Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en ingeniería del software

Sistema NFC para el control de accesos

Alumno:
D. Miguel Ángel García Sanz

Tutores:
D. Diego García Álvarez
D. José Baro Gonzalez

...

Agradecimientos

Gracias a mi tutor, Diego, por aguantarme dia tras dia en la realización de este documento.

Gracias a mi tutor, Pepe, por guiarme durante esta aventura con el desarrollo del TFG.

Gracias a todos mis amigos por todo su infinito apoyo, ayuda y estrés compartido.

Gracias a mi empresa, *Brooktec,* por haberme brindado la posibilidad de realizar y sacar adelante este proyecto.

Resumen

Este Trabajo de Fin de Grado de la titulación de Ingeniería Informática de la Universidad de Valladolid se centra en el desarrollo de un sistema de registro de entradas y salidas de usuarios utilizando la tecnología *Near Field Communication* (NFC). El objetivo principal es crear un sistema eficiente y seguro que permita a los usuarios acceder a espacios o servicios específicos mediante la identificación por radiofrecuencia en un entorno laboral.

El sistema propuesto debe cumplir diversas funcionalidades siendo estas permitir al empleado guardar un registro de las entradas y salidas al entorno de trabajo simplemente acercando su identificador NFC al lector. Además, puede consultar todos estos registros a través de una simple interfaz a la que accederá con unas credenciales personales. Por último, existe uno o más empleados con el rol de administrador, capaces de generar nuevos usuarios en el sistema guardando sus identificaciones personales y con la posibilidad de revisar todos los registros realizados por todos los empleados.

Este proyecto viene propuesto por parte de la empresa Brooktec. Brooktec es una asesoría de software ubicada en Valladolid pero con oficinas en Madrid y Jaén, especializada en la realización y mantenimiento de páginas web. La empresa desea actualizar su actual sistema de registros en la oficina con este proyecto.

El enfoque del proyecto se centrará en el desarrollo de un sistema robusto y escalable que pueda ser integrado en diferentes entornos. Se realizará un análisis y explicación detallado de las tecnologías NFC disponibles. El TFG cubrirá desde el diseño conceptual hasta la implementación y las pruebas del sistema. Se evaluará la viabilidad técnica y económica, considerando factores como el costo de desarrollo, implementación y mantenimiento.

Abstract

This Final Degree Project of the Computer Engineering Degree of the University of Valladolid is focused on the development of a system for the registration of user inputs and outputs using Near Field Communication (NFC) technology. The main objective is to create an efficient and secure system that allows users to access specific spaces or services through radio frequency identification in a work environment.

The proposed system must fulfill several functionalities being to allow the employee to keep a record of the entrances and exits to the work environment by simply bringing his NFC identifier close to the reader. In addition, he can consult all these records through a simple interface that he will access with personal credentials. Finally, there is one or more employees with the role of administrator, able to generate new users in the system by saving their personal IDs and with the possibility to review all the records made by all employees.

This project is proposed by the company Brooktec. Brooktec is a software consultancy located in Valladolid but with offices in Madrid and Jaén, specialized in the development and maintenance of web pages. The company wishes to update its current office registration system with this project.

The focus of the project will be on the development of a robust and scalable system that can be integrated in different environments. A detailed analysis and explanation of the available NFC technologies will be performed. The TFG will cover from conceptual design to implementation and testing of the system. Technical and economic feasibility will be evaluated, considering factors such as development, implementation and maintenance cost.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

Este proyecto parte de la necesidad de la empresa Brooktec de crear un sistema que permita registrar las horas de acceso a la oficina. Para ello se quiere implementar un sistema de identificación de usuarios mediante el uso de tarjetas o tokens NFC.

Esta propuesta viene ofrecida por parte de la empresa Brooktec para realizar el Trabajo de Fin de Grado de la titulación Grado en Ingeniería Informática de la Escuela de Ingeniería Informática de la Universidad de Valladolid así realizando en su totalidad esta asignatura que consta de la realización de un proyecto informático en el que se aplican los conocimientos y habilidades adquiridos del alumno durante su aprendizaje en la titulación y mención de Ingeniería del Software.

1.2. Motivación

La motivación del proyecto surge de la necesidad de una actualización del actual sistema de la empresa. Actualmente este registro se realiza desde una página web en la que el usuario inicia sesión con su cuenta personal y registra su llegada/salida.

El actual sistema tiene un gran problema de imprecisión, ya que los empleados para realizar este registro necesitan, mediante su ordenador personal, entrar en el sistema y realizar el *check in/check out*, creando un registro impreciso debido a que las horas de entrada y salida de la oficina no coinciden con exactitud con las registradas. Con este nuevo sistema se tendrían unos registros más precisos de estos datos.

1.3. Objetivos

El principal objetivo que se persigue con el TFG es el desarrollo de un sistema de registro de entradas y salidas de usuarios mediante NFC que cumpla con las necesidades y especificaciones de la empresa. Para lograrlo, se espera que el proyecto contemple las siguientes tareas o subobjetivos:

1. Desarrollo del registro de identificación mediante NFC para realizar el *check-in/check-out* en el sistema, permitiendo a los usuarios acceder de manera rápida y eficiente a los espacios o servicios específicos.
2. Adaptación y configuración de una *Raspberry Pi* junto con un adaptador NFC permitiendo la conexión entre el dispositivo y el servidor.
3. Creación de la consulta del historial de check-in/check-out por parte de los usuarios, brindando transparencia y control sobre el uso del sistema.
4. Implementación de un sistema de autenticación mediante *JSON Web Token (JWT)* entre dispositivo y servidor para proteger el sistema de accesos de terceros, asegurando la integridad y la confidencialidad de la información.

1.4. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: El presente capítulo.

Capítulo 2 Planificación: En este capítulo se describirá la planificación y desarrollo del proyecto.

Capítulo 3 Estado del tema: Tecnologías NFC: En este capítulo se expondrán las tecnologías NFC en la actualidad.

Capítulo 4 Requisitos: En este capítulo se describirán los diferentes requisitos y casos de usos planteados para el sistema

Capítulo 5 Análisis: En este capítulo se realizará un una explicación del análisis del proyecto, explicando los modelos de dominio, análisis y los casos de uso.

Capítulo 6 Diseño: En este capítulo se comentarán los diferentes patrones arquitectónicos y de diseño que deberá seguir el proyecto.

Capítulo 7 Tecnologías utilizadas: En este capítulo se comentarán las diferentes tecnologías que se han utilizado para desarrollar el TFG

Capítulo 8 Implementación y pruebas: En este capítulo se describirán las pruebas realizadas al sistema y un manual de despliegue para futuro desarrollo.

Capítulo 9 Conclusiones y Trabajo Futuro: En este capítulo se comentarán las conclusiones y resultados obtenidos al completar el desarrollo del proyecto así cómo los diferentes aspectos de mejoría propuestos para éste en el futuro.

Capítulo 2

Planificación

En este capítulo se presenta la planificación inicial del proyecto. En él se describen las pautas para su ejecución. Además, se presentan los diferentes plazos y entregas previstos, así como la metodología de trabajo a seguir para minimizar los riesgos.

2.1. SCRUM

Ese proyecto se ha desarrollado siguiendo el marco de trabajo Scrum. Scrum se basa en los principios y valores ágiles del desarrollo de software basado en la distribución de pequeñas tareas dentro de un marco de tiempo llamado *sprint*.

El desarrollo ágil de software se basa en el manifiesto ágil [3], publicado en el 2001, es un conjunto de valores y principios creado por un grupo de expertos en desarrollo software. Este manifiesto establece cuatro valores y doce principios fundamentales que buscan mejorar la forma en la que se desarrollan y entregan los proyectos de software.

Scrum proporciona un proceso de desarrollo basado en un enfoque iterativo e incremental para la entrega de un producto final. Consiste en el trabajo conjunto de diferentes desarrolladores que intentan lograr objetivos en cortos periodos de tiempo llamados *sprints*. Estos objetivos a corto plazo se ven reflejados en el *product backlog*, una lista ordenada de funcionalidades, características, mejoras y arreglos que el equipo de desarrollo debe completar para lograr los objetivos del proyecto. En otras palabras, es una lista de tareas que se deben realizar para crear y mantener el producto o servicio que se está desarrollando.

En los equipos scrum se diferencian tres roles principales:

Product Owner: Es el responsable de definir y priorizar el *product backlog*, es decir, la lista de tareas y funcionalidades del proyecto. El *product owner* es quien decide qué funcionalidades deben ser incluidas en el producto final, qué características son más importantes y cuál es el orden en que deben ser entregadas.

Scrum Master: Es el responsable de asegurar que el equipo de Scrum siga las prácticas y principios de la metodología. El *scrum master* ayuda al equipo a ser más productivo y eficiente, eliminando los obstáculos que puedan surgir y facilitando la colaboración entre los miembros del equipo. Además, el *scrum master* coordina las reuniones de Scrum y ayuda al equipo a realizar la retrospectiva al final de cada *sprint*.

Equipo de Desarrollo: Es el grupo de personas que trabaja en la implementación de las tareas definidas en el *product backlog*. El equipo de desarrollo es autoorganizado y multidisciplinario, lo que significa que cada miembro tiene habilidades y responsabilidades diferentes, pero trabajan juntos para lograr los objetivos del proyecto. Durante el *sprint*, el equipo de desarrollo se encarga de seleccionar las tareas del *product backlog* que se completarán en el *sprint* y de implementarlas de manera eficiente.

Otra característica principal de scrum son sus eventos. Estos eventos se focalizan en minimizar la necesidad de realizar reuniones adicionales que no estuvieran asimiladas en la planificación inicial. Los principales eventos son los siguientes:

Sprint: Como se ha comentado previamente, son los diferentes periodos de tiempo preestablecidos, comúnmente de duración entre 1 y 4 semanas, en los que se van a ir organizando las diferentes actividades durante todo el proyecto. Durante estos *sprints* no se pueden modificar los requisitos.

Sprint Planning: Evento realizado siempre al principio de cada *sprint* en el que se hablará sobre los diferentes objetivos que se proponen realizar en ese *sprint*. En esta reunión participarán todos los miembros del equipo, en la que se seleccionarán las diferentes tareas encasilladas dentro del *product backlog* que se realizará. Para seleccionar estas tareas se puede realizar un *planning poker*, que consiste en la estimación del tiempo de desarrollo que se cree necesario para su realización y, junto a las diferentes prioridades que tienen cada tarea, se generará el *sprint backlog*.

Daily Scrum: Son reuniones diarias de corta duración (5-20 minutos) en la que todos los miembros del equipo informan del progreso realizado entre reuniones. Se busca que en estas reuniones se obtenga una buena comunicación y coordinación del equipo.

Sprint Review: Evento realizado siempre al finalizar un *sprint*, en el cual todo el equipo y el cliente discutirán sobre el avance del proyecto focalizando lo desarrollado dentro del propio *sprint*. En esta reunión se pueden realizar modificaciones en el *product backlog* si fueran pertinentes.

Sprint Retrospective: Evento realizado al finalizar un *sprint* en el que se conversara sobre las conclusiones obtenidas dentro del *sprint*. Esta reunión sirve de referencia para los próximos *sprint planning* que se realicen.

2.1.1. Adaptación de la metodología al proyecto

Se ha decidido utilizar el marco de trabajo Scrum como marco de planificación del proyecto debido a diferentes factores. Primero, este proyecto nace de una idea no prioritaria de

implementación para la empresa. Esto resulta en una vaga especificación de los requisitos que se quieren conseguir en la inicialización del trabajo. Para solucionar este problema, Scrum ofrece una gran flexibilidad en la adaptación y creación de requisitos cuando el proyecto está iniciado. Por otra parte, al ser un proyecto de empresa, mi tutor desea realizar un seguimiento constante para poder ofrecer una retroalimentación adecuada. A mayores, la propia empresa utiliza ya de por sí metodologías ágiles, por lo que me pueden auxiliar en caso de necesitarlo.

Debido a que el proyecto va a ser desarrollado por solo una persona habría que adaptar esta metodología. Para ello, el estudiante será asignado con todos los roles principales (*product owner*, *scrum master* y desarrollador) Los *sprints* tendrán una duración de una semana. Debido al poco tiempo disponible que tiene el tutor, las reuniones de *sprint planning*, *sprint review* y *sprint retroactive* serán realizadas por el propio alumno y las reuniones de *daily scrum* pasarán a ser semanales intentando ajustar horarios entre el tutor y el alumno en las que se le ofrecerá al alumno una revisión del estado del proyecto.

2.2. Planificación inicial

Se ha creado un calendario con los diferentes *sprints* planificados antes de la finalización del proyecto. Según la guía docente, los trabajos de fin de grado han de tener una duración aproximada de 300 horas totales. Teniendo en cuenta el día en el que se inició el proyecto (11 de marzo) y el día en el que se planea finalizarlo(4 de junio), se finaliza con un total de 12 *sprints* de una semana cada uno. La cantidad de horas que se plantea para cada *sprint* es variable debido a motivos académicos y de trabajo. Se pueden considerar un extra de 4 *sprints* más si fueran necesarios para finalizar el proyecto. Esta división establecería entre 30 - 40 horas para cada *sprint*.

2.3. Planificación de riesgos

Antes de iniciar con el proyecto, es vital realizar un estudio exhaustivo de los posibles riesgos que el planning inicial puede generar, ya que, al no tener experiencia previa, esta planificación se basa en suposiciones. Por ello, es necesario saber localizar y tratar los riesgos más destacados que puedan haber.

Un riesgo es una posible situación o evento que podría afectar negativamente al éxito del proyecto. Un riesgo puede ser cualquier cosa que tenga el potencial de causar problemas, retrasos, costos adicionales o incluso la falla total del proyecto.

Para la identificación y tratamiento de los riesgos se les analizará cuatro principales elementos:

Probabilidad: Se trata de la posibilidad de que un riesgo ocurra. Su rango de valores son: baja, media y alta.

Impacto: El impacto de un riesgo se refiere a la medida en que ese riesgo podría afectar negativamente el éxito del proyecto. En otras palabras, el impacto es la magnitud de las consecuencias que el riesgo puede tener en el proyecto. Su rango de valores son: bajo, medio y alto.

Plan de mitigación: Un plan de mitigación es un conjunto de acciones y estrategias diseñadas para reducir la probabilidad y/o el impacto de un riesgo específico en un proyecto. Este plan se desarrolla para prepararse para situaciones imprevistas que puedan presentarse durante el desarrollo del proyecto.

Plan de contingencia: Un plan de contingencia es un conjunto de medidas o acciones preparadas de antemano para afrontar un riesgo que se ha materializado.

Analizando estos elementos para cada riesgo se podría deducir cuánta importancia habría que dar a cada uno de ellos y qué conjunto de acciones iniciales se podrían tomar para reducir su probabilidad o minimizar su impacto.

A continuación se expondrán los diferentes riesgos que se han tenido en cuenta para el desarrollo específico de este proyecto.

RSK001	
Nombre	Enfermedad.
Descripción	Mientras se realiza el TFG, el alumno se enferma impidiendo trabajar correctamente.
Categoría	Personal
Vulnerabilidad	Caer enfermo impide trabajar en plenas condiciones.
Amenaza	Al no poder trabajar correctamente, los plazos del TFG pueden ser retrasados.
Probabilidad	BAJA
Impacto	BAJO
Riesgo Total	BAJO
Estado del Riesgo	Abierto
Acciones de mitigación	1. Realizar ejercicio diariamente y comer saludable. 2. Comprar las medicinas más comunes.
Acciones correctivas	1. Ir al médico. 2. Guardar reposo para superar la enfermedad lo más rápido posible.

Tabla 2.1: Riesgo RSK001: Enfermedad.

RSK002	
Nombre	Profesor inaccesible.
Descripción	El profesor es incapaz de guiar el TFG por alguna razón (Ej: viaje, baja...).
Categoría	Académico
Vulnerabilidad	El profesor no puede ayudarnos con el TFG, lo que retrasará su evolución.
Amenaza	Al enfrentar el TFG sin tutorización, se tardará más tiempo en terminarlo.
Probabilidad	BAJA
Impacto	MEDIO
Riesgo Total	BAJO
Estado del Riesgo	Aceptado
Acciones de mitigación	Realizar todas las dudas necesarias al profesor con la mayor antelación posible.
Acciones correctivas	Intentar encontrar un nuevo profesor que pueda asistirme.

Tabla 2.2: Riesgo RSK002: Profesor inaccesible.

RSK003	
Nombre	Vacaciones.
Descripción	El alumno realiza un viaje de gran longitud, lo que le impide avanzar en el TFG.
Categoría	Personal
Vulnerabilidad	Estando de vacaciones, no se podría trabajar en el TFG.
Amenaza	Al no poder trabajar en el TFG, se tardará más en terminarlo.
Probabilidad	MEDIA
Impacto	BAJO
Riesgo Total	BAJO
Estado del Riesgo	Aceptado
Acciones de mitigación	Planear el viaje con antelación y adelantar trabajo en el TFG para que no afecte a los plazos fijados.
Acciones correctivas	A la vuelta del viaje, continuar intensamente con el TFG para paliar el retraso.

Tabla 2.3: Riesgo RSK003: Vacaciones.

RSK004	
Nombre	Uso de tecnología en la que no estoy formado.
Descripción	Durante el desarrollo del TFG, surge la necesidad de utilizar una tecnología con la cual no se tiene experiencia previa.
Categoría	Software
Vulnerabilidad	No conocer la tecnología que se está utilizando ralentiza el avance del TFG.
Amenaza	No conocer la tecnología utilizada retrasará los plazos del TFG.
Probabilidad	ALTA
Impacto	MEDIO
Riesgo Total	ALTO
Estado del Riesgo	Abierto
Acciones de mitigación	Analizar previamente, antes de comenzar el TFG, qué tecnologías serán necesarias para su desarrollo y estudiar su implementación.
Acciones correctivas	Buscar alternativas a la tecnología en cuestión que cumplan con los requisitos del proyecto.

Tabla 2.4: Riesgo RSK004: Uso de tecnología en la que no estoy formado.

RSK005	
Nombre	Cambio de TFG.
Descripción	Después de un tiempo desarrollando el TFG, se decide cambiar el tema por otro diferente.
Categoría	Personal
Vulnerabilidad	Al cambiar de tema de TFG, se requerirá reiniciar el trabajo desde cero.
Amenaza	Volver a empezar el TFG implicará un retraso en su finalización.
Probabilidad	BAJA
Impacto	ALTO
Riesgo Total	MEDIO
Estado del Riesgo	Abierto
Acciones de mitigación	Asegurarse previamente al inicio del TFG de la viabilidad del nuevo tema.
Acciones correctivas	1. Mejorar la planificación del nuevo TFG teniendo en cuenta la experiencia obtenida en el anterior. 2. Reutilizar la información y el material obtenidos en el anterior TFG.

Tabla 2.5: Riesgo RSK005: Cambio de TFG.

RSK006	
Nombre	Falta de tiempo.
Descripción	No se dispone del suficiente tiempo para seguir la planificación establecida.
Categoría	Personal
Vulnerabilidad	No se dispone de tiempo suficiente para cumplir con las tareas planificadas.
Amenaza	No seguir la planificación puede provocar retrasos en las entregas y una mayor carga de trabajo en los siguientes días
Probabilidad	MEDIA
Impacto	MEDIO
Riesgo Total	MEDIO
Estado del Riesgo	Mitigándose
Acciones de mitigación	1. Planificar preventivamente. 2. Tener una planificación auxiliar. 3. Modificar la planificación según se prevea una falta de disponibilidad. 4. Utilizar metodologías ágiles.
Acciones correctivas	1. Compensar con tiempo extra de trabajo. 2. Replanificar.

Tabla 2.6: Riesgo RSK006: Falta de tiempo.

RSK007	
Nombre	Trabajo de otras asignaturas.
Descripción	Se compagina con el trabajo de otra/s asignaturas, lo que genera una gran carga de trabajo.
Categoría	Personal
Vulnerabilidad	Tener una gran carga de trabajo y falta de tiempo para el proyecto.
Amenaza	Tener una gran carga de trabajo puede afectar la planificación del proyecto y generar retrasos.
Probabilidad	MEDIA
Impacto	MEDIO
Riesgo Total	MEDIO
Estado del Riesgo	Mitigándose
Acciones de mitigación	1. Planificación concisa del trabajo. 2. Utilizar metodologías ágiles.
Acciones correctivas	1. Dar prioridad a las tareas según su importancia. 2. Replanificar.

Tabla 2.7: Riesgo RSK007: Trabajo de otras asignaturas.

RSK008	
Nombre	Imposibilidad de terminarlo.
Descripción	A causa de circunstancia mayor, es posible que no se pueda presentar el TFG a tiempo, lo que afectaría la planificación establecida para los tiempos de entrega calculados.
Categoría	Planificación
Vulnerabilidad	Tiempos de entrega calculados para dedicar el tiempo justo.
Amenaza	Al tener menos tiempo del esperado, puede que no se pueda presentar el TFG dentro del plazo estipulado.
Probabilidad	BAJA
Impacto	ALTO
Riesgo Total	MEDIO
Estado del Riesgo	Abierto
Acciones de mitigación	1. Asignar más tiempo del necesario para realizar el trabajo. 2. Notificar a la universidad de posibles retrasos para intentar reestructurar la entrega.
Acciones correctivas	Presentar el TFG en otra convocatoria, si fuera posible.

Tabla 2.8: Riesgo RSK009: Imposibilidad de terminarlo.

RSK009	
Nombre	Cambiar de tutor.
Descripción	Debido a la indisponibilidad del tutor del TFG por motivos personales o incompatibilidad alumno-profesor, es necesario realizar un cambio de tutor.
Categoría	Personal
Vulnerabilidad	TFG ligado a un único tutor, pudiendo fallar
Amenaza	Al tener un solo tutor asignado, en caso de que este falte, el TFG queda comprometido hasta que se asigne otro profesor
Probabilidad	BAJA
Impacto	ALTO
Riesgo Total	MEDIO
Estado del Riesgo	Abierto
Acciones de mitigación	1. Buscar un tutor con buena reputación. 2. Tener un tutor alternativo planeado en caso de fallo del tutor principal.
Acciones correctivas	1. Buscar otro profesor para el TFG. 2. En caso de que el tutor principal vuelva a estar disponible, replanificar teniendo en cuenta el tiempo de retraso.

Tabla 2.9: Riesgo RSK010: Cambiar de tutor.

RSK010	
Nombre	Cambiar de tema.
Descripción	Cambio de tema a mitad de cuatrimestre en el TFG.
Categoría	Personal
Vulnerabilidad	Proceso de aceptación, validación, etc. del tema del TFG junto a un corto periodo de prueba.
Amenaza	Si se cambia de tema a mitad de cuatrimestre puede que no dé tiempo a presentar el TFG dentro del plazo establecido o puede que no se encuentre un nuevo tema aceptado.
Probabilidad	MEDIA
Impacto	ALTO
Riesgo Total	ALTO
Estado del Riesgo	Abierto
Acciones de mitigación	1. Tener varios temas pensados para agilizar el proceso de aceptación.
Acciones correctivas	1. Plantear la vuelta al tema anterior. 2. Utilizar un tiempo de entrega del TFG posterior.

Tabla 2.10: Riesgo RSK011: Cambiar de tema.

RSK011	
Nombre	Tema no admitido.
Descripción	Que el tribunal/profesor no acepte el tema propuesto.
Categoría	Personal
Vulnerabilidad	Necesidad de que el tema sea aceptado por el tutor.
Amenaza	Si ningún profesor acepta el tema propuesto o el tribunal no lo considera válido, puede retrasar el tiempo de entrega.
Probabilidad	MEDIA
Impacto	MEDIO
Riesgo Total	MEDIO
Estado del Riesgo	Abierto
Acciones de mitigación	1. Tener varios temas preparados/peticiones de TFGs.
Acciones correctivas	1. Plantear otro tema lo antes posible. 2. Buscar un profesor que acepte el tema o modificarlo para su aceptación.

Tabla 2.11: Riesgo RSK012: Tema no admitido.

2.4. Plan de presupuestos

El plan de presupuestos es una parte crítica de cualquier proyecto de software, ya que establece los recursos financieros necesarios para llevar a cabo el proyecto con éxito. En este apartado se detallan los costos estimados para el proyecto y se establece un presupuesto que servirá como guía para la gestión financiera del proyecto. El plan de presupuestos debe ser realista y preciso para asegurar que el proyecto se mantenga dentro de los límites financieros establecidos y se logren los objetivos establecidos en el proyecto. Al ser un proyecto de final de carrera, se ignorará el costo del esfuerzo trabajado por el estudiante.

Los primeros gastos más relevantes a tener en cuenta son los dispositivos hardware que se requerirán. Para este proyecto, se necesitará el uso de un kit de Raspberry Pi 3 A+ [4] con un coste en el momento de la compra de 66€ y un adaptador NFC [5] con un coste de 23,95€.

Además, habría que analizar que licencias software serían necesarias. En este caso, se prevé el uso de Astah Professional [6] que a pesar de tener un coste de 40 dolares anuales no contemplaría un coste alguno ya que la licencia para estudiantes es gratuitas.

También habría que estudiar los costos de infraestructura, en los cuales se incluirían la electricidad, un macbook pro consume alrededor de 100€ anuales. Teniendo en cuenta que el periodo de desarrollo planificado del proyecto dura aproximadamente 3 meses, podemos establecer un costo aproximativo de 30 € incluyendo otros gastos como por ejemplo la electricidad [7] y el alquiler del espacio físico en el que se realizaría el proyecto. Actualmente, la empresa está alojada en una oficina ubicada en pleno centro de valladolid, por lo que podemos asumir un costo aproximado de 900€/mes [8] Finalmente, se agregará un costo de contingencia para cubrir los gastos no asimilados inicialmente, que en este caso sería un 10 % del coste total previsto.

En conclusión, se estimaría que el presupuesto simulado sería de aproximadamente 3150€ sin contar con el salario que debería cobrar el alumno como programador. En cambio, el presupuesto real por parte del alumno sería 0€ debido a que, al ser un TFG de empresa, ellos correrían con todos los gastos previamente mencionados.

Capítulo 3

Estado del Tema: Tecnologías NFC

En este capítulo se abordarán en detalle las tecnologías *Near Field Communication (NFC)* en el contexto del desarrollo de un sistema de entradas y salidas de usuarios. Se explorarán los fundamentos y principios detrás de NFC, así como su funcionamiento y aplicaciones en diversos campos. Comprender el funcionamiento de las tecnologías NFC es crucial para desarrollar un sistema efectivo y seguro basado en esta tecnología de identificación por radiofrecuencia.

3.1. Introducción a las tecnologías NFC

NFC es una forma de comunicación inalámbrica de corto alcance que se basa en la tecnología de identificación por radiofrecuencia (RFID) y opera en la banda de frecuencia de 13.56 MHz. Estas tecnologías han ganado popularidad en la actualidad permitiendo la transferencia segura de datos e información entre dispositivos electrónicos cercanos.

Los NFC destacan por su capacidad para ofrecer una conexión simple y rápida entre dispositivos compatibles. Esta tecnología permite a los usuarios intercambiar datos, realizar pagos móviles, acceder a servicios y mucho más, todo con solo tocar o acercar los dispositivos[9].

3.2. Componentes de NFC

En un sistema NFC, hay hasta tres componentes principales:

- **Etiquetas NFC o tags**: Las etiquetas NFC, también conocidas como tags o transpondedores, son pequeños dispositivos pasivos que almacenan y transmiten datos a través de campos electromagnéticos.

- **Lectores NFC**: Los lectores NFC, presentes en dispositivos como teléfonos inteligentes, tabletas o lectores dedicados, son responsables de leer y escribir datos en las etiquetas NFC.
- **Dispositivos habilitados con NFC**: Los dispositivos habilitados con NFC, como teléfonos móviles, tarjetas de pago y otros dispositivos electrónicos, pueden actuar tanto como lectores como etiquetas NFC.

3.3. Modos de operación de NFC

NFC admite diferentes modos de operación que amplían su funcionalidad y permiten una variedad de aplicaciones. Estos modos permiten la comunicación entre dispositivos NFC y definen cómo se intercambian los datos. Los dos modos de operación principales en NFC son el modo de lectura/escritura y el modo punto a punto (peer-to-peer) [10].

- **El modo de lectura/escritura** es uno de los modos más comunes de NFC. En este modo, un dispositivo habilitado con NFC, como un teléfono móvil o una tableta, actúa como el lector y se comunica con una etiqueta NFC pasiva, también conocida como tag. El lector NFC envía una solicitud al tag y recibe la respuesta correspondiente. Esta comunicación puede implicar la lectura de información almacenada en la etiqueta, como un identificador único (UID), o la escritura de datos en la memoria de la etiqueta.
- **El modo punto a punto** permite la comunicación bidireccional entre dos dispositivos habilitados con NFC. En este modo, ambos dispositivos pueden actuar como lectores y etiquetas NFC simultáneamente. Esto permite el intercambio de datos, como contactos, fotos o archivos entre los dispositivos. Además, el modo punto a punto es utilizado en aplicaciones de pago móvil, donde un dispositivo se comporta como una tarjeta de pago y se comunica directamente con un terminal de pago habilitado con NFC.
- Otro modo de operación menos común en NFC es **el modo de emulación de tarjeta**. En este modo, un dispositivo habilitado con NFC puede emular una tarjeta inteligente sin contacto. Esto permite que el dispositivo sea utilizado para realizar transacciones de pago o para acceder a sistemas de seguridad que normalmente requerirían una tarjeta física. El dispositivo emula la funcionalidad de una tarjeta, permitiendo que se realicen transacciones mediante la comunicación con un lector NFC.

Cada modo de operación de NFC tiene sus propias características y aplicaciones específicas. El modo de lectura/escritura se utiliza en aplicaciones de identificación y acceso, como tarjetas de transporte o control de acceso a edificios. El modo punto a punto se utiliza para el intercambio de datos entre dispositivos, como compartir contactos o archivos entre teléfonos móviles. El modo de emulación de tarjeta se aplica en sistemas de pago móvil y acceso seguro, donde el dispositivo habilitado con NFC reemplaza a una tarjeta física.

3.4. Características clave de NFC

El *Near Field Communication (NFC)* posee una serie de características clave que lo hacen una tecnología versátil y confiable para una amplia gama de aplicaciones. Estas características contribuyen a su popularidad y adopción en diversos campos. A continuación, se detallan algunas de las características más destacadas de NFC [11]:

- **Alcance corto**: NFC se caracteriza por tener un alcance extremadamente corto, generalmente de solo unos pocos centímetros. Esto implica que los dispositivos deben estar muy próximos entre sí para establecer una conexión NFC. Esta limitación en el alcance proporciona un nivel adicional de seguridad, ya que evita que la comunicación se intercepte fácilmente y asegura que los dispositivos estén en proximidad directa.
- **Interacción táctil**: La interacción táctil es una característica esencial de NFC. Los dispositivos NFC deben estar físicamente en contacto o muy cerca uno del otro para que la comunicación se inicie. Esta interacción táctil crea una experiencia de usuario intuitiva y facilita la transferencia de datos o la realización de transacciones con solo tocar o acercar los dispositivos.
- **Baja potencia**: NFC es una tecnología de baja potencia, lo que significa que no requiere una fuente de alimentación externa para funcionar. Esto es especialmente importante en dispositivos móviles, como teléfonos inteligentes o wearables, donde la eficiencia energética es fundamental. El bajo consumo de energía permite que los dispositivos NFC operen durante períodos prolongados sin agotar rápidamente la batería.
- **Compatibilidad**: NFC es compatible con una amplia variedad de dispositivos electrónicos, desde teléfonos móviles y tabletas, hasta tarjetas de crédito y sistemas de control de acceso. Muchos dispositivos modernos vienen equipados con la capacidad NFC integrada, lo que facilita su uso en diversas aplicaciones. Además, NFC también es compatible con otros estándares de comunicación inalámbrica, como Bluetooth y Wi-Fi, lo que permite una mayor interoperabilidad y flexibilidad en las conexiones.
- **Seguridad**: La seguridad es una preocupación primordial en cualquier tecnología de comunicación. NFC ofrece características de seguridad robustas que protegen la privacidad y autenticidad de las transacciones. La corta distancia de alcance limita el riesgo de que terceros intercepten la comunicación. Además, se pueden implementar medidas de encriptación para proteger los datos transferidos a través de NFC, brindando un nivel adicional de seguridad en aplicaciones como pagos móviles y acceso seguro a información confidencial.

Estas características hacen de NFC una tecnología confiable y eficiente para una amplia gama de aplicaciones en diferentes sectores, como pagos móviles, transporte, identificación y acceso seguro, intercambio de contenido o seguimiento de productos entre otras. La combinación de su corto alcance, interacción táctil, baja potencia, compatibilidad y seguridad robusta ha contribuido al éxito y adopción de NFC en numerosos dispositivos y servicios utilizados en nuestra vida cotidiana.

3.5. Aplicaciones de los NFC

La tecnología Near Field Communication (NFC) ha encontrado una amplia gama de aplicaciones en diversos sectores [12]. Su capacidad para establecer conexiones rápidas y seguras a corta distancia ha abierto un abanico de posibilidades en términos de intercambio de datos, pagos móviles, acceso seguro y más. A continuación, se exploran algunas de las aplicaciones más destacadas de NFC 3.1:

- **Pagos móviles**: NFC ha revolucionado la forma en que realizamos transacciones financieras. Con solo tocar o acercar un dispositivo NFC a un terminal de pago habilitado, podemos realizar pagos sin necesidad de llevar efectivo o tarjetas físicas. Servicios de pago móvil como Apple Pay, Google Pay y Samsung Pay utilizan la tecnología NFC para realizar pagos seguros y convenientes.
- **Transporte**: Muchos sistemas de transporte público han adoptado NFC para agilizar el proceso de pago y acceso. Los usuarios pueden utilizar dispositivos NFC para pagar boletos de autobús, tren o metro simplemente acercándolos a los validadores. Esto mejora la experiencia de los pasajeros al eliminar la necesidad de llevar efectivo o buscar tarjetas específicas, agilizando el flujo de pasajeros y reduciendo el uso de papel.
- **Control de acceso**: NFC se utiliza en sistemas de control de acceso para permitir la entrada segura a edificios, oficinas o áreas restringidas. En lugar de utilizar llaves físicas o tarjetas de acceso, los usuarios pueden utilizar dispositivos NFC para autenticarse y desbloquear puertas o torniquetes. Esto proporciona mayor comodidad y flexibilidad, al tiempo que mejora la seguridad y simplifica la administración de los derechos de acceso.
- **Intercambio de contenido**: NFC permite el intercambio instantáneo de contenido entre dispositivos compatibles. Por ejemplo, se pueden compartir información de contacto, imágenes, videos o documentos simplemente tocando dos dispositivos NFC entre sí. Esto resulta útil en situaciones en las que se desea compartir rápidamente información sin tener que realizar configuraciones adicionales o emparejamientos complejos.
- **Etiquetado inteligente**: Las etiquetas NFC, también conocidas como tags NFC, se utilizan para proporcionar información adicional o realizar acciones específicas al interactuar con ellas. Por ejemplo, en museos, se pueden colocar etiquetas NFC cerca de las obras de arte para brindar información detallada a los visitantes cuando acerquen sus dispositivos. También se pueden utilizar etiquetas NFC en productos para proporcionar información de seguimiento, autenticidad o promociones especiales al interactuar con ellas.

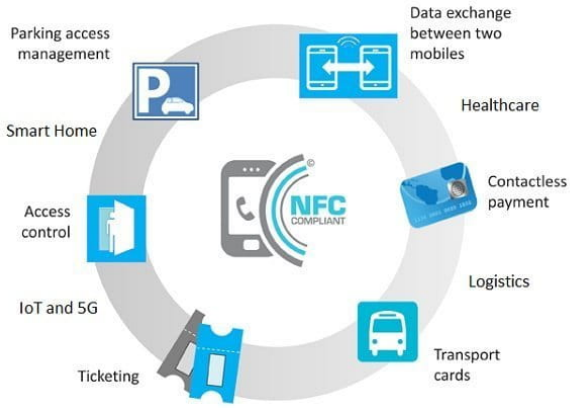

Figura 3.1: Usos de los nfcs [1]

3.6. Identificadores Únicos (UID) en NFC

Dentro del contexto de las tecnologías NFC, los Identificadores Únicos (UID, por sus siglas en inglés) desempeñan un papel fundamental. El UID es un número único asignado a cada etiqueta NFC y sirve como identificador único para diferenciar una etiqueta de otras en un sistema.

El UID es una cadena alfanumérica que se almacena en la memoria de la etiqueta NFC. Esta información se utiliza para identificar y rastrear la etiqueta en un sistema. Cada vez que un lector NFC se acerca a una etiqueta, puede leer el UID y utilizarlo para realizar diferentes acciones o proporcionar servicios específicos.

El UID tiene una longitud específica según el tipo de etiqueta NFC utilizada. Por lo general, se compone de 7 bytes (56 bits) o 4 bytes (32 bits). La longitud del UID puede variar dependiendo del estándar de la etiqueta y las especificaciones del fabricante.

Los UIDs en NFC son importantes para garantizar la autenticidad y seguridad de las transacciones. Al ser únicos permiten identificar de manera inequívoca una etiqueta y asegurar que no haya duplicados en el sistema. Esto es esencial en aplicaciones como pagos móviles y control de acceso, donde se requiere una identificación precisa y segura.

Además de ser utilizados en aplicaciones de seguridad, los UIDs también se emplean en otras funciones dentro de los sistemas NFC. Pueden ser utilizados para asociar información adicional con una etiqueta, como datos de usuario o detalles del producto. Asimismo, el UID puede servir como identificador para el seguimiento y gestión de inventario en entornos comerciales.

Es importante tener en cuenta que el UID, aunque único para cada etiqueta, no es inalterable. Algunos tipos de etiquetas NFC permiten la reescritura del UID, lo que proporciona

flexibilidad en ciertos escenarios donde se requiere cambiar o actualizar la información de la etiqueta.

Capítulo 4

Requisitos

En este capitulo se encontraran los requisitos planteados para el sitema. En ellos se describen las funcionalidades y características que se esperan del software, junto con las necesidades específicas del cliente.

La gestión de riesgos es fundamental en este capítulo, ya que permite anticipar posibles problemas y preparar planes de contingencia para minimizar su impacto. Se identifican los riesgos potenciales, se evalúa su probabilidad y su impacto, y se establecen medidas preventivas y de contingencia para mitigarlos en caso de que ocurran.

4.1. Requisitos

4.1.1. Descripción detallada del sistema

El trabajo de fin de grado consiste en un sistema de registros de entradas y salidas de empleados en un entorno laboral. En él, se plantea configurar un lector NFC junto con un servidor que lo respalde que permita a los empleados registrados en el sistema y en posesión de un identificador NFC único y personal, como una tarjera o llavero, fichar en el trabajo, guardando el dia y hora exactas en las que el trabajador entra o sale de la empresa. Además, se implementa una interfaz de usuario que permite tanto a los propios empleados revisar sus asistencias como a los administradores comprobar globalmente los registros o agregar nuevos empleados al sistema.

El sistema a desarrollar está compuesto por múltiples componentes que interactúan entre sí. Para ilustrar esta idea, se toma como ejemplo un sistema basado en tecnología NFC que permite la comunicación inalámbrica de corto alcance entre dispositivos. En este contexto, el sistema utilizará la tecnología NFC para capturar datos y procesarlos en diferentes etapas.

El sistema se desglosa en los siguientes subsistemas:

1. **Captura de datos NFC**: Este subsistema se encargará de capturar datos mediante el uso de tecnología NFC. Podrá leer etiquetas NFC y extraer la información relevante, como identificadores, datos de productos u otra información asociada a la etiqueta NFC.
2. **Procesamiento y almacenamiento de datos**: Una vez capturados los datos NFC, se procederá a su procesamiento y almacenamiento. Este subsistema validará y verificará los datos capturados contratandolos con una base de datos y los almacenará para su posterior consulta.
3. **Integración con sistemas externos**: El sistema NFC se integrará con otros sistemas externos para aprovechar la información capturada.
4. **Interfaz de usuario**: El sistema contará con una interfaz de usuario que permitirá a los usuarios interactuar con el sistema NFC. Esta interfaz puede será una interfaz web. A través de esta interfaz, los usuarios podrán visualizar la información capturada, realizar consultas y administrar los datos de los usuarios.
5. **Seguridad y autenticación**: Dado que el sistema NFC puede contener información sensible, se implementarán medidas de seguridad y autenticación para garantizar la integridad y confidencialidad de los datos. Esto puede incluir mecanismos de autenticación de usuarios, encriptación de datos y otras medidas necesarias para proteger la información.

Es importante destacar que cada subsistema del sistema NFC puede ser modular y estar encapsulado en su propio componente, lo que permitirá una mayor flexibilidad y escalabilidad en el desarrollo y mantenimiento del sistema. Además, se buscará utilizar tecnologías y estándares abiertos.

4.1.2. Requisitos funcionales

En esta sección se describen los requisitos funcionales [13] del sistema. Estos requisitos establecen las acciones y comportamientos esperados del sistema para cumplir con los objetivos planteados.

- **RF1: Crear usuarios**: El sistema debe permitir el registro de tarjetas para nuevos usuarios.
- **RF2: Realizar registro**: El sistema debe autenticar a los usuarios mediante la lectura de tarjetas y guardar las entradas y salidas de los usuarios en la base de datos.
- **RF3: Iniciar sesión**: El sistema debe permitir iniciar sesión en una interfaz de usuario.
- **RF4: Consultar registros propios** El sistema debe proporcionar una interfaz de usuario para la comprobación de los registros del propio usuario.
- **RF5: Consultar todos los registros** El sistema debe permitir a los adminstradores la consulta del historial de entradas y salidas de todos los usuarios.

4.2. Requisitos no funcionales

Los requisitos no funcionales [14] describen las características y restricciones que deben cumplirse para garantizar un rendimiento, seguridad y usabilidad adecuados. Estos requisitos se enfocan en aspectos que no están directamente relacionados con las funcionalidades específicas del sistema, pero son igualmente importantes para su correcto funcionamiento.

1. Rendimiento:
   - **RNF1: Rápido:** El sistema debe tener tiempos de respuesta rápidos para garantizar una experiencia fluida a los usuarios.
   - **RNF2: Escalable:** Se debe considerar la escalabilidad del sistema para manejar un aumento en la cantidad de usuarios y accesos simultáneos.
2. Seguridad:
   - **RNF3: Seguridad:** El sistema debe implementar medidas de seguridad para proteger la información personal de los usuarios y prevenir accesos no autorizados.
   - **RNF4: Encriptación:** Se deben utilizar técnicas de encriptación para garantizar la confidencialidad de los datos transmitidos entre el sistema y los dispositivos NFC.
3. Usabilidad:
   - **RNF5: Simple:** La interfaz del sistema debe ser intuitiva y fácil de usar para los usuarios, independientemente de su nivel de experiencia tecnológica.
   - **RNF6: Estandarizado:** Se deben seguir estándares de diseño y usabilidad para facilitar la navegación y comprensión de las funcionalidades del sistema.
4. Disponibilidad:
   - **RNF7: Disponibilidad continua:** El sistema debe estar disponible de manera continua, con un tiempo de inactividad mínimo para realizar tareas de mantenimiento.
   - **RNF8: Backups:** Se deben implementar mecanismos de respaldo y recuperación de datos para garantizar la disponibilidad de la información en caso de fallos o desastres.

4.3. Requisitos de información

Los requisitos de información [15] describen los datos necesarios para el funcionamiento y la gestión adecuada del sistema. Estos requisitos se enfocan en la estructura, el flujo y la integridad de la información que se manejará en el sistema.

1. Datos de usuarios:

- **RI1: Guardar información de usuarios:** El sistema debe almacenar la información de los usuarios, como sus nombres, credenciales de acceso y tarjetas.
- RI2: Guardar permisos de usuarios: Se deben registrar los roles y permisos asignados a cada usuario, como administrador o usuario regular.

2. Registros de accesos:

   - RI3: Guardar registros en el sistema: El sistema debe almacenar registros de cada acceso realizado por los usuarios, incluyendo la fecha, hora, y la identificación del usuario.
   - RI4: Guardar registros de los usuarios: Se deben generar identificadores únicos para cada registro de acceso con el fin de facilitar su gestión y consulta.

4.4. Especificación de los casos de uso

4.4.1. Actores

Antes de especificar los casos de uso es necesario analizar los actores principales o secundarios que interactuarán con el sistema. Actualmente hay dos roles diferenciados: Usuarios y Administradores.

- Los **usuarios** son las personas que utilizan el sistema para realizar sus entradas y salidas mediante tarjetas o dispositivos NFC. Aparte podrán revisar sus registros a través de la interfaz utilizando sus credenciales.
- Los **administradores** son usuarios con privilegios especiales que, aparte de realizar las funciones de usuario, tienen la responsabilidad de configurar y gestionar el sistema.

4.4.2. Modelo de casos de uso

A continuación en la Figura 4.1 se muestran el diagrama de casos de uso que se ha generado considerando los requisitos anteriormente descritos. En él, se observan como los diferentes usuarios con roles diferenciados pueden interactuar con el sistema de diferentes formas.

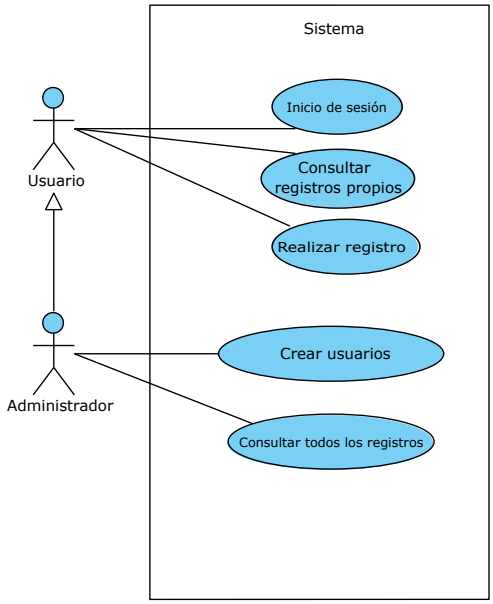

Figura 4.1: Diagrama de casos de uso

4.4.3. Casos de uso

UC1: Creación de nuevos usuarios

Descripción

Este caso de uso describe el proceso mediante el cual un administrador, con la ayuda del usuario, registra las credenciales NFC de este en el sistema.

Actores

- Administrador: Persona encargada de gestionar y configurar el sistema.
- Usuario: Persona que desea registrar sus credenciales NFC en el sistema.

Precondiciones

- El administrador ha iniciado sesión en el sistema.
- El usuario debe tener una tarjeta NFC personal para registrar.

Flujo Básico

1. El administrador selecciona la opción de registro de credenciales NFC en la interfaz de administración.
2. El sistema muestra una pantalla donde el administrador puede ingresar los datos del usuario y solicitarle que acerque su tarjeta NFC al lector correspondiente.
3. El administrador solicita al usuario que acerque su tarjeta al lector.
4. El usuario acerca la tarjeta o al lector NFC.
5. El sistema lee la información de la tarjeta o y verifica su validez.
6. El sistema asocia las credenciales NFC con la identidad del usuario en la base de datos del sistema.
7. El sistema muestra un mensaje de confirmación al administrador indicando que las credenciales NFC se han registrado exitosamente para el usuario.

Postcondiciones

Las credenciales NFC del usuario han sido registradas y asociadas a su identidad en el sistema.

Flujos alternativos

- El sistema no puede leer la información de la tarjeta: Se muestra un mensaje de error al administrador indicando que no se pudo completar el registro y se le solicita que verifique la tarjeta y vuelva a intentarlo.
- Las credenciales NFC ya están registradas para otro usuario: Se muestra un mensaje de error al administrador indicando que las credenciales NFC ya están asociadas a otra identidad en el sistema y se le solicita que utilice otras credenciales NFC para el registro.

UC2: Inicio de sesión

Descripción

Este caso de uso describe el proceso mediante el cual un usuario inicia sesión en el sistema utilizando sus credenciales de usuario y contraseña.

Actores

- Usuario: Persona que desea acceder al sistema utilizando sus credenciales.

Precondiciones

- El usuario debe estar registrado en el sistema.
- El usuario debe tener credenciales de usuario y contraseña válidas.

Flujo Básico

1. El usuario accede a la interfaz de inicio de sesión del sistema.
2. El sistema muestra los campos para ingresar el nombre de usuario y la contraseña.
3. El usuario ingresa su nombre de usuario y contraseña en los campos correspondientes.
4. El usuario envía la solicitud de inicio de sesión al sistema.
5. El sistema verifica las credenciales del usuario en la base de datos.
6. Si las credenciales son válidas, el sistema autentica al usuario y le otorga acceso al sistema.
7. El sistema muestra la interfaz principal del sistema al usuario.

Postcondiciones

El usuario ha iniciado sesión y tiene acceso al sistema.

Flujos alternativos

- Credenciales inválidas: Si las credenciales proporcionadas por el usuario no son válidas, el sistema muestra un mensaje de error indicando que las credenciales son incorrectas y solicita al usuario que vuelva a ingresar las credenciales correctas.

UC3: Registro de Entrada y Salidas

Descripción

Este caso de uso describe el proceso mediante el cual un usuario registra su entrada o salida en el sistema al pasar su tarjeta NFC por el lector correspondiente. Se verifica la información del usuario en la base de datos y se registra la entrada o salida en el sistema.

Actores

- Usuario: Persona que desea registrar su entrada o salida utilizando su tarjeta NFC.

Precondiciones

- El usuario debe estar registrado en el sistema.
- El usuario debe tener una tarjeta NFC válida asociada a su identidad en el sistema.

Flujo Básico

1. El usuario se acerca al lector de tarjetas NFC.
2. El sistema detecta la tarjeta NFC y lee la información almacenada en ella.
3. El sistema verifica la información de la tarjeta NFC en la base de datos para identificar al usuario correspondiente.
4. El sistema registra la entrada o salida del usuario en la base de datos, junto con la marca de tiempo correspondiente.
5. El sistema muestra un mensaje de confirmación al usuario indicando que su entrada o salida ha sido registrada exitosamente.

Postcondiciones

Se registra la entrada o salida del usuario en la base de datos del sistema, junto con la marca de tiempo correspondiente.

Flujos alternativos

- Tarjeta NFC no reconocida: Si la tarjeta NFC no se puede reconocer o no contiene información válida, el sistema muestra un error.
- Usuario no registrado: Si la información de la tarjeta NFC no coincide con ningún usuario registrado en el sistema, el sistema muestra un error indicando que no se ha podido llevar a cabo el registro.

UC4: Comprobar Registros

Descripción

Este caso de uso describe el proceso mediante el cual un usuario, una vez que ha iniciado sesión en el sistema, puede ver sus registros previos. El sistema muestra al usuario una lista de sus registros de entrada y salida, proporcionando información detallada sobre las fechas y horarios de cada registro.

Actores

- Usuario: Persona que desea ver sus registros previos en el sistema.

Precondiciones

- El usuario ha iniciado sesión en el sistema.

Flujo Básico

1. El usuario selecciona la opción de *Ver Registros* en la interfaz del sistema.
2. El sistema recupera los registros de entrada y salida asociados al usuario desde la base de datos.
3. El sistema muestra al usuario una lista de sus registros, mostrando la fecha, hora y tipo (entrada o salida) de cada registro.
4. El usuario puede desplazarse por la lista de registros para ver la información detallada de cada uno.

Postcondiciones

El usuario visualiza sus registros de entrada y salida almacenados en el sistema.

Flujos alternativos

- No hay registros encontrados: Si no se encuentran registros de entrada y salida asociados al usuario en la base de datos, el sistema muestra un mensaje indicando que no hay registros disponibles para mostrar.
- Error en la recuperación de datos: Si se produce un error al recuperar los registros de entrada y salida desde la base de datos, el sistema muestra un mensaje de error al usuario indicando que no se pueden mostrar los registros en este momento y sugiere intentarlo más tarde o contactar al administrador del sistema.

UC5: Ver Registros de Todos los Usuarios

Descripción

Este caso de uso describe el proceso mediante el cual un administrador, con los privilegios adecuados, puede acceder y visualizar los registros de entrada y salida de todos los usuarios en el sistema. El sistema mostrará al administrador una lista completa de registros, proporcionando información detallada sobre las fechas, horarios y usuarios asociados a cada registro.

Actores

- Administrador: Persona encargada de gestionar y configurar el sistema.

Precondiciones

- El administrador ha iniciado sesión en el sistema con los privilegios adecuados.

Flujo Básico

1. El administrador selecciona la opción de *Ver Registros de Todos los Usuarios* en la interfaz del sistema.
2. El sistema recupera todos los registros de entrada y salida almacenados en la base de datos.
3. El sistema muestra al administrador una lista completa de registros, mostrando la fecha, hora, tipo (entrada o salida) y usuario asociado a cada registro.
4. El administrador puede desplazarse por la lista de registros para ver la información detallada de cada uno.

Postcondiciones

El administrador visualiza los registros de entrada y salida de todos los usuarios almacenados en el sistema.

Flujos alternativos

- No hay registros encontrados: Si no se encuentran registros de entrada y salida en la base de datos, el sistema muestra un mensaje indicando que no hay registros disponibles para mostrar.

- Error en la recuperación de datos: Si se produce un error al recuperar los registros de entrada y salida desde la base de datos, el sistema muestra un mensaje de error al administrador indicando que no se pueden mostrar los registros en este momento y sugiere intentarlo más tarde o contactar al soporte técnico.

Capítulo 5

Análisis

5.1. Introdución

En este capitulo se cubre el análisis del proyecto. En él, se expondrá el modelo de dominio, el modelo de análisis y los diferentes diagramas de secuencia de los casos de uso que se han utilizado. El análisis desempeña un papel fundamental en la ingeniería de software al permitir examinar en profundidad un sistema, aplicación o software. Su objetivo principal es comprender la estructura, funcionamiento y requisitos del sistema en cuestión. Mediante el análisis, se pueden identificar posibles problemas, detectar oportunidades de mejora y establecer un plan sólido para el desarrollo del software.

5.2. Modelo de Dominio

El diagrama de clases del dominio [16] es una representación visual de las entidades principales, sus atributos y las relaciones entre ellas en un dominio de negocio.

En la Figura 5.1 podemos observar el diagrama de modelo de clases de dominio de nuestro proyecto. En él podemos encontrar la clase usuarios, que equivaldría a cada uno de los empleados registrados en el sistema. Para cada empleado se guardaría sus credenciales, una tarjeta de identificación y se le asignaría un rol que dista entre administrador y empleado. Además, se almacenan cada uno de los registros de los empleados en el sistema con un identificador único y la fecha en la que se realizó.

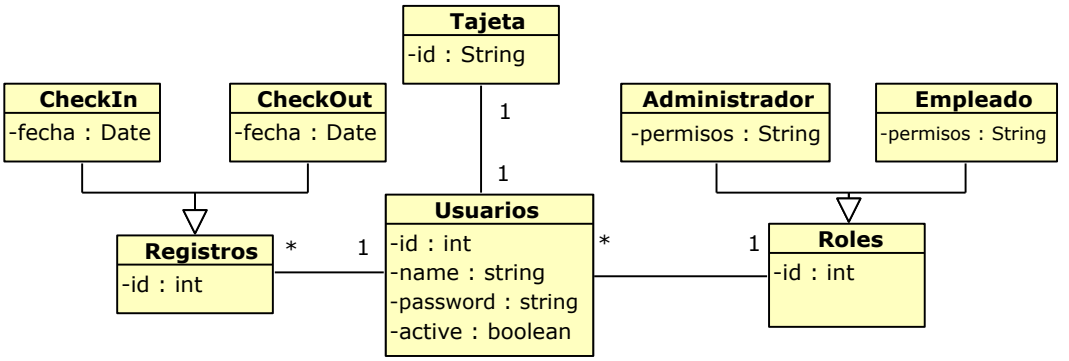

Figura 5.1: Diagrama de clases del modelo de dominio

5.3. Modelo de Análisis

5.3.1. Clases de análisis

El diagrama de clases de análisis [17], mostrado en la Figura 5.2, es una representación visual que muestra la estructura y las relaciones entre las entidades clave de un sistema en un nivel conceptual, sin detalles de implementación. Proporciona una visión general de las clases y sus atributos, ayudando a comprender los requisitos del sistema antes de pasar al diseño y la implementación.

A continuación se muestran el diagrama de clases del modelo de análisis, que engloba el diagrama 5.1 junto con las operaciones necesarias para llevar cubrir los diferentes casos de uso.

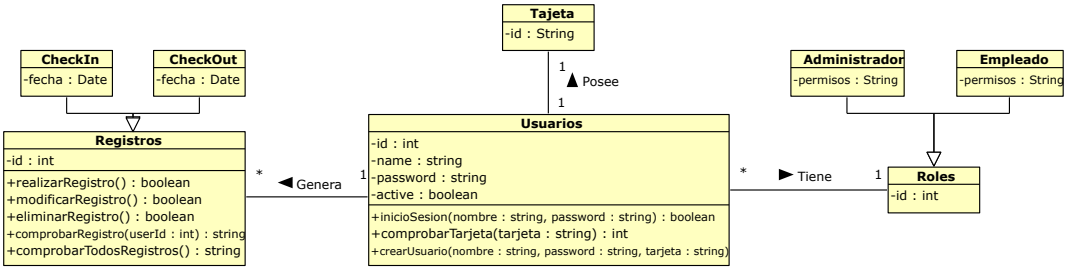

Figura 5.2: Modelo de Análisis

5.3.2. Diagramas de Realización de Casos de Uso

En esta sección se presentan los diagramas de secuencia [18] de los casos de uso.Un diagrama de secuencia es una herramientas visual que permite representar la interacción

entre los diferentes objetos o componentes de un sistema durante la ejecución de un caso de uso específico.

Estos diagramas muestran el orden y la secuencia de los mensajes intercambiados entre los objetos, lo que nos ayuda a comprender el flujo de control y las acciones realizadas en cada paso del caso de uso.

A continuación, se presentarán los diagramas de secuencia correspondientes a los diferentes casos de uso del sistema:

Creación de nuevos usuarios

La Figura 5.3 muestra el diagrama de secuencia correspondiente al caso de uso CU1.

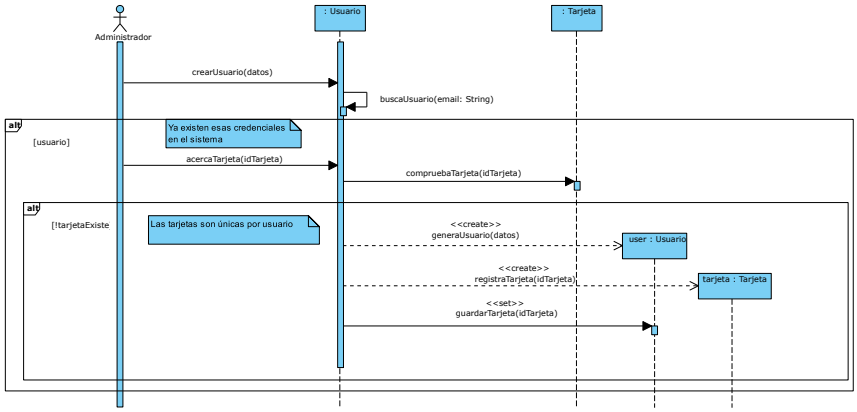

Figura 5.3: Diagrama de Secuencia - Creación de nuevos usuarios

Registro de entradas y salidas

La Figura 5.4 muestra el diagrama de secuencia correspondiente al caso de uso CU2.

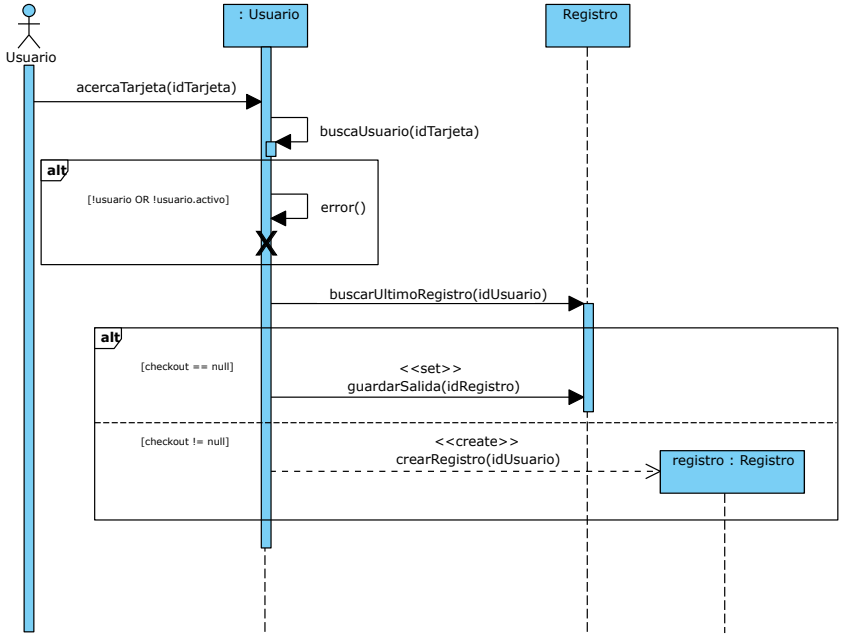

Figura 5.4: Diagrama de Secuencia - Registrar entrada/salida

Inicio de sesión

La Figura 5.5 muestra el diagrama de secuencia correspondiente al caso de uso CU3.

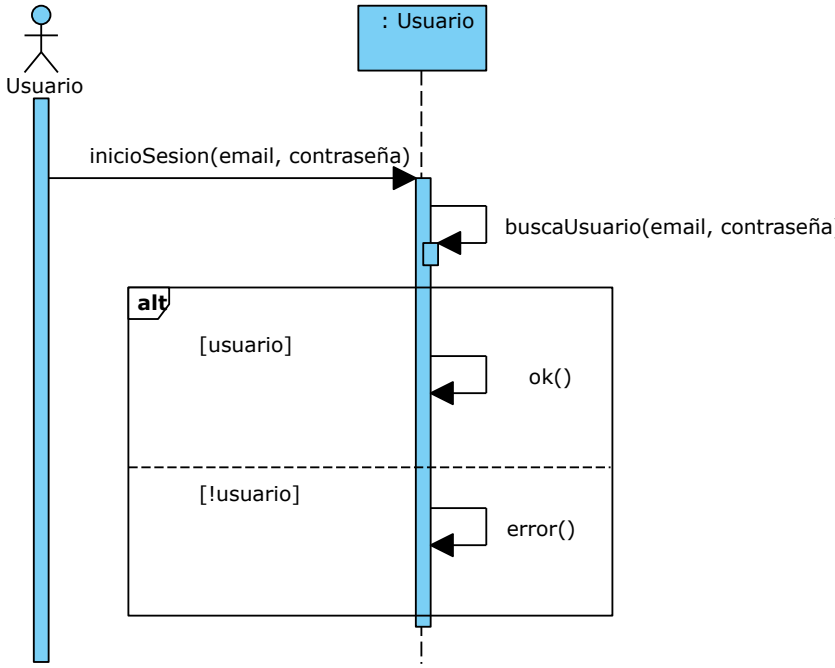

Figura 5.5: Diagrama de Secuencia - Inicio de sesión

Comprobar registros de un usuario

La Figura 5.6 muestra el diagrama de secuencia correspondiente al caso de uso CU4.

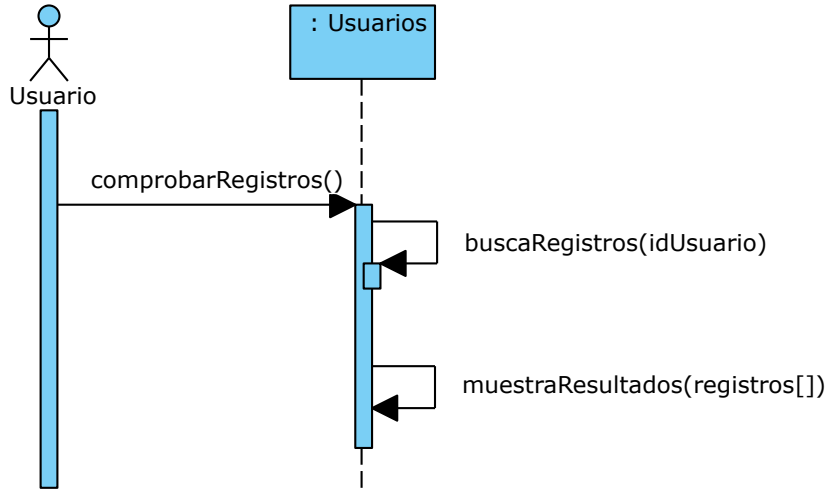

Figura 5.6: Diagrama de Secuencia - Comprobar registros de un usuario

Comprobar registros de todos los usuarios

La Figura 5.7 muestra el diagrama de secuencia correspondiente al caso de uso CU5.

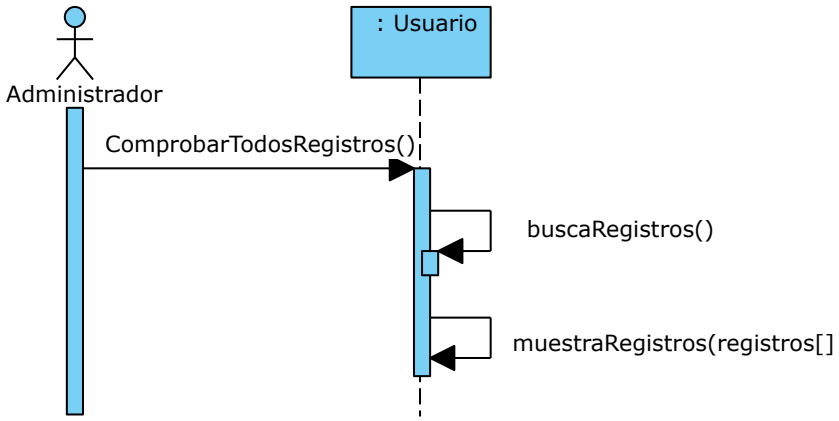

Figura 5.7: Diagrama de Secuencia - Comprobar registros de todos los usuarios

Capítulo 6

Diseño

6.1. Introducción

En este capítulo, se presentarán los diferentes aspectos del diseño del sistema, desde la arquitectura hasta los patrones de diseño y base de datos utilizados. El diseño en la ingeniería de software es una etapa crucial en el desarrollo de sistemas y aplicaciones. Consiste en la creación de una solución estructurada y detallada que cumpla con los requisitos establecidos durante el análisis. Durante esta fase, se definen las especificaciones técnicas, se establece la arquitectura del sistema y se diseñan los componentes individuales. Un diseño sólido sienta las bases para la implementación eficiente y efectiva del software, asegurando su funcionalidad, rendimiento y mantenibilidad.

6.2. Arquitectura Lógica del Sistema

La arquitectura de software [19] juega un papel fundamental en el desarrollo de sistemas robustos. Los patrones arquitectónicos son enfoques probados y consolidados que proporcionan soluciones generales a problemas comunes en el diseño y organización de un sistema. Estos patrones ofrecen una estructura y guía para la distribución de responsabilidades, la interacción entre componentes y la escalabilidad del sistema. Al utilizar patrones de arquitectura, los desarrolladores pueden beneficiarse de soluciones probadas y establecidas, lo que les permite construir sistemas más eficientes, mantenibles y flexibles. Estos patrones proporcionan un marco conceptual para el diseño y permiten la reutilización de soluciones exitosas en diferentes contextos de desarrollo.

6.2.1. Patrón Arquitectónico del sistema

El sistema en su totalidad uso un modelo de 3 capas [20]. Cada capa se muestra de diferente color en la Figura 6.1. El modelo de capas es un patrón arquitectónico ampliamente utilizado en el desarrollo de sistemas de software. Se basa en la idea de dividir la aplicación en capas lógicas o niveles, cada una con un conjunto específico de responsabilidades y funcionalidades.

En el sistema del proyecto, se adopta el patrón arquitectónico de capas para organizar y estructurar la arquitectura del software. Las capas se definen en función de las responsabilidades y el nivel de abstracción de cada componente. A continuación, se describen las tres capas principales de nuestro sistema.

Capa de Presentación o *Front-end*: Esta capa corresponde a la interfaz de usuario y se encarga de presentar los datos al usuario final y recibir sus interacciones. En este caso, la capa de presentación está representada por la aplicación Web desarrollada con Angular. Esta capa se comunica con la capa de lógica a través de una API para obtener y enviar datos.

Capa de Lógica o *Back-end*: En esta capa se encuentra la lógica de negocio de la aplicación. Aquí se procesan las solicitudes del usuario, se aplican reglas de negocio y se realizan operaciones de procesamiento y manipulación de datos. Se ha utilizado Node.js junto con el *framework* Express para implementar la capa de lógica, creando una API RESTful que se comunica con la capa de persistencia y la capa de presentación.

Capa de Persistencia o Base de datos: Esta capa es responsable de almacenar y gestionar los datos de la aplicación. En el contexto del sistema, se ha implementado una base de datos relacional de tipo *MySQL*.

Cada capa en el modelo de capas tiene sus propias responsabilidades y se comunica con las capas inferiores utilizando interfaces bien definidas. Esto proporciona un acoplamiento débil entre las capas y facilita la escalabilidad y la evolución del sistema. Además, esta arquitectura permite que cada capa se desarrolle y mantenga de forma independiente, lo que mejora la productividad del equipo de desarrollo.

6.2.2. Aplicación Web

La parte del front-end está representada por la aplicación Web desarrollada con Angular, un *framework* ampliamente utilizado para la construcción de aplicaciones de este tipo. Angular, por defecto, adopta el patrón Modelo-Vista-Controlador (MVC) como puede verse en el paquete Angular de la Figura 6.1 para organizar y estructurar el código.

El patrón MVC [21] se basa en la separación de responsabilidades en tres componentes principales: el Modelo, la Vista y el Controlador. Esta estructura modular permite un desarrollo más organizado y mantenible de la aplicación.

El **Modelo** representa los datos y la lógica de negocio subyacente. En Angular, el Modelo se define mediante servicios y proveedores que encapsulan la lógica de la aplicación, como la

obtención y manipulación de datos. Estos servicios proporcionan una interfaz para acceder a los datos y gestionar su estado.

La **Vista** se encarga de la representación visual de los datos y la interacción con el usuario. En Angular, la Vista se compone de componentes, plantillas y estilos CSS. Los componentes definen la estructura y comportamiento de cada parte de la interfaz de usuario, mientras que las plantillas determinan cómo se visualizan los datos en la pantalla. Los estilos CSS se utilizan para darle un aspecto visual agradable a la aplicación.

El **Controlador** actúa como intermediario entre el Modelo y la Vista, gestionando la interacción del usuario y actualizando el estado de la aplicación en función de las acciones realizadas. En Angular, los controladores se implementan a través de componentes y clases controladoras, que manejan eventos, realizan validaciones y se comunican con los servicios del Modelo. Su objetivo es garantizar que la Vista y el Modelo estén sincronizados y actualizados en todo momento.

Al utilizar Angular como *framework* para el desarrollo del *front-end*, se aprovecha su soporte nativo para el patrón Modelo-Vista-Controlador (MVC). Esto brinda una estructura clara y modular para organizar el código, separando las responsabilidades.

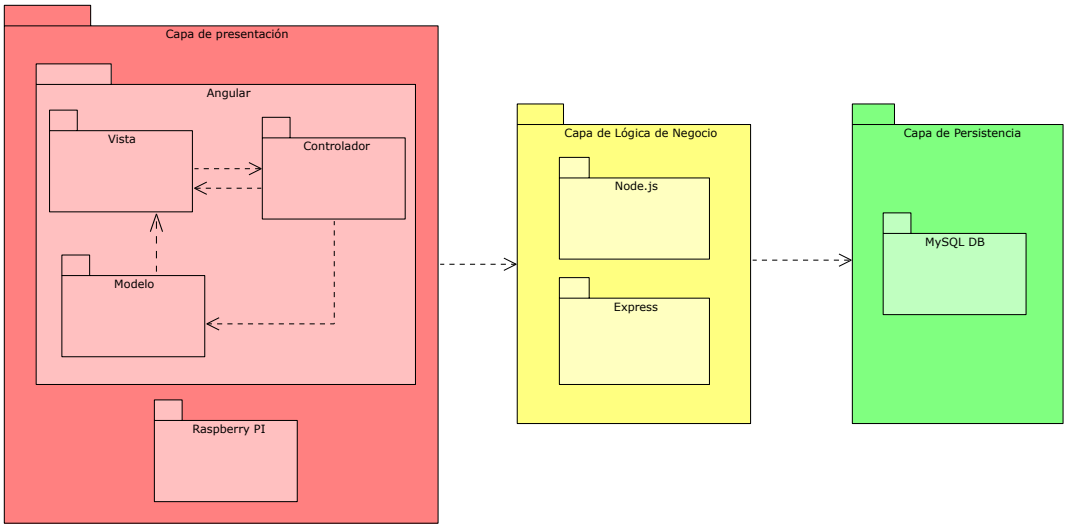

Figura 6.1: Diagrama de paquetes

6.3. Arquitectura física del sistema

La arquitectura física de un sistema se refiere a la disposición y configuración de los componentes físicos que forman parte del sistema, como servidores, redes y dispositivos. Para representar dicha estructura, se utilizará un diagrama de despliegue.

Un diagrama de despliegue [22] es una representación visual que muestra la configuración y distribución física de los componentes de un sistema en un entorno de ejecución. Este tipo de diagrama muestra cómo los diferentes elementos del sistema, como hardware, software, redes y dispositivos, interactúan y se interconectan entre sí. Proporciona una visión clara de la arquitectura física del sistema, incluyendo servidores, computadoras, dispositivos móviles y cómo se comunican entre sí. El diagrama de despliegue es útil para comprender la infraestructura necesaria para implementar y ejecutar el sistema, así como para identificar los componentes que se distribuyen en diferentes ubicaciones físicas.

La Figura 6.2 muestra el diagrama de despliegue del proyecto.

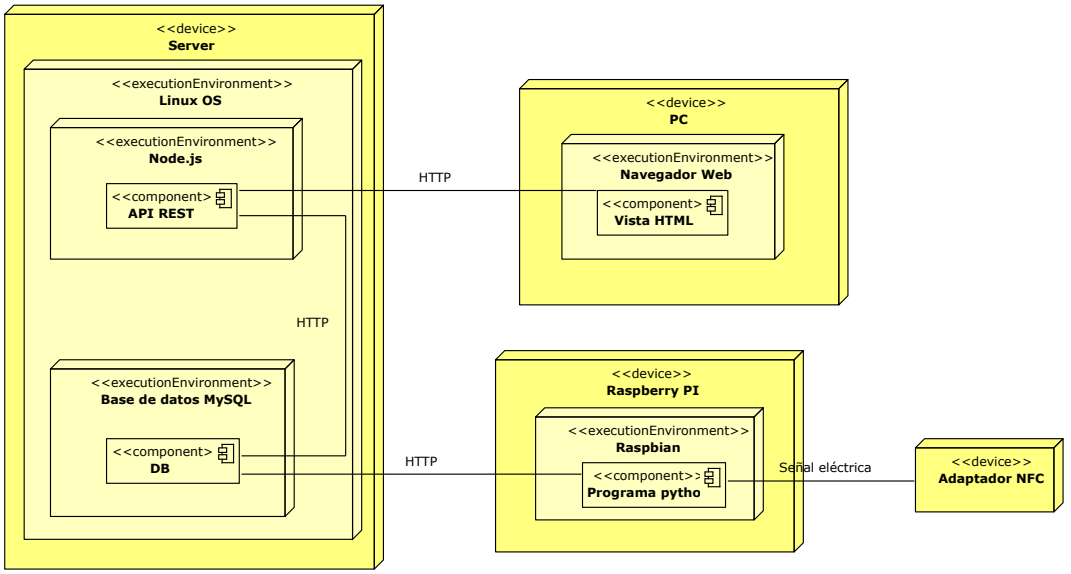

Figura 6.2: Diagrama de Despliegue

6.4. Diseño de la base de datos

El diseño de la base de datos es una etapa crucial en el desarrollo de un sistema de software, ya que define la estructura y organización de los datos que se almacenarán y manipularán. En nuestro proyecto, hemos utilizado una base de datos MySQL alojada en el servidor para gestionar y persistir la información.

En la Figura 6.3, se presenta un diagrama relacional de la base de datos, que representa las tablas y las relaciones entre ellas. Este diagrama proporciona una visión general de la estructura de la base de datos y muestra cómo se organizan los datos en diferentes entidades y cómo se relacionan entre sí. Esta estructura se basa en el diseño del modelo de dominio en la Figura 5.1 anteriormente comentado.

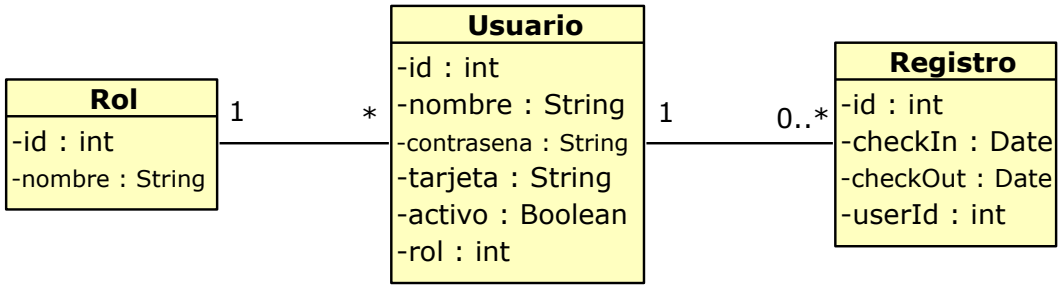

Figura 6.3: Diagrama Relacional de la Base de Datos

El diagrama relacional [23] muestra las tablas de la base de datos, identificando las columnas y los tipos de datos que contienen. También se representan las relaciones entre las tablas a través de las claves primarias y foráneas. Esto permite establecer la integridad referencial y garantizar la consistencia de los datos.

En el diseño de base de datos, se consideran las entidades y relaciones relevantes para el sistema, asegurando una estructura eficiente y coherente para el almacenamiento y acceso a los datos. El uso de una base de datos relacional, como en este caso MySQL, proporciona flexibilidad y escalabilidad para gestionar la información de manera robusta.

Es importante destacar que este diagrama de base de datos es una representación visual del diseño, y puede haber detalles adicionales a nivel de atributos y restricciones que no se muestran en este diagrama de alto nivel. Sin embargo, sirve como una guía útil para comprender la estructura general de la base de datos y las relaciones entre las entidades.

Capítulo 7

Tecnologías Ytilizadas

En este capítulo se presentan las tecnologías utilizadas en el desarrollo del sistema de entradas y salidas de usuarios mediante NFC. Se abordan los *frameworks* de Angular y Node.js, los lenguajes de programación JavaScript y Python, y el concepto de *API (Application Programming Interfaces) REST (Representational State Transfer).*

7.1. Frameworks

7.1.1. Angular

Angular es un *framework* de desarrollo de aplicaciones Web de código abierto desarrollado por Google. Utiliza el lenguaje de programación TypeScript, que agrega características de tipado estático a JavaScript, lo que facilita el desarrollo de aplicaciones más robustas. Angular sigue el patrón arquitectónico MVC (Modelo-Vista-Controlador) y proporciona una estructura modular y escalable para construir interfaces de usuario interactivas [24].

Entre las características principales de Angular se incluyen:

- **Enlace de datos bidireccional** que permite una actualización automática de la vista cuando los datos cambian y viceversa.
- **Enrutado dinámico** que permite navegar entre las diferentes vistas dentro de la aplicación al interactuar con ello o modificar su dirección en la URL, así ofrenciendo la posibilidad de tener las diferentes vistas y funciones modularizadas e independientes unas de otras.
- **Inyección de dependencias** que facilita la gestión de las dependencias entre los diferentes componentes de la aplicación.

Angular proporciona una amplia gama de herramientas y bibliotecas que simplifican el desarrollo de aplicaciones Web complejas y optimizan el rendimiento.

7.1.2. Node.js

Node.js es un entorno de ejecución de código abierto basado en el motor de JavaScript V8 de Google Chrome. A diferencia de JavaScript, que se ejecuta en el navegador, Node.js permite ejecutar código JavaScript en el servidor. Esto brinda la posibilidad de construir aplicaciones Web escalables y en tiempo real permitiendo procesar y responder a eventos y actualizaciones de forma instantánea, sin la necesidad de que los usuarios realicen una acción adicional para obtener los cambios. [25].

Las principales características de Node.js incluyen:

- Modelo de E/S sin bloqueo y orientado a eventos que permite manejar eficientemente múltiples conexiones concurrentes. Esto brinda la posibilidad de dar servicio a varios usuarios concurrentes para las mismas peticiones.
- Biblioteca estándar y un amplio ecosistema de módulos y paquetes que facilitan el desarrollo de aplicaciones Web.
- Acceso directo a las API de sistema operativo, lo que permite interactuar con archivos, redes y otros recursos del sistema.
- Escalabilidad horizontal que permite manejar grandes volúmenes de solicitudes utilizando múltiples hilos de ejecución.

Node.js es ampliamente utilizado en el desarrollo de servidores Web, aplicaciones en tiempo real, servicios de *backend* y microservicios.

7.1.3. Express

Express es un *framework* Web minimalista y flexible para Node.js, que proporciona una capa de abstracción sobre el manejo de solicitudes y respuestas HTTP *(Hypertext Transfer Protocol)*. Es uno de los *frameworks* más populares y ampliamente utilizados en el desarrollo de aplicaciones Web basadas en Node.js [26].

Las características clave de Express incluyen:

- **Enfoque minimalista**: Express adopta una filosofía simple. Proporciona solo las funcionalidades esenciales necesarias para construir una aplicación Web, lo que permite un mayor control y personalización por parte del desarrollador.

- **Enrutamiento sencillo**: Una ruta es una dirección que se utiliza para acceder a un recurso específico en un servidor. En el contexto de desarrollo Web se refiere a las URLs *(Uniform Resource Locator)*. Express facilita la definición de rutas y la gestión de las solicitudes entrantes. Permite crear rutas para diferentes URL y métodos HTTP, y asociar controladores de ruta que se ejecutan cuando se accede a esas rutas. Esto proporciona una estructura clara y organizada para el manejo de solicitudes.
- ***middleware***: Express utiliza *middleware* para realizar diversas tareas en el procesamiento de solicitudes y respuestas. El *middleware* es una función que se ejecuta en el flujo de procesamiento de la solicitud y puede realizar operaciones como el análisis de datos, la autenticación, la gestión de sesiones y la compresión de respuestas. Esto permite una mayor modularidad y reutilización de código. Un ejemplo de uso de middleware puede ser una función de validación de peticiones, que comprobaría si en los parametros de la solicitud se encuentran los datos necesarios para poder realizarse con exito.
- **Amplia compatibilidad**: Express es compatible con una amplia gama de módulos y bibliotecas de Node.js, lo que amplía aún más sus capacidades y permite integrar fácilmente otras funcionalidades en una aplicación Express.
- **Gran comunidad y soporte**: Express cuenta con una gran comunidad de desarrolladores activos que contribuyen con módulos, plugins y recursos en línea. Esto garantiza un soporte sólido, una documentación extensa y una resolución rápida de problemas.

7.2. Lenguajes de Programación

7.2.1. JavaScript

JavaScript es un lenguaje de programación versátil y ampliamente utilizado en el desarrollo Web. Aunque inicialmente se diseñó para su ejecución en el navegador Web del cliente, con el tiempo, JavaScript ha evolucionado y se ha expandido hacia otros ámbitos, incluido el desarrollo del lado del servidor [27].

En este proyecto, JavaScript se utiliza tanto en el desarrollo del lado del cliente con Angular como en el lado del servidor con Node.js. A continuación, se presentan algunas características clave de JavaScript en este contexto:

- **Desarrollo del lado del cliente**: JavaScript permite la interacción dinámica con el DOM (Document Object Model) de una página Web. Esto significa que se puede utilizar para manipular elementos HTML, gestionar eventos, validar formularios y proporcionar una experiencia interactiva al usuario en el navegador.
- **Desarrollo del lado del servidor**: Gracias a la introducción de Node.js ahora también es posible utilizar JavaScript para desarrollar aplicaciones de servidor. Esto significa que se puede utilizar el mismo lenguaje de programación tanto en el lado del cliente

como en el lado del servidor, lo que simplifica el desarrollo y mejora la coherencia del código.

- **Ecosistema de bibliotecas y *frameworks***: JavaScript cuenta con un vasto ecosistema de bibliotecas y *frameworks* que facilitan el desarrollo tanto en el lado del cliente como en el lado del servidor. En el desarrollo del sistema de entradas y salidas de usuarios mediante NFC, se pueden utilizar *frameworks* populares como Express.js para construir el servidor Web y gestionar las rutas y solicitudes HTTP.
- **Facilidad de uso y aprendizaje**: JavaScript es conocido por su sintaxis sencilla y su curva de aprendizaje relativamente baja. Esto lo convierte en una opción atractiva tanto para desarrolladores novatos como experimentados. Además, la gran cantidad de recursos, documentación y comunidades en línea disponibles hacen que sea fácil obtener soporte y resolver problemas durante el desarrollo.

7.2.2. Python

Python es un lenguaje de programación de alto nivel conocido por su legibilidad y su enfoque en la facilidad de uso. Su sintaxis clara y concisa hace que sea fácil de aprender y leer, lo que agiliza el proceso de desarrollo y mantenimiento del código [28].

Las características principales de Python son:

- **Sintaxis legible**: Python destaca principalmente por su legibilidad, lo que significa que el código escrito en Python tiende a ser más claro y comprensible en comparación con otros lenguajes. Esto facilita la colaboración en proyectos y la comprensión del código existente.
- **Amplia colección de bibliotecas y *frameworks***: Python cuenta con una amplia gama de bibliotecas y *frameworks* que permiten a los desarrolladores realizar diversas tareas de manera eficiente. Por ejemplo, Flask y Django son dos *frameworks* populares de Python para el desarrollo Web, que proporcionan herramientas y estructuras para construir aplicaciones Web robustas y escalables.
- **Multiplataforma**: Python es compatible con diferentes sistemas operativos, como Windows, macOS y Linux. Esto significa que el código escrito en Python se puede ejecutar en diferentes entornos sin necesidad de modificaciones significativas, lo que agiliza el proceso de desarrollo y despliegue.
- **Gran comunidad y soporte**: Python cuenta con una comunidad activa de desarrolladores que contribuyen a su crecimiento y mejora continua. Además, existe una gran cantidad de recursos en línea, documentación y tutoriales que facilitan el aprendizaje y la resolución de problemas.

7.3. API REST

Una API REST (Representational State Transfer) es un conjunto de reglas y convenciones para la comunicación entre sistemas a través del protocolo HTTP. La API REST se basa en el concepto de recursos, que son objetos o servicios a los que se puede acceder mediante identificadores únicos (URLs).

La comunicación con una API REST se realiza mediante los métodos HTTP, como *get, post, put y delete*, que representan operaciones básicas sobre los recursos. Por ejemplo, GET se utiliza para obtener información sobre un recurso, POST para crear uno nuevo, PUT para actualizarlo y DELETE para eliminarlo.

La arquitectura REST es *stateless*, lo que significa que no guarda información sobre las solicitudes anteriores del cliente. Esto permite una alta escalabilidad y simplicidad en el diseño de las API. Además, las respuestas de una API REST suelen estar en formato JSON (JavaScript Object Notation), que es fácilmente legible y manejable por aplicaciones cliente [29].

En el desarrollo del sistema de entradas y salidas de usuarios mediante NFC, se ha implementado una API REST para facilitar la interacción entre los componentes del sistema, como la aplicación Web y la base de datos. La API REST permitirá realizar operaciones de registro, consulta y actualización de datos de manera eficiente y segura.

7.4. JSON Web Tokens (JWT)

Los *JSON Web Tokens (JWT)*[2] son un estándar abierto (RFC 7519) para la emisión de tokens de seguridad que se utilizan en la autenticación y autorización en aplicaciones Web. Un JWT consta de tres partes como se puede observar en la Figura 7.1:

- Encabezado (*Header*): El encabezado de un JWT contiene información sobre el tipo de token y el algoritmo de firma utilizado. Esta sección está codificada en Base64 y proporciona detalles sobre cómo se debe procesar el *token*.
- Carga útil (*Payload*): La carga útil de un JWT contiene la información adicional que se desea transmitir, como datos de usuario, roles o permisos. Esta sección también está codificada en Base64 y puede incluir información personalizada según las necesidades de la aplicación.
- Firma (*Signature*): La firma del JWT se utiliza para verificar la integridad de los datos transmitidos y garantizar que no hayan sido modificados en el proceso. La firma se calcula utilizando el encabezado, la carga útil, una clave secreta conocida solo por el servidor y el algoritmo de firma especificado en el encabezado. La verificación de la firma asegura que el token no ha sido manipulado y que proviene de una fuente confiable.

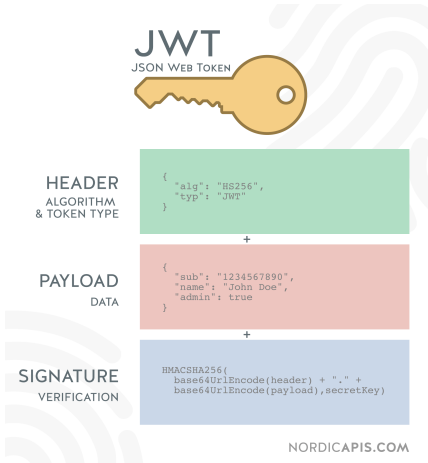

Figura 7.1: Estructura de un JWT [2]

La principal ventaja de utilizar JWT en la autenticación es que el servidor no necesita almacenar información de sesión para cada usuario. En su lugar, la información necesaria se encuentra dentro del propio token, lo que simplifica el mantenimiento del estado del servidor. Además, JWT permite la propagación eficiente de la información de autenticación entre diferentes servicios o dominios, lo que facilita la creación de arquitecturas distribuidas y escalables.

7.5. Herramientas para documentación

7.5.1. Visual Paradigm

Visual Paradigm es una herramienta de modelado y diseño de software que permite crear diagramas y modelos visuales para representar conceptos, estructuras y procesos en el desarrollo de software. Proporciona una amplia gama de características y notaciones estándar para facilitar el diseño de sistemas, incluyendo diagramas UML, diagramas de flujo, diagramas de clases y muchos más. Para este proyecto se ha utilizado la versión v16.2 para elaborar todos los diagramas recopilados en este documento. La licencia es la herramienta es de pago pero la Universidad de Valladolid ofrece a sus alumnos licencias de manera gratuita.

7.5.2. Overleaf

Overleaf es una plataforma en línea para la edición y colaboración de documentos LaTeX. LaTeX es un sistema de composición de documentos ampliamente utilizado en el ámbito académico y científico debido a su capacidad para producir documentos de alta calidad tipográfica. Overleaf proporciona una interfaz fácil de usar y potentes herramientas de edición

y compilación que permiten a los usuarios crear, editar y compilar documentos LaTeX en tiempo real. Además, ofrece características de colaboración que permiten a múltiples usuarios trabajar juntos en un documento, facilitando la colaboración en proyectos académicos, científicos y técnicos. Esta herramienta ha sido la utilizada para redactar este documento en conjunto con el tutor.

7.6. Herramientas de comunicación y organización

7.6.1. Trello

Trello es una plataforma en línea que permite la gestión y organización de proyectos de manera colaborativa. Utilizando un sistema de tableros, listas y tarjetas, Trello permite a los usuarios organizar tareas, asignar responsabilidades, establecer fechas límite y realizar un seguimiento del progreso de los proyectos. Con una interfaz intuitiva y fácil de usar, Trello es una herramienta eficiente para la planificación y gestión de proyectos, tanto a nivel individual como en equipo. Se ha utilizado esta herramienta para organizar las tareas de los *sprints* y coordinar los avances con el tutor.

7.6.2. Microsoft Teams

Microsoft Teams es una plataforma de colaboración en línea que permite a los equipos de trabajo comunicarse, compartir archivos y colaborar en tiempo real. A través de salas de chat y videollamadas, Teams facilita la comunicación y la colaboración eficiente entre los miembros del equipo. Además, ofrece características como la compartición de pantalla, la gestión de tareas y la posibilidad de crear canales temáticos para organizar la información de manera estructurada. Con una interfaz intuitiva y accesible desde diferentes dispositivos, Esta herramienta se ha utilizado para mantener una comunicación constante con el tutor.

Capítulo 8

Implementación y Pruebas

En este capítulo, se proporcionará un manual detallado para ejecutar el TFG en un entorno de desarrollo. Además, se presentará una explicación sobre cómo se ha organizado el código fuente del proyecto.

El manual de ejecución guiará al usuario a través de los pasos necesarios para poder configurar y desplegar el sistema en su propia máquina. Se incluirán instrucciones sobre la instalación de las dependencias, la configuración de la base de datos y la ejecución de la aplicación. En cuanto a la organización del código, se describirá la estructura general del proyecto y se explicará cómo se han distribuido los diferentes componentes y módulos

8.1. Manual de Despliegue

En esta sección, se proporcionarán las instrucciones detalladas para desplegar y ejecutar el sistema en los diferentes componentes: la Raspberry Pi, la API desarrollada con Node.js, la base de datos y el frontend construido con Angular.

8.1.1. Raspberry Pi

Para desplegar el programa en la Raspberry Pi, siga los siguientes pasos:

1. Asegúrese de tener instalado Python en la Raspberry Pi. Puede verificar si Python está instalado ejecutando el comando 'python –version' en la terminal. Si no está instalado, siga las instrucciones de instalación proporcionadas por el sistema operativo de la Raspberry Pi.
2. Transfiera el archivo del programa Python a la Raspberry Pi, ya sea mediante una conexión de red o mediante un medio de almacenamiento físico.

3. Abra una terminal en la Raspberry Pi y navegue hasta el directorio donde se encuentra el programa Python.

4. Ejecute el siguiente comando para iniciar la aplicación:

```
python programa.py
```

   Una vez completado todos los pasos se iniciará el programa y comenzará a tomar medidas establecidas en el programa ejecutado. Puede ver la salida en la terminal y verificar que la Raspberry Pi esté funcionando correctamente.

8.1.2. API (Node.js)

Para desplegar la API desarrollada con Node.js, siga los siguientes pasos:

1. Clone el repositorio de la API desde el repositorio remoto o copie el código fuente en su máquina local. Puede utilizar herramientas de control de versiones como Git para clonar el repositorio.

2. Asegúrese de tener instalada la versión adecuada de Node.js en su máquina. Puede verificar la versión instalada ejecutando el comando 'node –version' en la terminal. Si no tiene Node.js instalado, puede descargarlo e instalarlo desde el sitio web oficial de Node.js. En el archivo .nvmrc se encuentra la versión de node utilizada. Si tiene instalado el control de versiones de node 'nvm' podrá usar la versión del programa automaticamente con el comando:

```
nvm use
```

3. Abra una terminal y navegue hasta el directorio raíz de la API y posteriormente ejecute el siguiente comando para instalar las dependencias del proyecto:

```
npm install
```

   Esto descargará e instalará todas las dependencias necesarias especificadas en el archivo 'package.json'.

4. Configure el archivo '.env' con los valores correspondientes para su entorno de desarrollo para las variables encontradas en el archivo '.env.example'. Este archivo contiene variables de entorno utilizadas por la API, como la configuración de la base de datos o la configuración del token JWT. Asegúrese de proporcionar los valores correctos para su entorno.

5. Una vez configurado el archivo '.env', ejecute el siguiente comando para iniciar la API:

```
npm start
```

Una vez inicializado el programa se levantará el servidor y estará preparada para recibir y responder a las solicitudes. Es posible que se requiera previamente haber creado la base de datos descrita en el siguiento apartado

8.1.3. Base de Datos

Para inicializar la base de datos, siga los siguientes pasos:

1. Asegúrese de tener desplegado un servidor de base de datos relacional compatible, como MySQL, PostgreSQL o SQLite, en su máquina local o en un servidor remoto.
2. Cree una base de datos vacía en el servidor de base de datos con el nombre y el esquema que desee utilizar para el sistema. Puede utilizar una herramienta de administración de base de datos, como phpMyAdmin o pgAdmin, para crear la base de datos.
3. Abra un terminal y navegue hasta el directorio raíz del proyecto donde se encuentra el archivo de configuración de Sequelize. En este archivo, se especifican los detalles de la conexión a la base de datos, como el nombre de usuario, la contraseña y el nombre de la base de datos.
4. Ejecute el siguiente comando para realizar las migraciones de Sequelize y crear las tablas en la base de datos:

   ```
   npx sequelize-cli db:migrate
   ```

   Esto ejecutará las migraciones definidas en el proyecto y creará las tablas en la base de datos de acuerdo con el esquema especificado en los archivos de migración.
5. A continuación, ejecute el siguiente comando para ejecutar los seeders y poblar la base de datos con datos de ejemplo:

   ```
   npx sequelize-cli db:seed:all
   ```

   Esto ejecutará los seeders definidos en el proyecto y creará registros iniciales en las tablas de la base de datos.

Finalizados todos los pasos, se debería tener una base datos encendida y operativa con las tablas requeridas para la API y un conjunto de datos inicial.

8.1.4. Frontend (Angular)

Para desplegar el frontend construido con Angular, siga los siguientes pasos:

1. Asegúrese de tener instalada la versión adecuada de Angular en su máquina. Puede verificar la versión instalada ejecutando el comando 'ng –version' en la terminal. Si no tiene Angular instalado, puede instalarlo utilizando el siguiente comando:

   ```
   npm install -g @angular/cli
   ```

   Esto instalará la CLI de Angular globalmente en su máquina.

2. Clone el repositorio del frontend desde el repositorio remoto o copie el código fuente en su máquina local. Puede utilizar herramientas de control de versiones como Git para clonar el repositorio.

3. Abra una terminal y navegue hasta el directorio raíz del frontend.

4. Ejecute el siguiente comando para instalar las dependencias del proyecto:

   ```
   npm install
   ```

   Esto descargará e instalará todas las dependencias necesarias especificadas en el archivo 'package.json'. Asegúrese de tener una conexión a Internet estable para que las dependencias se descarguen correctamente.

5. Una vez instaladas las dependencias, ejecute el siguiente comando para iniciar el servidor de desarrollo:

   ```
   ng serve
   ```

Esto iniciará el servidor de desarrollo de Angular y estará disponible en 'http://localhost:4200' en su navegador web. Puede acceder a la aplicación desde esta URL y realizar pruebas en el entorno local.

8.2. Organización del código

En cuanto a la organización del código, se han seguido buenas prácticas en la estructuración del repositorio para mejorar la mantenibilidad y perseverar la escalabilidad del proyecto. A continuación, se describen las principales características de la organización del código:

1. **Estructura de carpetas**: El proyecto se encuentra en un repositorio GIT, donde cada sistema atómico se encuentra en una carpeta separada del resto. Esto permite una fácil navegación y localización de los diferentes componentes del sistema.

2. **Módulos y componentes reutilizables**: Se han creado módulos y componentes en los diferentes sistemas permitiendo un desarrollo modular con buena escalabilidad.

3. **Separación de responsabilidades**: Se ha aplicado el principio de "separación de responsabilidades"para dividir las tareas en diferentes partes del código. Cada componente o módulo tiene una responsabilidad específica y no se mezclan funcionalidades diferentes en un solo lugar.

4. **Comentarios**: Se han incluido comentarios en el código para facilitar su comprensión y explicar la funcionalidad de cada uno de los componentes y sus funciones. Esto facilita el desarrollo para otros desarrolladores que puedan trabajar en el proyecto en el futuro.

8.3. Pruebas

La sección de pruebas es un componente fundamental en el desarrollo de software [30], ya que nos permite garantizar la calidad y fiabilidad del sistema. En esta sección, se describirán los diferentes tipos de pruebas. A través de pruebas rigurosas, se puede identificar y corregir posibles errores o problemas en el funcionamiento del software, asegurando así un sistema robusto y confiable.

Cada prueba estará identificada por un número único y se proporcionarán los siguientes detalles: las precondiciones necesarias para ejecutar la prueba, los parámetros de entrada utilizados, la salida esperada, el escenario que indica los pasos a seguir para ejecutar la prueba, la fecha de la última ejecución y los problemas encontrados durante la prueba

8.3.1. Pruebas Unitarias

Las pruebas unitarias son un tipo de prueba que se enfoca en verificar el correcto funcionamiento de unidades individuales de código, como funciones, métodos o clases, de forma aislada. Dado el alcance del proyecto no se han realizado este tipo de pruebas.

8.3.2. Pruebas de Integración

Las pruebas de integración se centran en comprobar el correcto funcionamiento de la interacción entre diferentes componentes o módulos del sistema. Estas pruebas se realizan para asegurar que las unidades individuales se integren de manera adecuada y que no existan problemas o conflictos en su interacción. Dado el alcance del proyecto no se han realizado este tipo de pruebas en las Tablas 8.1, 8.2 y 8.3.

8.3.3. Pruebas de Sistema

Las pruebas de sistema se llevan a cabo para verificar que todo el sistema funcione correctamente como un conjunto integrado. Estas pruebas se enfocan en evaluar el sistema en su totalidad, incluyendo su arquitectura, funcionalidades y rendimiento. A continuación se describen las pruebas utilizados:

PS1: Inicio de sesión	
Precondiciones	El usuario debe estar registrado en el sistema.
Datos de entrada	Correo electrónico de usuario y contraseña válidos.
Salida esperada	El sistema autentica al usuario y muestra la página principal.
Escenario	1. El usuario ingresa su nombre de usuario y contraseña en el formulario de inicio de sesión. 2. El sistema verifica la autenticidad de las credenciales del usuario. 3. Si las credenciales son válidas, el sistema redirige al usuario a la página principal.
Trazabilidad	Caso de uso: Inicio de sesión
Fecha de ejecución	03-07-2023
Defectos encontrados	Si incio de sesión te da el token JWT no puedes restringir la llamada de inicio de sesión a los usuarios que no lo tengan.

Tabla 8.1: PS1: Autenticación de usuarios

PS2: Creación de usuarios	
Precondiciones	El administrador ha iniciado sesión en el sistem y el nuevo usuario tiene un identificador único en su propiedad.
Datos de entrada	Nombre, correo electrónico y contraseña del nuevo usuario.
Salida esperada	El sistema crea un nuevo usuario y lo almacena en la base de datos.
Escenario	1. El administrador selecciona la opción de creación de usuarios en la interfaz de administración. 2. El administrador ingresa el nombre, correo electrónico y contraseña del nuevo usuario en el formulario. 3. El sistema verifica que los datos sean válidos y únicos. 4. El sistema solicita la lectura del identificador único. 5. El administrador acerca el identificador al lector NFC. 6. El sistema lee y verifica la singularidad del identificador 7. Si los datos son correctos, el sistema crea un nuevo usuario y lo almacena en la base de datos.
Trazabilidad	Caso de uso: Creación de nuevos usuarios
Fecha de ejecución	03-07-2023
Defectos encontrados	No se verificaba la singularidad del identificador NFC

Tabla 8.2: PS2: Creación de usuarios

PS3: Registro de entrada o salida	
Precondiciones	El usuario esta guardado en el sistema.
Datos de entrada	Identificador NFC del usuario (entrada o salida).
Salida esperada	El sistema registra la entrada o salida del usuario en la base de datos.
Escenario	1. El usuario acerca su identificador al lector NFC. 2. El sistema busca el usuario al que le corresponde el identificador. 3. Si el usuario existe, el sistema busca su registro que no tenga salida. 4. Si el usuario tiene un registro sin salida se la asigna, si no, genera un nuevo registro y guarda su entrada.
Trazabilidad	Caso de uso: Registro de entrada y salida
Fecha de ejecución	03-07-2023
Defectos encontrados	Ninguno

Tabla 8.3: PS3: Registro de entrada y salida

8.3.4. Pruebas de Aceptación

Las pruebas de aceptación se realizan para evaluar si el sistema cumple con los requisitos y expectativas del cliente y de los usuarios finales. Estas pruebas se enfocan en validar que el software se ajuste a las especificaciones acordadas y que cumpla con los criterios de aceptación establecidos. El objetivo es asegurar que el sistema satisfaga las necesidades y funcionalidades esperadas, garantizando su utilidad y satisfacción por parte de los usuarios. El tutor académico realizó a fecha 05-07-2023 pruebas en el sistema y dió el visto bueno.

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Conclusiones

Todos los requisitos principales propuestos para este proyecto se han realizado con exito, asi teniendo un sistema de registros mediante NFC con un frontal para comprobarlos estable y escalable.

El objetivo principal que se intentaba lograr con este proyecto era poder obtener un producto final capaz de registrar usuarios a través de los identificadores únicos. A mayores, se habia propuesto otros subobjetivos como poder acceder a una aplicación cliente para verificar todos los registros o implementar una roles de usuario para poder realizar acciones diferenciadas restringidas a esos roles. Todos estos subobjetivos y el objetivo principal se ha logrado completar satisfactoriamente.

Algunos requisitos no funcionales no han podido ser completados debido al alcance del proyecto y las horas planteadas para su realización siendo estos el RNF:4 encriptación de datos y el RNF:8 generación de *backups*.

En primera instacia el alcance del proyecto solo cubría el requisito de realizar los registros de los usuarios sin poder comprobarlos con el frontal, ya que la idea era conectar este proyecto con el actual sistema de registros de la empresa llamado redmind, pero sin este simple frontal, el alcance del proyecto no sería lo suficientemente elevado como para cubrir las horas minimas necesarias para aprobar los creditos del trabajo de fin de grado.

9.2. Trabajo Futuro

A futuro quedaría pendientes realizar las siguientes tareas:

1. Conexión e implementación del sistemas de registros actual de la empresa *Redmind*.

2. Agregar conexiones exteriores para permitir abrir puertas mediante el registro del usuario.

3. Implementación del *RNF4: Encriptación* encriptando datos como las contraseñas o identificadores de los usuarios.

4. Implementación del *RNF8: Generación de backups* creando copias de seguridad de los datos almacenados periodicamente.

5. Escalar el sistema para permitir la conexión de más de un punto de acceso o lector NFC.

Apéndice A

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código: Al ser un TFG de empresa el código se encuentra en un repositorio privado de la empresa. Si se desea ver el código mande una solicitud de acceso al correo miguel.garcia@brooktec.com.

Bibliografía

[1] Rajiv. Applications and future of near field communication. https://www.rfpage.com/applications-near-field-communication-future/, 2022. Accessed: 2023-06-15.

[2] Json web token introduction. https://jwt.io/introduction. Accessed: 2023-06-21.

[3] Autores del manifiesto ágil. Manifiesto Ágil. https://agilemanifesto.org/iso/es/manifesto.html. Accessed: 2023-06-10.

[4] Raspberry pi 3 a+. https://www.kubii.es/los-kits-raspberry-pi-3-y-3-kubii/3587-kit-inicio-oficial-pi-3-a-3272496309500.html. Accessed: 2023-04-05.

[5] Adaptador nfc para raspberry pi 3. https://www.amazon.es/Raspberry-Interface-Communication-Supports-Provided/dp/B07WCRTKSF/ref=sr_1_2?keywords=Waveshare+NFC+HAT+%28PN532%29&qid=1676033014&sr=8-2. Accessed: 2023-04-05.

[6] Astah professional. https://astah.net/pricing/academic/. Accessed: 2023-04-05.

[7] Antiguas. ¿cuanto cuesta cargar la batería del mac al año? https://hablandodemanzanas.com/old/cuanto-dinero-cuesta-cargar-ordenador-portatil-macbook-apple-al-ano-euros/#:~:text=La%20potencia%20(en%20W)%20de,150Wh%20(consumo%20por%20carga, 2017. Accessed: 2023-04-05.

[8] Precios para el alquiler de una oficina en el centro de valladolid. https://www.idealista.com/alquiler-oficinas/valladolid/centro/. Accessed: 2023-04-05.

[9] George Lawton. near-field communication (nfc). https://www.techtarget.com/searchmobilecomputing/definition/Near-Field-Communication, 2022. Accessed: 2023-06-16.

[10] CARLA TARDI. Near field communication (nfc) definition. https://www.investopedia.com/terms/n/near-field-communication-nfc.asp, 2022. Accessed: 2023-06-15.

[11] Adriana Capasso. Caracteristicas de los nfc. https://es.mobiletransaction.org/nfc-significado/, 2022. Accessed: 2023-06-15.

[12] Punith. Uses of nfc technology. https://www.contus.com/blog/uses-of-nfc-technology/, 2022. Accessed: 2023-05-12.

[13] Visure. Qué son los requisitos funcionales. https://visuresolutions.com/es/blog/functional-requirements/, unknown. Accessed: 2023-06-30.

[14] Visure. Qué son los requisitos no funcionales. https://visuresolutions.com/es/blog/non-functional-requirements/#:~:text=Los%20requisitos%20no%20funcionales%20(NFR,la%20confiabilidad%20y%20muchos%20m%C3%A1s., unknown. Accessed: 2023-06-30.

[15] Junta de andalucia. Procedimiento para desarrollar los requisitos de un sistema software que satisfaga las necesidades de negocio. https://www.juntadeandalucia.es/servicios/madeja/contenido/procedimiento/20#:~:text=Requisitos%20de%20informaci%C3%B3n%2C%20describen%20la,cumplimiento%20debe%20velar%20el%20sistema., unknown. Accessed: 2023-06-30.

[16] Craig Larman. *UML y Patrones*. Prentice Hall, 2002. Accessed: 2023-06-08.

[17] Ian Sommerville. *Ingeniería del Software*. Addison Wesley, 2005. Accessed: 2023-06-06.

[18] Unknown. Tutorial de diagrama de secuencia uml. https://www.lucidchart.com/pages/es/diagrama-de-secuencia, 2022. Accessed: 2023-07-01.

[19] Frank buschmann y otros coautores. *Pattern-oriented software Architecture*. Wiley, 1995. Accessed: 2023-06-07.

[20] Unknown. Patrón arquitectónico de capas / layers. https://platzi.com/tutoriales/1248-pro-arquitectura/5439-patron-arquitectonico-de-capas-layers/, 2021. Accessed: 2023-06-25.

[21] Rafael D. Hernandez. The model view controller pattern – mvc architecture and frameworks explained. https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/, 2021. Accessed: 2023-06-02.

[22] Creately. La guía fácil de los diagramas de despliegue uml. https://creately.com/blog/es/diagramas/tutorial-de-diagrama-de-despliegue/, 2022. Accessed: 2023-06-28.

[23] Jim Arlow y Ila Neustadt. *UML 2*. Anaya, 2005. Accessed: 2023-06-15.

[24] Unknown. The a to z of angularjs architecture with its features and facts. https://www.elluminatiinc.com/angularjs-architecture/. Accessed: 2023-06-20.

[25] Características destacables de nodejs. https://desarrolloweb.com/articulos/caracteristicas-nodejs.html, 2013. Accessed: 2023-06-20.

[26] Express. https://expressjs.com/es/. Accessed: 2023-06-21.

[27] Mozilla. Javascript main page. https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript. Accessed: 2023-06-20.

[28] Python main page. `https://www.python.org/`. Accessed: 2023-06-20.

[29] ¿qué es una api de rest? `https://www.redhat.com/es/topics/api/what-is-a-rest-api`, 2020. Accessed: 2023-06-21.

[30] Ron Patton. *Pruebas de Software*. Sams, 2005. Accessed: 2023-05-22.