectar posibles intrusos o violaciones	El sistema permite detectar posibles intrusos o violaciones	Alta

Tabla 0 Descripción del *Product Backlog*

2.2.2 Requisitos no Funcionales

Los requisitos no funcionales (RNF) son restricciones o estándares de calidad que imponen limitaciones en el diseño o la implementación de un producto. Estos requisitos se refieren a las propiedades o cualidades que el producto debe poseer. (Gause, 1989).

Según la norma ISO/IEC 25010 el modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. La calidad del producto software se puede interpretar como el grado en que dicho producto

RNF1-Requisitos de Hardware:

Los dispositivos móviles deben tener habilitado la opción de cámara para realizar el registro

RNF2-Usabilidad:

Capacidad del producto software para ser entendido, aprendido, usado y resultar atractivo para el usuario, cuando se usa bajo determinadas condiciones.

- El sistema debe poseer una interfaz de usuario fácil e intuitiva.

2.2.3 Modelado de los requisitos:

Según la metodología SCRUM, las Historias de usuario son unas de las variantes que permiten encapsular los requisitos funcionales del sistema.

Historias de Usuario

En el marco de la ingeniería de requisitos ágil, las historias de usuario se utilizan como una herramienta de comunicación que combina las ventajas de la comunicación escrita y verbal. Estas historias describen, en una o dos frases, una funcionalidad del software desde la perspectiva del usuario, utilizando el lenguaje que este último emplearía. (Cohn, 2004)

Elementos de la Historia de Usuario:

Número: Indica el id del requisito funcional.

Usuario: Indica que usuario utilizara este requisito.

Nombre de Historia: Nombre del requisito funcional.

Prioridad en Negocio: Representa que requisitos deben implementarse primero. La prioridad se establece como: alta, media o baja, según la importancia de este requisito para el negocio.

Riesgo en Desarrollo: Evidencia el nivel de riesgo en caso de no realizarse la HU. El riesgo se establece como medio, alto o bajo.

Puntos Estimados: Este atributo no es más que una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU, 1 punto equivale a 1 día de programación.

Programador Responsable: Persona encargada de implementar el requisito.

Iteración Asignada: Determina en que sprint se comienza a implementar el requisito según su prioridad.

Descripción: Describe de forma breve que espera el usuario de esta funcionalidad.

Validación: Confirma y valida que el requisito cumple las expectativas del usuario.

A continuación, se muestran algunas de las historias de usuarios más significativas:

Tabla 1 Historia de usuario RF8

	Historia de Usuario
Número:8	**Usuario:** Guardia Seguridad
Nombre de Historia: Modificar punto de control.	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 0.5	**Iteración Asignada:**1
Programador Responsable: Danel Castro García	
Descripción: El Guardia de Seguridad desea modificar por cual punto de control va a estar realizando las operaciones de acceso a la entidad.	
Validación: El guardia puede modificar el punto de control por el cual va a dar acceso.	

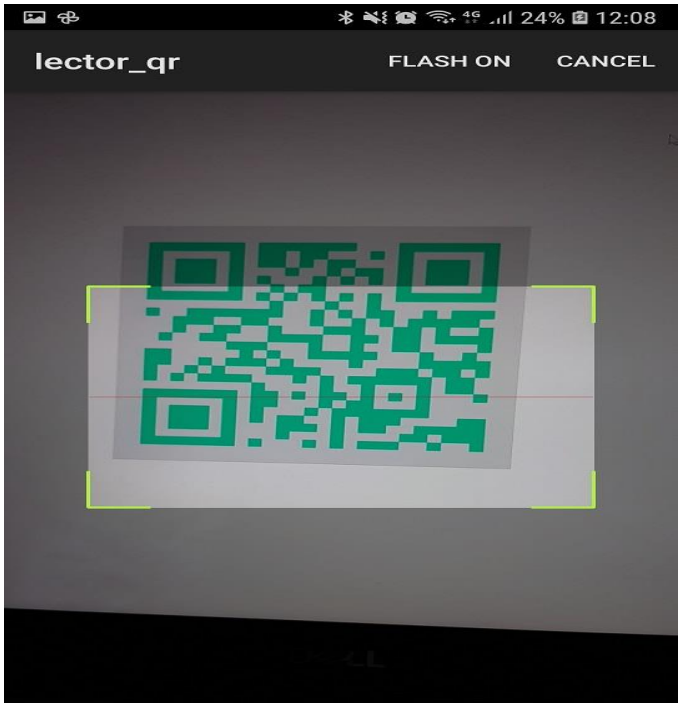
Tabla 2 Historia de usuario RF1

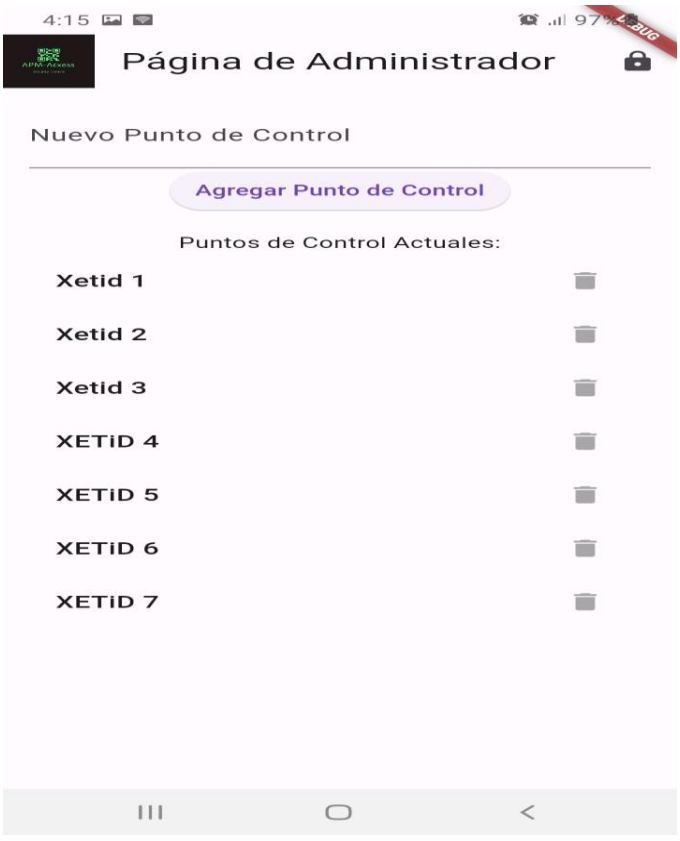
	Historia de Usuario
Número:1	**Usuario:** Guardia Seguridad
Nombre de Historia: Escanear código QR	
Prioridad en Negocio: Alta	**Riesgo en Desarrollo:** Alto
Puntos estimados: 0.5	**Iteración Asignada:**1
Programador Responsable: Danel Castro García	
Descripción: El Guardia de Seguridad desea escanear el código de identificación QR del personal que desea entrar a la entidad	
Validación: El guardia puede escanear la identificación de la persona	

2.3 Diseño de la Interfaz de Usuario

El diseño de la interfaz de usuario es un proceso que permite establecer una comunicación efectiva entre los usuarios y la computadora. Siguiendo un conjunto de principios de diseño, se identifican los objetos y acciones relevantes para la interfaz, y se crea una plantilla de pantalla que sirve como base para el prototipo de la interfaz de usuario. (Shneiderman, 2016 (6ª edición)).

A continuación, se muestra el prototipo de interfaz de usuario que debe tener el sistema a implementar.





2.3.1 Arquitectura especializada:

A continuación, se describe el concepto de arquitectura de software, se representan los patrones arquitectónicos, el estilo arquitectónico y los patrones de diseño a utilizar en la implementación del sistema

La arquitectura de software se refiere al diseño estratégico que aborda los requisitos globales de una actividad. Su implementación se realiza a través de paradigmas de programación, estilos arquitectónicos y estándares de ingeniería de software basados en componentes. Además, se consideran aspectos como patrones arquitectónicos, seguridad, escalabilidad, integración y consistencias regidas por leyes. (Bass, 2012 (3ª edición))

Es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan, y la estructura de datos que utilizan los componentes.

2.3.2 Patrón Arquitectónico:

Un patrón arquitectónico proporciona una descripción de un problema de diseño específico y recurrente que se encuentra en contextos de diseño particulares. Además, presenta un esquema genérico que ha demostrado ser exitoso para su solución. El esquema de solución se especifica mediante la descripción de los componentes que lo componen, sus respectivas responsabilidades y cómo colaboran entre sí. (Buschmann, 1996).

2.3.3 Estilo Arquitectónico:

Un estilo arquitectónico se refiere a un conjunto de tipos de componentes que describen patrones o interacciones a través de ellos. Este estilo tiene una influencia en toda la arquitectura del software y puede combinarse en una propuesta de solución. Los estilos arquitectónicos ayudan a abordar el tratamiento estructural que se enfoca en la teoría, la investigación académica y la arquitectura a un nivel de abstracción más alto (Shaw, 1996).

Modelo: El modelo, dentro de MVVM es el encargado de representar el modelo del negocio, proveyendo de esta manera la base necesaria para la manipulación de los datos de la aplicación.

Vista: La vista es la parte encargada de la parte visual de nuestra aplicación, no teniéndose que ocupar en ningún momento en el manejo de datos. En MVVM la vista tiene un rol activo, esto significa que en algún momento la vista recibirá o manejará algún evento y tendrá que comunicarse con el modelo, para poder cumplir el requerimiento.

Modelo de Vista: Es el encargado de ser la capa intermedia entre el modelo y la vista, procesando todas las peticiones que tenga la vista hacia el modelo, además de tener que ocuparse de manejar las reglas del negocio, la comunicación con aplicaciones externas.

Figura 2 Modelado del patrón arquitectónico (Elaboración propia)

Ventajas:

- Separación clara de la lógica de la interfaz y la lógica de negocio.

- Facilita el desarrollo colaborativo entre diferentes roles (diseñadores, desarrolladores de interfaz, desarrolladores de datos).

- Permite escribir pruebas unitarias más eficaces para la lógica de presentación.

- Promueve la reutilización de *ViewModel* en diferentes vistas.

- Facilita el enlace de datos bidireccional, lo que mejora la coherencia entre la Vista y el *ViewModel.*

Desventajas:

- Puede ser percibido como inicialmente más complejo para nuevos desarrolladores.

- Puede aumentar la cantidad de archivos y la complejidad en proyectos.

- Puede consumir más memoria, especialmente con *ViewModels* complejos.

- No siempre es necesario y puede ser excesivo para aplicaciones pequeñas y simples.

- Requiere una biblioteca de enlace de datos para su implementación completa.

2.3.4 Patrones del diseño:

Los patrones de diseño tienen como objetivo recopilar una colección de recursos que ayuden a los desarrolladores de software a abordar problemas comunes durante el proceso de desarrollo. Estos patrones proporcionan un lenguaje común que facilita la comunicación de perspectivas y experiencias relacionadas con esos problemas y sus soluciones. La formalización de estas soluciones y sus interacciones contribuye al crecimiento exitoso del conjunto de conocimientos que respalda nuestra comprensión de las mejores prácticas en arquitectura para satisfacer las necesidades de los usuarios. (Erich Gamma, 1994)

Dentro del campo del desarrollo de software, existen diversos patrones de diseño que describen principios fundamentales en la construcción de objetos. Estos patrones se agrupan principalmente en dos categorías amplias: los Patrones de Software para la Asignación General de Responsabilidades (GRASP) y los Patrones del "*Gang of Four*" (GoF).

2.3.5 Patrones Generales de Asignación de Responsabilidades de Sistemas

Patrones GRASP

Los patrones GRASP son una herramienta que ayuda a comprender el diseño de objetos y aplican un enfoque sistemático, racional y explicativo para el diseño. Codifican principios básicos ampliamente utilizados en el diseño de software. (Larman, 2005). Los patrones GRASP empleados en el desarrollo son descritos a continuación.

Patrón Experto

El patrón de asignación de responsabilidad busca resolver el problema de cómo asignar una tarea a la clase que posee o puede acceder a los datos necesarios Al realizar estas asignaciones de responsabilidades de forma adecuada, los sistemas tienden a ser más comprensibles, mantenibles y escalables, lo que brinda la garantía de poder reutilizar los componentes en futuras aplicaciones. Este principio de guía básico se utiliza con frecuencia en el diseño de objetos y juega un papel fundamental en la asignación de responsabilidades. (Erich Gamma R. H., 1994.)

Este patrón se utiliza en la clase ***AdminPage.*** Esta clase es responsable de crear instancias de objetos y tiene acceso a la información necesaria para hacerlo, tiene acceso a ***puntoControlController***, que es un objeto de la clase Puntos_Control.

```
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'package:prueba/Puntos_Control.dart'; // Asegúrate de importar la clase
PuntosDeControlManager
import 'package:shared_preferences/shared_preferences.dart';

class AdminPage extends StatefulWidget {
  final String adminPassword;

  AdminPage(this.adminPassword);

  @override
  _AdminPageState createState() => _AdminPageState();
}

class _AdminPageState extends State<AdminPage> {
  TextEditingController puntoControlController = TextEditingController();
```

Figura 3 Patrón experto (Elaboración propia)

2.3.6 Patrón Creador

El patrón de diseño Creador ofrece orientación sobre cómo asignar responsabilidades relacionadas con la creación de objetos, una tarea común en la programación. Su concepto principal es identificar un "creador" que esté vinculado al objeto que se está creando en una situación específica. Esto ayuda a reducir el acoplamiento entre los componentes del software y promueve una mejor organización del código. (Erich Gamma R. H., "Design Patterns: Elements of Reusable Object-Oriented Software", 1994)

Este patrón ayuda a identificar quien debe ser el responsable de la creación de nuevos objetos. Es donde se asigna la responsabilidad de que una clase B cree un objeto de la clase A.

```
class _MyHomePageState extends State<MyHomePage> {
  String _counter = "";
  String _value = "";
  String selectedPuntoDeControl = 'Punto de Control 1';

  Future<void> _incrementCounter(BuildContext context) async {
    _counter = await FlutterBarcodeScanner.scanBarcode(
      "#004297",
      "Cancel",
      true,
      ScanMode.DEFAULT,
    );
    setState(() {
      _value = _counter;
    });
DateTime now= DateTime.now();
String fechaHoraActual = "${now.toLocal()}";

    int idCard = int.tryParse(_counter) ?? 0;
    final resultado = await VerificarPersonaPage(idCard:
idCard).verificarPersona();
```

Figura 4 Patrón creador (Elaboración propia)

2.3.7 Los patrones GOF

Los patrones GOF, conocidos como la "Banda de los Cuatro", son herramientas esenciales en el campo de la programación. Estos patrones se enfocan en la organización de diversos tipos de objetos y cómo interactúan entre sí. Su objetivo principal es establecer relaciones entre clases y crear estructuras más complejas. Además, facilitan la creación de conjuntos de objetos que colaboran en la ejecución de tareas complicadas. (Erich Gamma R. H., "Design Patterns: Elements of Reusable Object-Oriented Software", 1994).

Un ejemplo de un patrón GOF es el "Método de Fabricación" *(Factory Method*). Este patrón define una interfaz para crear objetos, pero delega la decisión sobre qué clase concreta instanciar a las subclases. Su objetivo principal es proporcionar instancias de diferentes tipos de objetos, generalmente derivados de una misma clase base, que se distinguen por aspectos específicos de su comportamiento. En el proyecto, se puede observar que las clases creadoras derivan de la clase base AdminPage.

```
class _AdminPageState extends State<AdminPage> {
```

Figura 5 Ejemplo de patrón método de fabricación (Elaboración propia)

2.3.8 Patrón Inyección de dependencias

Patrón Inyección de dependencias: Este patrón es utilizado en los contenedores de servicios (o contenedores de inyección de dependencias), los cuales son objetos DART que gestionan la creación de instancias de servicios, es decir, objetos.

```
final authProvider = Provider.of<AuthProvider>(context);
```

Figura 6 Ejemplo de patrón inyección de dependencias (Elaboración propia)

El patrón Iterador es de tipo comportamiento a nivel de objetos. A través de su utilización es posible acceder de forma porque es posible utilizar nuevas formas de recorrer una estructura con solo modificar el iterador en uso, cambiarlo por otro o definir uno nuevo secuencial a cada uno de los elementos de un objeto agregado sin exponer su representación interna. Además, permite realizar recorridos sobre objetos compuestos independientemente de la implementación de estos. Su utilización tributa al incremento de la flexibilidad. (Erich Gamma R. H., "Design Patterns: Elements of Reusable Object-Oriented Software" 1994)

2.4 Conclusiones del capítulo

La utilización de un modelo de negocio bien definido y la generación de los artefactos del mismo, permitieron conocer, el funcionamiento del proceso de control de acceso en la empresa XETiD. Se identificaron 17 requisitos funcionales y 3 requisitos no funcionales que le aportan cualidades significativas al producto.

CAPÍTULO 3. IMPLEMENTACIÓN Y VALIDACIÓN DE LA APLICACIÓN MÓVIL PARA EL SISTEMA DE CONTROL DE ACCESO DE LA XETID

En este capítulo se abordan las actividades asociadas a los procesos de implementación, que incluyen la planificación del sprint y la creación del diagrama de despliegue. Se llevan a cabo pruebas en la aplicación para verificar su funcionamiento adecuado, seguridad y la satisfacción del cliente con cada incremento del producto de software entregado.

3.1 Implementación del Sistema

La etapa de implementación es una fase crítica en el desarrollo de software, ya que es aquí donde se lleva a cabo la definición y organización del código para la solución propuesta. Durante esta etapa, se materializan en forma de código todos los artefactos de implementación, descripciones y arquitectura propuestos en las fases de análisis y diseño. Esto permite crear el producto final requerido por el cliente. (Sommerville I. , 2016)

3.1.1 Planificación del Sprint Backlog

Durante la fase de planificación de un sprint, el equipo elabora una lista de tareas que deben ser completadas. Estas tareas son asignadas a cada miembro del equipo junto con el tiempo estimado para su realización. Este enfoque permite descomponer el proyecto en unidades más pequeñas y medir el progreso en función de las tareas completadas, estas deben tener una duración estimada similar, generalmente entre 15 y 30 días. (Schwaber, 2020)

Figura 7 Planificación del Sprint (Elaboración propia)

Tabla 3 Planificación del Sprint 0(Elaboración propia)

ID	Tarea	Responsable	Estado	Fecha inicial	Fecha final
T1	Montaje del ambiente de desarrollo con la integración de los marcos de trabajo seleccionados para el desarrollo.	Danel Castro García	Resuelta	1/09/2023	15/09/2023
T2	Realizar el mapeo de la base de datos.	Danel Castro García	Resuelta	1/09/2023	15/09/2023
RF1	Implementar funcionalidad: Escanear código QR	Danel Castro García	Resuelta	1/09/2023	15/09/2023
RF2	Implementar funcionalidad: Escanear código de barra	Danel Castro García	Resuelta	1/09/2023	15/09/2023
RF3	Implementar funcionalidad: Controlar acceso a la empresa	Danel Castro García	Resuelta	1/09/2023	15/09/2023
RF6	Implementar funcionalidad: Adaptar interfaz de usuario a diferentes tamaños de pantalla y orientaciones	Danel Castro García	Resuelta	1/09/2023	15/09/2023
RF7	Implementar funcionalidad: Integrarse con servicio nativo de cámara	Danel Castro García	Resuelta	1/09/2023	15/09/2023

Tabla 4 Planificación del Sprint 1 (Elaboración propia)

ID	Tarea	Responsable	Estado	Fecha inicial	Fecha final
RF8	Implementar funcionalidad: Mostrar punto de control.	Danel Castro García	Resuelta	1/09/2023	15/08/2023
RF10	Implementar funcionalidad: Acceder privilegios admin	Danel Castro García	Resuelta	1/09/2023	15/08/2023

RF11	Implementar funcionalidad: Modificar privilegios admin	Danel Castro García	Resuelta	16/09/2023	31/08/2023
RF12	Implementar funcionalidad: Cambiar clave de acceso a privilegios admin	Danel Castro García	Resuelta	16/09/2023	31/08/2023
RF15	Implementar funcionalidad: Registrar horario/fecha de entrada y salida	Danel Castro García	Resuelta	16/09/2023	31/08/2023
RF16	Implementar funcionalidad: Sincronizar acceso.	Danel Castro García	Resuelta	16/09/2023	31/08/2023

3.1.2 Resultados de la Revisión del Sprint

A continuación, se presentan los resultados obtenidos con cada iteración del Sprint, se obtuvieron un total de 11 no conformidades, las cuales se distribuyen en 6 en la primera iteración de las cuales se solucionaron 4, 3 no conformidades en la segunda las cuales fueron solucionadas y se le dio solución en esta iteración a las otras 2 no conformidades del sprint anterior y en la tercera iteración se obtuvieron 2 no conformidades y también fueron resueltas por parte del equipo de desarrollo. En la última iteración se halló 1 no conformidad a la cual se le dio solución y se entregó el producto de software listo y sin ninguna no conformidad.

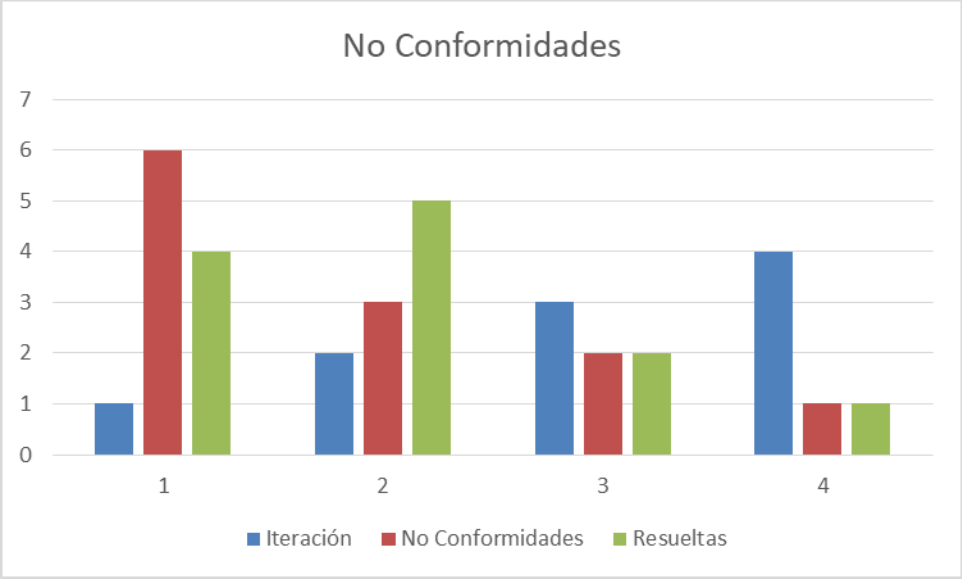


Figura 8 Resultados de la Revisión del Sprint (Elaboración Propia)

3.1.3 Diagrama de Despliegue:

El diagrama de despliegue representa la disposición de los nodos de procesamiento durante la ejecución, así como las conexiones de comunicación entre ellos y las instancias de componentes y objetos que residen en dichos nodos. Este diagrama incluye nodos, dispositivos y conectores. El objetivo principal del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre ellos en el sistema. (Fowler M. , "UML Distilled: A Brief Guide to the Standard Object Modeling Language" , 2019)

Este tipo de diagrama se realiza con el objetivo de mostrar las relaciones físicas de los distintos nodos que componen un sistema y la distribución de los componentes sobre dichos nodos. Además, se modela la arquitectura en tiempo de ejecución de un sistema.

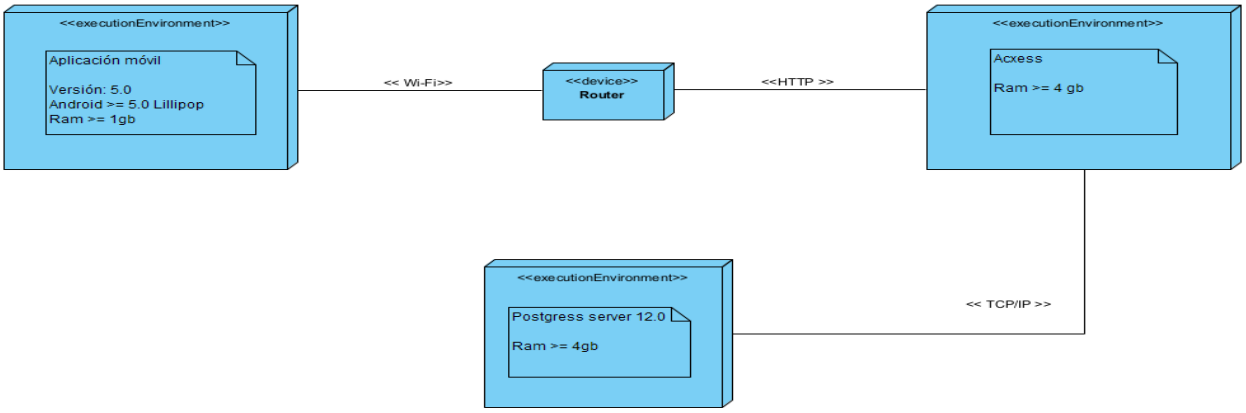


Figura 9 Diagrama de Despliegue (Elaboración propia)

- Servidor de Base de datos: Es el nodo encargado de controlar el acceso a los datos y guardar toda la información referente a los mismos. Navegador Web, Memoria RAM >= 4gb

- Móvil Cliente: Es el nodo que representa la estación de trabajo que permite al usuario mediante el protocolo HTTPS y el puerto 5432 acceder a la información del personal y realizar la función de acceso a la entidad Android >= 5.0 *Lollipop*, memoria RAM >= 1gb

3.1.4 Estándares de Codificación:

Un conjunto completo de directrices de codificación abarca todos los aspectos relacionados con la generación de código. Aunque los programadores deben seguir estas directrices de manera juiciosa, es importante que el estándar sea práctico en su implementación. El código fuente debe mantener un estilo coherente que dé la impresión de haber sido escrito por un único programador en una sola ocasión. (McConnell, 2004)

Para facilitar el entendimiento del código y establecer un modelo a seguir, se establecieron los estándares de codificación mostrados a continuación para el lenguaje de programación *Dart* en el *framework* de desarrollo *Flutter*.

- ✓ Las líneas podrán tener ochenta caracteres o menos.
- ✓ La declaración de importación debe escribirse en líneas separadas.

- ✓ La declaración de importación debe colocarse al principio del archivo, después de la descripción del módulo y la cadena de documentos, y antes de las variables globales.
- ✓ Los módulos deben tener un nombre corto y en minúscula.

```
import 'package:flutter/material.dart';
import 'package:flutter_barcode_scanner/flutter_barcode_scanner.dart';
import 'package:prueba/verificar_persona.dart';
import 'package:provider/provider.dart';
import 'package:prueba/Puntos_Control.dart';
import 'package:sqflite_sqlcipher/sqflite.dart'; // Importando el paquete de
SQLCipher: cifrado de la db
import 'package:path/path.dart';
```

Figura 10 Importaciones en Django (Elaboración propia)

3.2 Pruebas de Software

Las pruebas de software son un proceso fundamental que consiste en poner a prueba el software con el objetivo de detectar y corregir errores. Estas pruebas se encargan de verificar que el software cumpla correctamente con una función específica y validan que las acciones realizadas respondan a los requisitos establecidos por el cliente. (Myers, 2011).

3.2.1 Estrategia de Pruebas

La estrategia de pruebas de software se encarga de brindar una guía detallada sobre los pasos necesarios para llevar a cabo las pruebas, incluyendo la planificación, ejecución, diseño de casos de prueba, así como la recolección y evaluación de los resultados. Esta estrategia proporciona información sobre el esfuerzo, tiempo y recursos requeridos para llevar a cabo cada etapa de las pruebas de manera efectiva. (Martin, 2008).

Para la validación del sistema se propone realizar la siguiente estrategia de pruebas:
Las pruebas funcionales: Se centran en la ejecución, revisión y retroalimentación de las funcionalidades implementadas en el software, así como en las especificaciones definidas por el usuario. Estas pruebas buscan evaluar el sistema utilizando modelos de pruebas, enfocándose en los requisitos funcionales del sistema y su correcto funcionamiento.

Durante estas pruebas, se obvia la estructura de control y se pone énfasis en suministrar datos de entrada y estudiar la salida generada, sin preocuparse por el funcionamiento interno del módulo en cuestión. (Desikan, 2006)

Las pruebas de integración: Se encargan de evaluar la interacción entre dos o más elementos, como clases, módulos, paquetes o subsistemas, así como la interacción del sistema con su entorno de producción. Estas pruebas tienen como objetivo verificar el correcto ensamblaje de los diferentes componentes, una vez que han sido probados de forma individual, para comprobar que interactúan de manera adecuada a través de sus interfaces, tanto internas como externas. Además, se verifica que cubran la funcionalidad establecida y se ajusten a los requisitos no funcionales especificados durante las verificaciones correspondientes. (Pressman, "Software Engineering: A Practitioner's Approach", 2014).

Pruebas de aceptación: Las pruebas de aceptación son llevadas a cabo principalmente por los usuarios, con el respaldo del equipo del proyecto. Su objetivo es confirmar que el sistema se encuentra completo, que cumple de manera precisa con los requisitos de la organización y que es aceptado por los usuarios finales. Estas pruebas se enfocan en evaluar el comportamiento y las capacidades del sistema o producto en su entorno real. Durante esta etapa, es necesario tener en cuenta los siguientes elementos: (Black, 2019)

- ✓ Se verifica la adecuación al uso del sistema por parte de usuarios de negocio.
- ✓ Las pruebas se realizan utilizando el entorno del cliente.
- ✓ El entorno de cliente puede reproducir nuevos fallos.

3.3 Aplicación de las Pruebas:

A continuación, se muestra la aplicación de las pruebas que se le efectuaron a la solución:

Tabla 5 Aplicación de pruebas (Elaboración propia)

Pruebas	Método	Técnica
Prueba de unidad	Caja blanca	Camino básico
Prueba de Aceptación	Caja negra	Partición de equivalencia

Pruebas Funcionales:

En este tipo de pruebas, se lleva a cabo la ejecución de los diferentes servicios con datos tanto correctos como incorrectos. En el caso de que los datos sean incorrectos, se verifica que los mensajes de error sean los esperados. Por otro lado, en el caso de datos correctos, se comprueba que los resultados obtenidos sean los esperados. (Offutt, 2013).

Método de Prueba de Caja Negra:

El enfoque de caja negra se utiliza para evaluar la interfaz del software y analizar algún aspecto funcional del sistema que tiene poca o ninguna relación con la estructura interna del software. (Beizer, 1995).

Se concentran en los requisitos funcionales del software, tratando de encontrar los siguientes errores (Offutt, 2013):

- ✓ Funciones incorrectas o faltantes.
- ✓ Errores de interfaz.
- ✓ Errores de estructuras de datos o en acceso a bases de datos externas.
- ✓ Errores de comportamiento o desempeño.
- ✓ Errores de inicialización y término.

Los casos de pruebas pretenden demostrar que:

- ✓ Las funciones del software son operativas.
- ✓ La entrada se acepta de forma correcta.
- ✓ Se produce una salida correcta.
- ✓ La integridad de la información externa se mantiene.

En el diseño de los casos de prueba utilizando el enfoque de Caja Negra, se empleó la Técnica de Partición Equivalente. Esta técnica divide el dominio de entrada de un programa en diferentes clases de datos, a partir de las cuales se pueden generar casos de prueba. El objetivo de la partición equivalente es identificar casos de prueba que revelen posibles errores, lo cual permite reducir el número total de casos de prueba necesarios. (Pressman R. S., 2010).

Pruebas de aceptación

Las pruebas de aceptación se realizan en estrecha colaboración con el cliente, con el objetivo de establecer los criterios de prueba. Esta participación activa del cliente es fundamental, ya que el desarrollador del software puede enfrentar dificultades para identificar posibles fallos o resultados incorrectos por sí solo. En conjunto con el equipo de desarrollo, el cliente desempeña un papel importante en la definición de los escenarios de prueba, los cuales describen el comportamiento esperado del sistema y los resultados que se espera obtener. (Pressman R. S., "Ingeniería de Software: Un Enfoque Práctico", 2014)

Dentro del campo de la ingeniería de software, se llevan a cabo las pruebas de aceptación (PA) con el objetivo de evaluar el nivel de confianza en un sistema, sus componentes y características no funcionales. La confianza en el sistema se establece en base a su grado de cumplimiento de las necesidades, requerimientos y procesos de negocio especificados por el usuario o cliente. Es responsabilidad del usuario tomar la decisión de aceptar o no el sistema una vez que ha sido evaluado en función de estos aspectos. (Pressman R. S., "Ingeniería de Software: Un Enfoque Práctico", 2014)

“En términos generales, las pruebas de aceptación se centran en verificar que el sistema en desarrollo cumple los requisitos establecidos por el cliente y en asegurar su correcto funcionamiento en el contexto del negocio. Estas pruebas no se enfocan tanto en la detección de errores en el código, sino en cumplir con las expectativas y necesidades del cliente o su negocio.” (Pressman R. S., "Ingeniería de Software: Un Enfoque Práctico", 2014)

Tabla 6 Prueba de aceptación de la HU Escanear código QR (Elaboración propia)

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU_RF1	**Nombre Historia de Usuario:** Escanear código QR
Nombre de la persona que realiza la prueba: Danel Castro García	
Descripción de la Prueba: Consiste en evaluar el requisito Escanear código QR desde la aplicación APMAcxess.	
Condiciones de Ejecución: Debe existir un código escrito en la sección correspondiente del sistema.	
Entrada / Pasos de ejecución: Para evaluar el requisito es necesario: Se debe utilizar y dar permisos de acceso al terminal de la cámara del dispositivo móvil, luego escanear el código QR, utilizando el botón existente en la página principal, permitiendo o denegando el acceso al personal.	
Resultado Esperado: El agente de seguridad puede brindar / denegar el servicio de entrada a la entidad.	
Evaluación de la Prueba: Satisfactoria.	

Método de Caja Negra

En el contexto de las pruebas de caja negra, el enfoque se dirige hacia los requisitos funcionales del software. Este método permite obtener conjuntos de condiciones de entrada que cubren de manera exhaustiva todos los requisitos funcionales de un programa. (Desikan S. , 2006). El objetivo de las pruebas de caja negra es identificar posibles errores en las siguientes categorías:

- Funciones incorrectas o ausentes.

- Errores de interfaz.

- Errores en estructuras de datos o en accesos a bases de datos externas.

- Errores de rendimiento.

- Errores de inicialización y terminación.

En el método de caja negra, se emplea la técnica de partición equivalente para llevar a cabo las pruebas. Esta técnica implica dividir el dominio de entrada de un programa en diferentes clases de datos, las cuales permiten generar casos de prueba. El diseño de estos casos de prueba se basa en la evaluación de las clases de equivalencia de una condición de entrada. Una condición de entrada puede ser un valor numérico, un rango de valores, un conjunto de valores relacionados o una condición lógica. Cada clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada. (Srinivasan Desikan, 2006).

Para aplicar la técnica de partición equivalente, se debe realizar el diseño de casos de prueba (DCP) con el objetivo de obtener un conjunto de pruebas que tenga la mayor probabilidad de descubrir los posibles defectos del software.

Tabla 7 Caso de prueba del escenario Acceder privilegios admin (Elaboración propia)

Descripción	Variable 1	Respuesta del sistema	Flujo central
Permite acceder a los privilegios de administrador	V (Introducir valores en los campos.)	El sistema permite acceder a los privilegios	1. Seleccionar el botón admin 2. Introducir valores en el campo. 4. Seleccionar la opción Aceptar. 5. El sistema debe permitir acceder a los privilegios
	I (introducir un valor de contraseña menor a 6 dígitos.)	El sistema muestra el mensaje "No cumple con los requisitos mínimos".	

	I (Dejar campos vacíos)	El sistema muestra el mensaje "Los campos no pueden estar vacíos".	

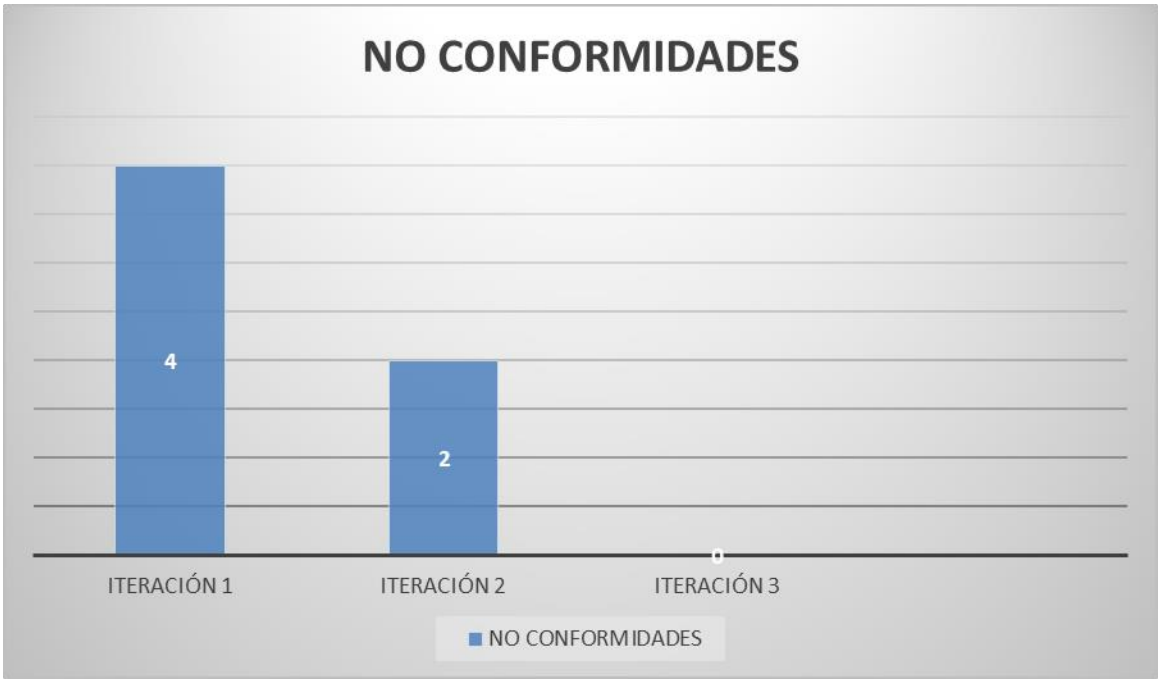


Figura 11 No conformidades del método acceder privilegios admin (Elaboración propia)

Con el objetivo de comprobar que las funcionalidades de la herramienta se realizaron correctamente y responden a las necesidades del cliente, el método se aplicó en tres iteraciones como se muestra en la Figura 11. En la primera se detectaron un total de 4 No Conformidades (NC). En la segunda iteración los resultados mejoraron al disminuir a 2 NC, ya en la tercera iteración se logró resolver las mismas obteniéndose cero NC. La imagen anterior ilustra los resultados de aplicar el método de caja negra, teniendo en cuenta las NC.

Prueba de Unidad

Las pruebas de unidad se centran en validar la unidad más pequeña dentro del diseño de software. Su enfoque se dirige hacia la lógica interna de procesamiento y las estructuras de datos, tales como el código fuente, archivos binarios y archivos de datos, entre otros elementos. Estas pruebas tienen la capacidad de aplicarse de forma simultánea a varios componentes. (Meszaros, 2007).

La ejecución de las pruebas de unidad se fundamenta en el enfoque de caja blanca o estructural, también conocido como prueba de caja de cristal. Este método de diseño de casos de prueba utiliza la estructura de control del diseño procedimental para generar los casos de prueba necesarios. En este contexto, se emplea específicamente la técnica de camino básico, que permite evaluar la complejidad lógica de un diseño procedimental y utilizar esa evaluación como guía para definir un conjunto esencial de caminos de ejecución. (Tian, 2008).

Técnica de Camino Básico

En el contexto del método de caja blanca, se utiliza la técnica de camino básico con el propósito de verificar la ejecución independiente de cada camino dentro de un componente o programa, lo cual proporciona una medida de la complejidad lógica del diseño. La idea principal radica en generar casos de prueba a partir de un conjunto de caminos independientes por los cuales puede fluir el control del programa. Para identificar estos caminos independientes, se construye un Grafo de Flujo asociado y se calcula su complejidad ciclomática. (McCabe, 1976).

Pressman sugiere una estrategia para aplicar la técnica de camino básico, que implica analizar la complejidad ciclomática de cada procedimiento que forma parte de las clases del sistema. Una vez completado este análisis, se selecciona el método que tenga un valor que indique la posible presencia de errores, además, proporciona una medida del número de pruebas necesarias para validar la implementación correcta de una función específica. (Pressman R. S., INGENIERIA DE SOFTWARE, 2010)

Luego se determina la complejidad ciclomática V(G) del grafo resultante, el cual indica el número de caminos independientes que existen en un grafo, es cualquier camino

dentro del código que introduce por lo menos un nuevo conjunto de sentencias de procesos o una nueva condición. La complejidad ciclomática se puede calcular de 3 formas:

1. V (G) = (A - N) + 2
2. V (G)=P+ 1
3. V (G) = R

Donde:

A: es la cantidad de aristas.

N: la cantidad de nodos.

P: es el número de nodos predicado contenidos en el grafo de flujo G

R: representa la cantidad de regiones en el grafo

Realizando los cálculos correspondientes se obtiene por cualquiera de las variantes el siguiente resultado:

1V (G) = (A- N) + 2

V (G) = (4 – 4) + 2

V (G) = 2

2V (G) = P + 1

V (G) = 1+1

V (G) = 2

3.V (G) = R = 2

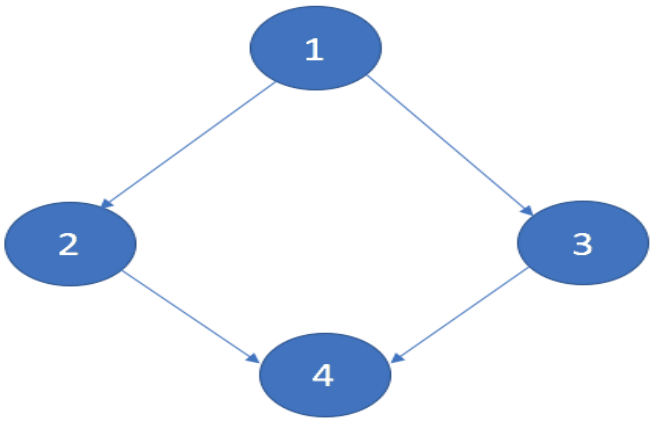


Figura 12 Grafo de flujo (Elaboración propia)

```
TextButton(
              onPressed: () {
                if (passwordController.text == widget.adminPassword) {
                  Navigator.pop(context);
                  _navigateToAdminPage(context);
                } else {
                  print('Password incorrecto');

}
```

Figura 13 Ejemplo de código utilizado para calcular la complejidad ciclomática (Elaboración propia)

La siguiente tabla muestra el análisis de la complejidad ciclomática en el software:

Tabla 8 Análisis de riesgo de la Complejidad Ciclomática (Elaboración propia)

Complejidad ciclomática.	Evaluación del riesgo.
1 – 10	Programa simple, sin mucho riesgo.
11 – 20	Más complejo, riesgo moderado.
21 – 50	Complejo, programa de alto riesgo.
50 en adelante.	Programa no testeable, muy alto riesgo.

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que la complejidad ciclomática del código es de 2. Existen 2 posibles caminos por donde el flujo puede circular, coincidiendo con el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 9 Trayectoria básica obtenida en el grafo (Elaboración propia)

No.trayectorias	Trayectorias.
1	1-2-4
2	1-3-4

A partir de la ejecución los casos de pruebas obtenidos a través de la aplicación de la técnica camino básico, se concluye que los mismos fueron probados satisfactoriamente demostrando que el código generado no se encontraron ciclos infinitos y no existe código innecesario en el sistema desarrollado

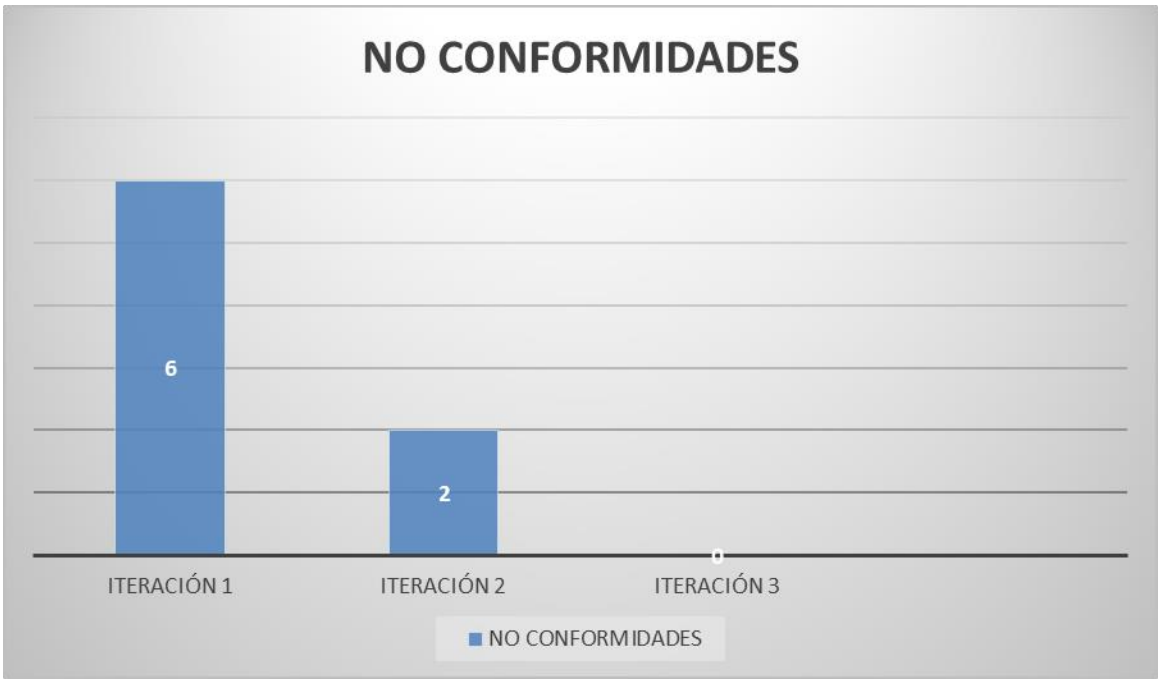


Figura 14 No conformidades en el método de caja blanca (Elaboración propia)

3.4 Conclusiones del capítulo

En este capítulo se presentaron los estándares de diseño y codificación que se aplicaron en la implementación de la aplicación para el sistema de control de acceso en la entidad XETiD. Estos estándares contribuyeron a una mejor organización y comprensión del código. A partir de la validación de los requisitos identificados, se obtuvo un listado de requisitos redactados correctamente. Además, se describieron las pruebas utilizadas para asegurar la calidad del software, como las pruebas de unidad que se llevaron a cabo utilizando la técnica del camino básico. Estas pruebas permitieron corregir errores en el código de forma independiente, lo que ayudó a reducir la complejidad de las pruebas funcionales. También se realizaron pruebas de aceptación para verificar el cumplimiento de los requisitos funcionales establecidos en las Historias de Usuario (HU).

CONCLUSIONES FINALES

- En el estudio de los referentes teóricos y el análisis de las diferentes herramientas y tendencias para el control de acceso, permitió determinar que no existe un sistema informático que satisfaga las necesidades requeridas por el cliente.

- A través del proceso de diseño de la propuesta de solución, permitió crear los elementos más relevantes siguiendo la metodología de desarrollo de software Scrum, tomando como guía el *product backlog* como referencia principal.

- Las técnicas de validación aplicadas a la propuesta de solución permitieron la detección y corrección de las no conformidades y evidenciaron que la aplicación constituye una solución funcional.

RECOMENDACIONES

- Inclusión de técnicas de reconocimiento facial y lecturas de huellas dactilares como métodos de identificación de acceso en la aplicación.

REFERENCIAS BIBLIOGRÁFICAS

Ambler, S. W. (2004). *"Introduction to UML 2 Class Diagrams" .* Cambridge, UK: Cambridge University Press.

Anderson, R. (2020). *"Flutter: A Quick-Start Guide to Programming Cross-Platform Apps with Dart and Flutter".* Berkeley, CA: Pragmatic Bookshelf.

Bass, L. C. (2012 (3ª edición)). *"Software Architecture in Practice" .* Boston, MA: Addison-Wesley.

Beizer, B. (1995). *"Black-Box Testing: Techniques for Functional Testing of Software and Systems".* Nueva York, EE. UU.: Wiley.

Bell, D. E. (1973). *"Secure Computer Systems: Mathematical Foundations and Model".* Cambridge, MA: The MIT Press.

Black, R. (2019). *"Advanced Software Testing - Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager".* Santa Barbara, California, EE. UU: Rocky Nook.

Buschmann, F. M. (1996). *"Pattern-Oriented Software Architecture: A System of Patterns".* New York, NY: John Wiley & Sons.

Castells, M. (1996). *"La era de la información: Economía, sociedad y cultura" .* Madrid : Alianza Editorial.

Cohn, M. (2004). *"User Stories Applied: For Agile Software Development".* Boston, MA: Addison-Wesley.

Connolly, T. M. (2014 (6ª edición)). *"Database Systems: A Practical Approach to Design, Implementation, and Management".* Boston, MA : Pearson.

Desikan, S. (2006). *"Software Testing: Principles and Practices".* Nueva Delhi: Pearson.

Desikan, S. y. (2006). *"Software Testing: Principles and Practice" .* No especificada: Pearson ISBN-13: 978-0201794293.

Erich Gamma, R. H. (1994). *"Design Patterns: Elements of Reusable Object-Oriented Software".* Boston, Estados Unidos ISBN: 0-201-63361-2: Addison-Wesley.

Erich Gamma, R. H. (1994). *"Design Patterns: Elements of Reusable Object-Oriented Software".* Boston: Addison-Wesley .

Erich Gamma, R. H. (1994). *"Design Patterns: Elements of Reusable Object-Oriented Software" .* no especificada: Addison-Wesley ISBN: 978-0201633610.

Erich Gamma, R. H. (1994.). *Design Patterns: Elements of Reusable Object-Oriented Software.* Boston.: Addison-Wesley.

Ferraiolo, D. S. (2001). *"Proposed NIST Standard for Role-Based Access Control".* Gaithersburg, MD: National Institute of Standards and Technology (NIST).

Finkenzeller, K. (2010). *"RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication" .* Chichester, UK : Wiley.

Flutter.dev. ([Sin especificar]). *"Effective Dart: Style Guide".* [Sin especificar]: https://dart.dev/guides/language/effective-dart/style.

Fowler, M. &. (2019 (3ª edición)). *"UML Distilled: A Brief Guide to the Standard Object Modeling Language" .* Boston, MA: Addison-Wesley Professional.

Fowler, M. (2019). *"UML Distilled: A Brief Guide to the Standard Object Modeling Language" .* [Sin especificar] : Addison-Wesley.

Fowler, M. (2019 (4ª edición)). *"UML Distilled: A Brief Guide to the Standard Object Modeling Language" .* Boston, MA: Addison-Wesley.

García, J. M. (2019). *Sistemas de Control de Acceso y Políticas de Seguridad: Tecnologías y Aplicaciones.* Madrid : Ediciones Técnicas Avanzadas.

Gause, D. C. (1989). *"Exploring Requirements: Quality Before Design".* New York, NY: Dorset House Publishing.

Gómez, R. (2019). *Gestión de acceso a laboratorios de producción: Desarrollo de una aplicación centralizada.* La Habana : Editorial ABC.

Harris, S. &. (2016). *"Cyber Security Basics: Protect your organization by applying the fundamentals" .* Sebastopol, CA : O'Reilly Media.

Jain, A. K. (2011). *Introduction to Biometrics .* New York: Springer.

Johnson, R. (2020). *Web-Based Applications for Access Control Systems: An In-depth Exploration.* New York: Wiley.

Jones, R. (2018). *RFID: Technologies and Applications for Identification and Access Control.* New York: Wiley.

Kim, J. (2019). *Biometric Access Control Systems: A Comprehensive Guide.* Berlin: Springer.

Larman, C. (2005). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design.* Upper Saddle River, Nueva Jersey: Prentice Hall.

Martin, R. C. (2008). *"Clean Code: A Handbook of Agile Software Craftsmanship".* [no especificada]: Prentice Hall ISBN-13: 978-0132350884.

Martínez. (2012).

Martínez-Ortega, J.-F. (2016). *"Seguridad y Control de Acceso: Fundamentos, Modelos y Tecnologías".* Madrid, España: RA-MA Editorial.

Mattord, M. E. (2020). *"Principles of Information Security" .* no especificado: Cengage Learning.

McCabe, T. (1976). *A Complexity Measure.* Washington, DC, USA: IEEE Computer Society.

McConnell, S. (2004). *Code Complete: A Practical Handbook of Software Construction.* Redmond, Washington: Microsoft Press .

Meszaros, G. (2007). *xUnit Test Patterns: Refactoring Test Code.* Upper Saddle River, NJ, USA: Addison-Wesley.

Momjian, B. (2001). *"PostgreSQL: Introduction and Concepts" .* Redwood City, CA: Addison-Wesley.

Myers, G. J. (2011). *"The Art of Software Testing" .* [Sin especificar] : Wiley ISBN-13: 978-0471469120.

Offutt, J. (2013). *"Introduction to Software Testing".* Nueva York, EE. UU.: Cambridge University Press.

Pérez, A. (2018). *Desarrollo de sistemas de control de acceso en la Universidad de las Ciencias Informáticas (UCI): Un enfoque en el sistema de identificación.* La Habana: Editorial UCI.

Pérez, J. (2020). *"Sistemas de Control de Acceso y Seguridad Integral en Empresas".* Madrid: Ediciones Seguridad Empresarial.

Pressman, R. S. (2010). *INGENIERIA DE SOFTWARE.* no especificada: Mcgraw-Hill.

Pressman, R. S. (2014). *"Software Engineering: A Practitioner's Approach".* Nueva Delhi: McGraw-Hill Education ISBN-13: 978-9339220929.

Pressman, R. S. (2014). *"Ingeniería de Software: Un Enfoque Práctico".* Ciudad de México: McGraw-Hill Interamericana.

Pressman, R. S. (2014). *"Software Engineering: A Practitioner's Approach".* Nueva York, Estados Unidos: McGraw-Hill Education ISBN-13: 978-0078022128.

Rodríguez, J. (2020). *Implementación de un sistema de control de acceso para el centro CISED en la UCI.* La Habana: Editorial XYZ.

Sandhu, R. C. (1996). *"Role-Based Access Control Models".* New York : IEEE Computer Society.

Schwaber, K. &. (2020). *"The Scrum Guide".* [Sin especificar] : Scrum.org.

Seacord, R. C. (2013). *"Secure Coding in C and C++".* Upper Saddle River, NJ : Addison-Wesley Professional.

Sebesta, R. W. (2015 (11ª edición)). *"Concepts of Programming Languages" .* Boston, MA: Pearson.

Shaw, M. &. (1996). *"Software Architecture: Perspectives on an Emerging Discipline".* Upper Saddle River, NJ: Prentice Hall.

Shneiderman, B. (2016 (6ª edición)). *"Designing the User Interface: Strategies for Effective Human-Computer Interaction" .* Boston, MA: Pearson.

Smith, K. (2020). *Control de Acceso y Gestión de Personal: Tecnologías y Aplicaciones.* Barcelona: Ediciones Técnicas Avanzadas.

Sommerville, I. (1997). *"Requirements Engineering: A Good Practice Guide".* Chichester, Reino Unido: Wiley.

Sommerville, I. (2015). *"Software Engineering" .* Harlow, Essex, Reino Unido : Pearson ISBN-13: 978-0133943030.

Sommerville, I. (2015 (10ª edición)). *"Software Engineering" .* Boston, MA: Pearson.

Sommerville, I. (2016). *Software Engineering.* no especificado: Pearson .

Srinivasan Desikan, G. R. (2006). *"Software Testing: Principles and Practices".* Nueva Delhi: Pearson Education.

Stutzman, J. &. (2017). *"Network Security Essentials: Applications and Standards" .* Upper Saddle River, NJ : Pearson.

Sutherland, J. (2014). *Scrum: The Art of Doing Twice the Work in Half the Time .* Nueva York : Crown Business.

Tian, J. (2008). *Software Testing: Principles and Practices.* Hoboken, NJ, USA: Wiley.

Toledano, M. A. (2019). *"Seguridad y Control de Acceso en Sistemas de Información".* no especificada: Universidad de Castilla-La Mancha.

Whitman, M. E. (2018). *"Principles of Information Security" .* Boston : Cengage Learning.

Whitman, M. E. (2020). *"Management of Information Security".* Boston : Cengage Learning.

Whitten, J. L. (2018 (8ª edición)). *"Systems Analysis and Design Methods" .* New York, NY: McGraw-Hill Education.

Yang, D. F. (2019). *"Access Control Policies: A Practitioner's Guide".* Boca Raton, FL: CRC Press.

Zhang, D. &. (2020). *"Handbook of Access Control Systems and Techniques".* Boca Raton, FL: CRC Press.

ANEXOS

xetid

Habana, 1 de noviembre de 2023

Estimado Danel Castro García.

Nos complace informarle que su solicitud de aceptación de software ha sido aprobada. En nombre de XETID nos complace aceptar su software Aplicación de identificación y control de acceso para dispositivos móviles para su uso en nuestra organización, expresamos nuestra satisfacción con respecto al software que se ha implementado recientemente en nuestra empresa.

Atentamente, Daniel Meriño Tamayo

Cargo: Jefe de Proyecto

Firma:

XETID

Calle 206A, e/ 217 y 203,
municipio Boyeros, La Habana
(+537) 837 3600 / (+537) 837 3611
xetid@xetid.cu
www.xetid.cu

Figura 15 Carta de aceptación del cliente.

Tabla 10 Historia de usuario RF3. Fuente: Elaboración propia

	Historia de Usuario
Número:3	**Usuario:** Guardia de seguridad
Nombre de Historia: Controlar acceso a la empresa	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 0.5	**Iteración Asignada:**2
Programador Responsable: Danel Castro García	
Descripción: El guardia de seguridad desea verificar si la persona puede acceder o no a la entidad.	
Validación: El Guardia de seguridad permite o deniega el acceso de la persona.	

Tabla 11 Historia de usuario RF4. Fuente: Elaboración propia

	Historia de Usuario
Número:4	**Usuario:** Guardia de seguridad
Nombre de Historia: Registrar evento de acceso	
Prioridad en Negocio: Media	**Riesgo en Desarrollo:** Medio
Puntos estimados: 0.5	**Iteración Asignada:**2
Programador Responsable: Danel Castro García	
Descripción: El guardia de seguridad desea revisar el registro de acceso de las personas.	
Validación: El Guardia de seguridad puede revisar el registro de acceso	