**PROYECTO**

**DRCARS**

**LOGOTIPO**

**CICLO FORMATIVO DE GRADO SUPERIOR**

**DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

**AUTORES**

**RICARDO ROMERO LOBATO Y MIGUEL BLANCO JIMENEZ**

**TUTOR DEL PROYECTO**

**VICTOR GARCÍA DELGADO**

**Licencia**

**Esta obra está bajo una licencia Reconocimiento-Compartir bajo la misma licencia 3.0 España de Creative Commons. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by-sa/3.0/es/ o envíe una carta a Creative Commons, 171 Second Street, Suite 300, San Francisco, California 94105, USA.**

**Índice de contenido**

[1 INTRODUCCIÓN 5](#_a7wkj9iekl7e)

[2 ALCANCE DEL PROYECTO 6](#_1tf7s5h0dpuf)

[3 ESTUDIO DE VIABILIDAD 7](#_hpgpci7vbs17)

[3.1 Estado actual del sistema 7](#_3fsbmuluh7b3)

[3.2 Requisitos del cliente 7](#_ycyswp1mbobg)

[3.3 Posibles soluciones 7](#_sjm78b4jeh0z)

[3.4 Solución elegida 7](#_jrp9ftwq5a4i)

[3.5 Planificación temporal de las tareas del proyecto [nuevo proyecto] 7](#_545c9b24parg)

[3.6 Planificación de los recursos a utilizar 7](#_2t0of1fxrwjz)

[4 ANÁLISIS 8](#_a27f6qjl4gbk)

[4.1 Requisitos funcionales 8](#_2uz6mhsqaoz4)

[//4.2 Requisitos no funcionales 8](#_1pzwvm5ufqrq)

[5 DISEÑO 9](#_ts1hxmx79ijw)

[5.1 Estructura de la aplicación 9](#_hl4dhq50fxz3)

[5.2 Componentes del sistema 9](#_6azri1ixdfkc)

[5.3 Arquitectura de la red 9](#_e471lo2kvdsi)

[5.4 Herramientas 9](#_28b6p4apnl65)

[6 IMPLEMENTACIÓN 10](#_afebzj29pekw)

[6.1 Entorno de implementación 10](#_fnevihh2uqel)

[6.2 Tablas creadas 10](#_6lvpx632shbt)

[6.3 Carga de datos 10](#_gm9p5l39iiae)

[6.4 Ficheros de configuración actualizados 10](#_45fu9ol2gqq3)

[6.5 Configuraciones realizadas en el sistema 10](#_3dc9keu9gfe5)

[6.6 Implentaciones de código realizadas 10](#_lxprhtb76kvj)

[7 PRUEBAS 11](#_xpgbil1pl1c)

[7.1 Casos de pruebas 11](#_gltuzj3nlass)

7.2 Plan de ejecución…………………………………………………………………………. 11

[8 EXPLOTACIÓN 13](#_lgghjcpzlly9)

[8.1 Planificación 13](#_a3l9g8sk0z8y)

[8.2 Preparación para el cambio 13](#_lrtt2l7zdflm)

[**8.3 Plan de formación 13**](#_7swh36jw44ap)

**(Manual)**

[9 DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN 14](#_knlxa3v8bez1)

[11 FUENTES (Páginas de referencia) 16](#_i5kcj11h5w20)

[12 ANEXOS 17](#_9yiwdzusfhux)

12 CONCLUSIONES

# INTRODUCCIÓN

Este documento recoge el trabajo realizado para el **módulo de Proyecto** del CFGS en

Desarrollo de Aplicaciones Multiplataforma. Este módulo profesional complementa la formación establecida para el resto de los módulos profesionales que integran el título en las funciones de **análisis** del contexto, **diseño** del proyecto y organización de la **ejecución**.

//////////

En este documento se mostrará todo el proceso realizado para el desarrollo de un software de gestión y venta de vehículos de importación para una empresa llamada DRCars.

# ALCANCE DEL PROYECTO

El presente proyecto tiene como **propósito** fundamental desarrollar una solución integral para la gestión de citas y el inventario de vehículos en un servicio de importación, proporcionando dos interfaces diferenciadas y optimizadas para las necesidades de clientes y personal interno.

Se **pretende conseguir** una mejora sustancial en la eficiencia operativa del negocio, agilizando el proceso de solicitud de citas por parte de los clientes y centralizando la gestión de citas, vehículos y usuarios para el personal de la empresa. Esto se traducirá en una mejor experiencia para el cliente, una reducción de la carga administrativa y una mayor capacidad de organización y control para la empresa.

El **objetivo general** del proyecto es el desarrollo e implementación de un sistema multiplataforma, compuesto por:

* Una aplicación web accesible para clientes, que permita la solicitud y gestión autónoma de citas para la compra de vehículos.
* Una aplicación de escritorio para el personal de la empresa, que facilite la gestión integral de citas, la administración del catálogo de vehículos y la gestión de usuarios y empleados.
* Una API RESTful que actúe como intermediario entre las aplicaciones y la base de datos, garantizando una comunicación eficiente y segura.
* Una base de datos MySQL para el almacenamiento persistente de la información (citas, vehículos, usuarios, etc.).
* Una arquitectura basada en contenedores Docker para la API y la base de datos, que asegure la portabilidad y escalabilidad del sistema.
* El uso de ngrok para permitir el acceso externo a la API, superando las limitaciones de una IP dinámica.

El desarrollo de este proyecto se llevará a cabo en varias fases: estudio de viabilidad, análisis, diseño, implementación y pruebas, y explotación o ejecución. Las actividades/tareas/procedimientos de cada una de estas fases se detallarán en las siguientes secciones de este documento.

# ESTUDIO DE VIABILIDAD

Tras realizarse un exhaustivo estudio de viabilidad se ha concluido que el proyecto puede ser realizado sin costes monetarios adicionales. Además, se cuenta con dos desarrolladores en el proyecto por lo que se ha determinado que es posible la entrega del software en el plazo establecido por el cliente.

Para poder llegar a la conclusión de el actual estudio de viabilidad se ha tenido que realizar una valoración de los recursos actuales que dispone el cliente en la actualidad y presentar un estudio de posibles soluciones siendo una de esta elegida por el cliente.

## Estado actual del sistema

El sistema actual del cliente consta de múltiples hojas de calculo (Excel) y cuadernos tradicionales para la gestión interna de su empresa y una página web de mala calidad para el cliente y sin posibilidad de realizar ventas online sobre un catálogo mal estructurado.

Tampoco se cuenta con una seguridad sobre los datos, dado que si alguna hoja de cálculo y/o cuaderno se pierde no existiría redundancia de datos y se podría estar violando la ley de protección de datos (si dicho caso se produjese fuera de las instalaciones).

## Requisitos del cliente

Tras ponernos en contacto con el cliente, se han establecido sus necesidades. La primera necesidad es un software que permita gestionar de forma eficiente, centralizada y segura la lógica del negocio, gestionando así las cuentas del personal, el stock de los vehículos (pudiendo cambiar entre distintos estados, precios, características y más), las citas que soliciten los clientes y las ventas realizadas.   
Por otra parte, el cliente tiene otra necesidad, una remodelación de su página web, realizando una remaquetación completa y desarrollando un UX intuitivo y sencillo. El usuario final debe poder visualizar el catalogo de forma eficiente y rápida, proporcionándole los detalles esenciales como el precio, los kilómetros, el año y el nombre y modelo desde el propio catálogo, si el usuario lo considera debe poder acceder a los detalles específicos de un vehículo y si decide comprarlo, podrá solicitar una cita desde la pagina de detalles del vehículo seleccionado, rellenando un pequeño cuestionario con algunos datos, esta solicitud se mostrara directamente en el software empresarial nombrado anteriormente. El usuario también debe poder solicitar citas relacionadas con otros motivos como solicitar información.

El software debe asegurar una persistencia de datos seguros con validaciones y reglas de seguridad conectando todas las interfaces disponibles la lógica de negocio junto a el servicio de almacenamiento de datos.

## Posibles soluciones

Las soluciones posibles comentadas con el cliente fueron las siguientes:

* Utilizar un software de gestión empresarial ya existente.
* Desarrollar un software completo utilizando una base de datos NoSQL
* Desarrollar un software completo utilizando una base de datos SQL

## Solución elegida

Tras pensar en la mejor opción para el cliente se ha concluido que la opción correcta es desarrollar un software completo utilizando una base de datos SQL.  
El motivo principal de esta decisión es que dado el alcance del negocio y el bajo presupuesto era inviable contratar los servicios de cualquier software empresarial.

El proyecto que se desarrollara constara de 4 partes que formaran un software empresarial completo, persistente y seguro, las distintas partes son las siguientes:

* Lo primero será desarrollar el modelo de datos, ya que es la base del software y el que se encargara de almacenar cualquier dato.
* Lo segundo será desarrollar una API REST que se encargará de procesar los datos entrantes y almacenarlos de forma segura.
* Después continuaremos con la página web con un UX intuitivo y sencillo para que el usuario no se pierda en funcionalidades que no sean necesarias.
* Por último, se desarrollará la aplicación de escritorio desde la que los trabajadores podrán gestionar vehículos, usuarios, citas, ventas y estadísticas.

## Planificación temporal de las tareas del proyecto [nuevo proyecto]

Identificar las tareas de todo el proyecto y decir cuánto tiempo llevará hacerlas. Hacer una estimación de cuántas personas se necesitan para llevar a cabo dichas tareas. Posibilidad de utilizar algún diagrama de Gantt.

## Planificación de los recursos a utilizar

En este proyecto no será necesaria la contratación de ningún empleado, por ende, tampoco hará falta formar personal. No obstante, se necesitará invertir en herramientas de desarrollo de despliegue web y en un servidor dedicado.

Se necesitará invertir también en un domino para que la pagina pueda ser accesible desde la web.

# ANÁLISIS

En esta fase se establecen los requisitos que el sistema debe cumplir.

## Requisitos funcionales

1. Autenticación y autorización:
   * RF 1.1: Los usuarios internos (administradores y empleados) deben iniciar sesión en la aplicación de escritorio.
   * RF 1.2: Los clientes deben poder registrarse y autenticarse en la página web.
   * RF 1.3: La API REST debe validar credenciales
2. Gestión de usuarios internos:
   * RF 2.1: Crear, modificar, eliminar y listar usuarios internos desde la aplicación de escritorio.
   * RF 2.2: Asignar roles (ADMIN, USER) a cada usuario interno.
3. Gestión de clientes:
   * RF 3.1: Registrar nuevos clientes desde la aplicación de escritorio.
   * RF 3.2: Consultar datos de clientes existentes.
4. Gestión de vehículos:
   * RF 4.1: Registrar nuevos vehículos importados, con datos: marca, modelo, año, color, kilometraje, matrícula, chasis, precio de compra, estado, proveedor, tipo de combustible y transmisión.
   * RF 4.2: Actualizar datos de vehículos existentes.
   * RF 4.3: Cambiar estado de vehículos (STOCK → GARAJE → VENTA → VENDIDO).
   * RF 4.4: Consultar catálogo de vehículos en la página web con filtros por marca, modelo, año y precio.
5. Gestión de reservas:
   * RF 5.1: Cliente puede realizar una cita indicando el id de vehículo, fecha y hora, motivo y descripción.
   * RF 5.2: Aplicación de escritorio muestra lista de reservas pendientes.
   * RF 5.3: Empleado puede confirmar o rechazar reservas.
6. Gestión de ventas:
   * RF 6.1: Actualizar estado del vehículo a VENDIDO una vez realizada la venta.
7. Gestión de favoritos (web):
   * RF 7.1: Cliente autenticado puede marcar vehículos como favoritos.
   * RF 7.2: Consultar lista de favoritos del cliente.
8. Reportes y dashboards:
   * RF 8.1: Aplicación de escritorio muestra dashboard con estadísticas de ventas y catálogo.

## Requisitos no funcionales

1. Rendimiento:
   * RNF 1.1: La API REST debe atender hasta 50 peticiones concurrentes sin degradar la respuesta por debajo de 200 ms.
   * RNF 1.2: La página web debe cargar el catálogo de vehículos en menos de 3 segundos con 10.000 registros en la base de datos.
2. Escalabilidad:
   * RNF 2.1: El sistema debe permitir añadir nuevos módulos (piezas) sin reestructurar la base de datos principal.
3. Seguridad:
   * RNF 3.1: Comunicaciones entre componentes deben realizarse sobre HTTPS.
   * RNF 3.3: Los datos sensibles (contraseñas) deben almacenarse cifrados.
4. Usabilidad:
   * RNF 4.1: Interfaces web y de escritorio deben ser intuitivas, con diseño responsive en la web y layout adaptado a escritorio.
5. Mantenibilidad:
   * RNF 5.1: El código debe seguir estándares de estilo (C# para escritorio, Java para API, TypeScript/React para web).
   * RNF 5.2: Documentación (JavaDoc, XML Docs) para facilitar futuras modificaciones.
6. Fiabilidad:
   * RNF 6.1: La base de datos MySQL debe contar con backups diarios.
   * RNF 6.2: Logs de errores y auditoría disponibles en API y escritorio.

# DISEÑO

En esta fase se describe cómo se van a materializar los requisitos establecidos.

## Estructura de la aplicación

La estructura de la aplicación se basa en un modelo vista-controlador.

La estructura del controlador esta compuesta de una base de datos MySQL y de una API REST que se encarga de manejar los datos y formatearlo de forma correcta para su posterior visualización o almacenamiento. Por otra parte tenemos las vistas, que están compuestas de la pagina web y de la aplicación de escritorio, estas vistas se encargan de visualizar los datos y poder interactuar con ellos.

## Componentes del sistema

1. **Base de datos (MySQL 5.7):**

Una base de datos con un conjunto de 11 tablas para poder almacenar toda la información y datos del negocio, en estas tablas se pueden almacenar información de vehículos, usuarios, clientes, etc.

1. **API RESTfull (Spring Boot):**  
   Servidor local encargado de modelar los datos para enviarlos a las vistas o recibirlos y almacenarlos en las tablas correspondientes de la base de datos.
2. **Ngrok:**   
   Encargado de permitir accesos a peticiones externas desde las vistas del servidor dedicado que contiene la API superando las limitaciones que contienen las IP dinámicas.
3. **Vistas:**

Las vistas se componen de una página web y una aplicación de escritorio, estas encargan de recibir los datos y mostrarlos de forma intuitiva para que tanto clientes como empleados puedan consultarlos, modificarlos o eliminarlos.

## Arquitectura de la red

La red contiene la siguiente configuración para cada componente:

* La base de datos esta instalada en un servidor dedicado cuyo puerto ha sido habilitado para recibir y/o proporcionar los datos que requiera el software
* La API, al igual que la base de datos, está instalada en el servidor dedicado, conectado a ngrok, de esta forma se podrán realizar conexiones cifradas al api desde ip’s públicas.
* La página web esta hosteada en un servicio llamado Vercel, este proporciona un servidor público y seguro con un dominio con el que se puede acceder, cuando realiza cualquier conexión cifra los datos sensibles y los envía en formato JSON al servidor, este procesa los datos que recibe y almacena en las tablas correspondientes los datos recibidos.

## Herramientas

MySQL, SpringBoot, ngrok, Servidor privado dedicado, Visual Studio, Vercel.

# IMPLEMENTACIÓN

A partir del diseño establecido, en esta fase se lleva a cabo la construcción del sistema, incluyendo el despliegue y configuración de sus componentes principales.

Por ejemplo, se lleva a cabo la implementación de la página web en el lenguaje que se haya determinado, la creación de las tablas de la BBDD, la carga de datos, se modifican los ficheros de configuración necesarios para que Asterisk cumpla con los requisitos (o se ejecutan los comandos pertinentes y se muestra cuáles son).

[Algunas subfases de implementación dependiendo del tipo de proyecto podrían ser las que aparecen en los siguientes puntos]

## Entorno de implementación

El sistema está alojado en un servidor Ubuntu ubicado en entorno doméstico, el cual contiene tanto la API como la base de datos. Para garantizar una gestión eficiente y portátil de los servicios, se emplea Docker, permitiendo la contenerización de la API y la base de datos, lo que facilita su despliegue y mantenimiento.

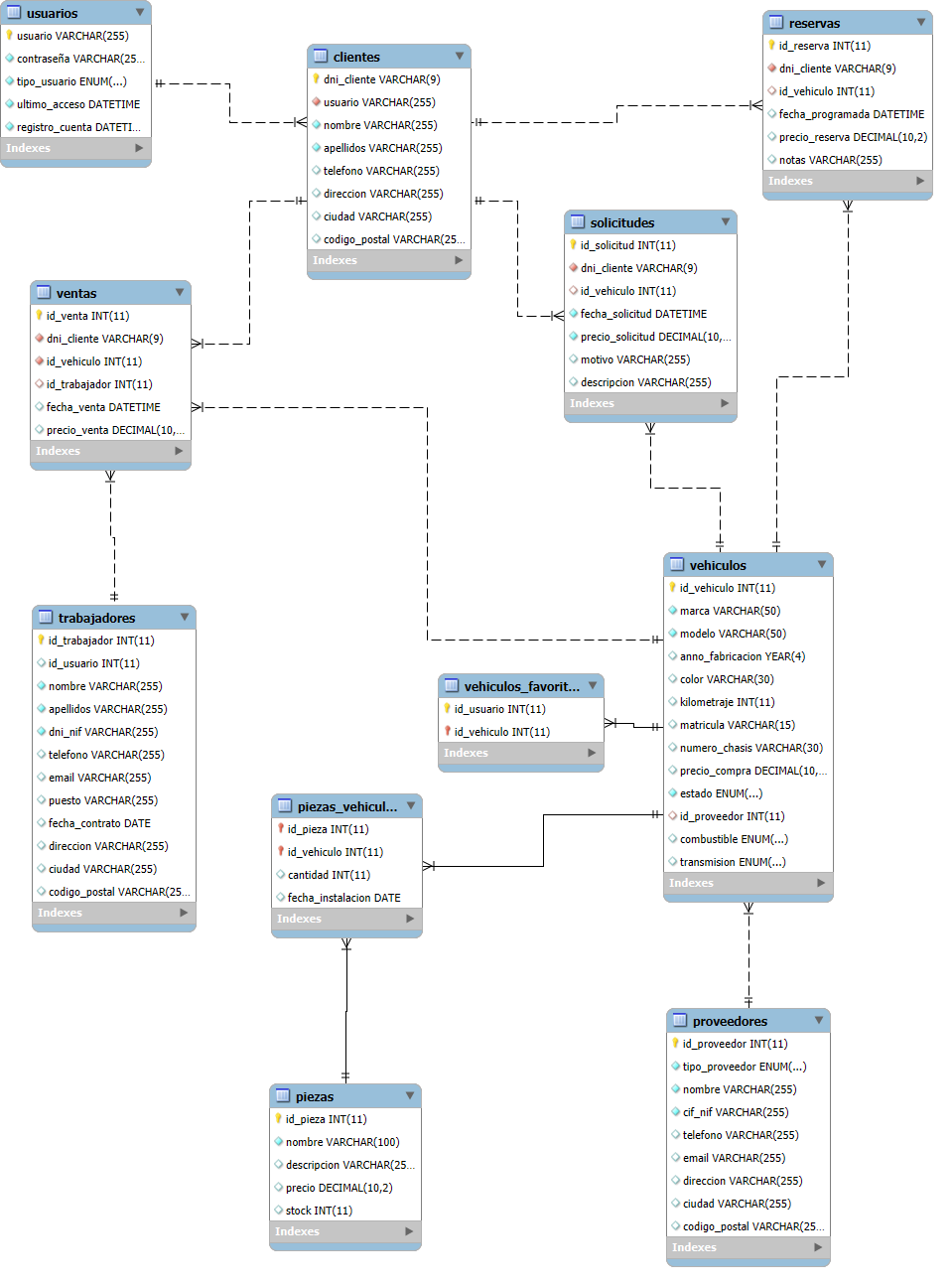
Para que la API pueda ser accedida externamente por los clientes (aplicación de escritorio y web), se utiliza Ngrok, una herramienta que proporciona túneles seguros y expone el servicio en internet. Esto permite que la aplicación de escritorio desarrollada en Visual Studio y la web implementada en Next.js interactúen con la base de datos a través de la API.

Se han desarrollado distintos módulos dentro de la API que permiten gestionar solicitudes de los clientes, realizar validaciones y optimizar el acceso a la base de datos. Se ha priorizado el uso de buenas prácticas de desarrollo, asegurando la correcta modularización y mantenimiento del código.

La aplicación de escritorio y la web consumen la API para modificar y leer datos de la base de datos, garantizando que los cambios sean reflejados de manera inmediata en el sistema.

Este diseño permite una implementación eficiente del sistema, asegurando escalabilidad y facilidad de administración.

## Tablas creadas



Usuarios: Gestiona la autenticación y preferencias.

Clientes: Permite realizar reservas, compras y solicitudes de piezas.

Trabajadores: Encargados de gestionar las operaciones.

Vehículos: Almacena información sobre los modelos disponibles.

Piezas: Relacionadas con los vehículos y proveedores.

Proveedores: Distribuyen piezas y vehículos al negocio.

Reservas: Vinculadas a clientes que desean probar o adquirir vehículos.

Ventas: Registra las transacciones realizadas.

## Carga de datos

Se pueden cargar datos de diferentes maneras:

* Creando citas a través de la web como un cliente
* Añadiendo coches y usuarios desde la app de escritorio como un usuario
* Utilizando la herramienta MySQL Workbench para cargar manualmente los registros
* Scripts sql cargados desde la herramienta previamente dicha

## Ficheros de configuración actualizados

API, Docker y MySQL



App Escritorio



Ngrok



Web



## Configuraciones realizadas en el sistema

Se han realizado configuraciones en Docker para gestionar la persistencia de los datos y optimizar el rendimiento del sistema. Además, la API tiene medidas básicas de seguridad para gestionar el acceso y evitar vulnerabilidades (paquete com.DRCars.config), considerando la posibilidad de uso de certificados SSL para proteger la comunicación con los clientes.

## Implementaciones de código realizadas

Como se ha explicado anteriormente, el código consiste en una app de escritorio junto con una página web que interactúan con una base de datos por medio de una API alojada en un servidor, haciendo peticiones y manejando las respuestas y/o errores.

Enlace al repositorio GitHub: [DRCars\_Code](https://github.com/PdePancetas/TFG_RyM-Code.git)

# PRUEBAS

Son muchas las pruebas que pueden realizarse en un proyecto, para eliminar los posibles errores y garantizar su correcto funcionamiento. Los casos de prueba establecen las condiciones/variables que permitirán determinar si los requisitos establecidos se cumplen o no.

A continuación, se detallan algunos de los casos de prueba que se ejecutarán para comprobar la correcta construcción de este proyecto.

## Casos de pruebas

A modo de ejemplo se facilitan los campos de una posible plantilla **para definir casos de pruebas**, que podrá modificarse como se estime oportuno en función del proyecto y las pruebas que se considere necesario realizar.

* Fecha/autor/[versión a probar]
* Caso de prueba

Identificador del caso de prueba (nombre único)

* Descripción

Breve explicación sobre el objetivo del caso de prueba

* Condiciones de ejecución

Descripción de las condiciones de ejecución que se deben cumplir antes de iniciar el caso de prueba, por ejemplo, que se haya realizado correctamente el login en el sistema...

* Entrada

Datos necesarios para poder ejecutar la prueba

* Resultado esperado

Valor esperado para el correcto funcionamiento del proyecto

* Resultado obtenido

Valor de salida al ejecutar el caso de prueba

* Evaluación

Comparación del valor esperado y obtenido para concluir, finalmente, si el aspecto chequeado por el caso de prueba confirma el correcto funcionamiento del proyecto o comunicación a quien corresponda del error.

A modo de ejemplo se facilitan los campos de una posible plantilla **para reportar los errores** detectados en los casos de prueba ejecutados.

* Fecha/autor/[versión probada ]
* Caso de prueba
* Evaluación

Comparación del valor esperado y obtenido para concluir, finalmente, si el aspecto chequeado por el caso de prueba confirma el correcto funcionamiento del proyecto o comunicación a quien corresponda del error.

* Posible causa de error

Detallar la posible causa o causas que han podido generar el error detectado.

* Posible corrección

Detallar la posible forma de corregir el problema

* Áreas afectadas

Detallar qué áreas (módulos, componentes, documentos.) se verán afectados al implementar la corrección.

# EXPLOTACIÓN

La implantación es la fase más crítica del proyecto ya que el sistema entra en producción, es decir opera en un entorno real, con usuarios reales.

## Planificación

Planificar cómo se va a llevar a cabo la implantación del proyecto en la empresa. Enumerar las tareas a realizar en función de las necesidades de implementación.

Determinar los recursos y la logística necesaria para cada tarea, planificado la asignación de recursos materiales y humanos y los tiempos de ejecución.

Identificar los riesgos inherentes a la ejecución.

## Preparación para el cambio

Considerar las necesidades de permisos y autorizaciones para llevarlas a cabo las actividades.

Tener en cuenta las posibles reticencias al cambio por parte de los usuarios, por lo que conviene determinar claramente los procedimientos de actuación o ejecución de las actividades.

## Plan de formación

Definir y elaborar la documentación necesaria para la formación de los usuarios del sistema. [Se puede incluir como anexo].

## Implantación propiamente dicha

Demostrar de alguna forma que el sistema está implantado, que está en producción.

## Pruebas de implantación

Realización de las pruebas del sistema ya implantado en el cliente.

# DEFINICIÓN DE PROCEDIMIENTOS DE CONTROL Y EVALUACIÓN

A lo largo del ciclo de vida del proyecto se producirán cambios e incidencias que deberán controlarse y registrarse.

A modo de ejemplo se facilitan los campos de una posible plantilla para definir el **procedimiento** de evaluación de las **incidencias** que puedan presentarse durante la realización de las diferentes actividades.

* Fecha/autor/ [ versión probada]
* Identificación incidencia
* Descripción de la incidencia

Comparación del valor esperado y obtenido para concluir, finalmente, si el aspecto chequeado por el caso de prueba confirma el correcto funcionamiento del proyecto o elevación detallada del correspondiente error.

* Posible causa de error

Detallar la posible causa o causas que han podido generar el error detectado.

* Posible corrección

Detallar la posible forma de corregir el problema

* Áreas afectadas

**Detallar qué áreas (módulos, componentes, documentos,) se verán afectados al implementar la corrección.**

A modo de ejemplo se facilitan los campos de una posible plantilla para **gestionar** los posibles **cambios** en los recursos y en las actividades.

# CONCLUSIONES

Breve resumen sobre el trabajo realizado, y si se han cumplido o no los objetivos.

Incluir valoración personal, económica....

# FUENTES

Incluir las páginas web y bibliografía consultada.

# ANEXOS

Cualquier añadido se incluirá en este apartado. Por ejemplo, un manual de usuario.

En el siguiente anexo se incluyen las pautas a seguir a la hora de elaborar la documentación del proyecto.

## Guía de estilo

Obviamente, este anexo deberá ser eliminado del documento final a entregar.

**Título del proyecto**

Elegir un nombre llamativo y relacionado con la temática que va a tratar.

**Figuras y tablas**

Cualquier figura, tabla... incluida en el documento deberá tener un título a pie de página.

Incluir tablas, gráfico, mapas conceptuales...que ayuden a leer y comprender el documento.

**Índices**

Ya incluido en la plantilla. Se puede añadir a continuación un índice de figuras, si se desea.

**Redacción**

Se evitarán las mayúsculas, salvo en los títulos y poco más.

No se emplearán formas personales (instalamos, seleccionamos...) en su lugar se utilizarán formas impersonales (instalar, se instalará, seleccionar, se selecciona,).

Se evitará la voz pasiva (casi siempre traducción literal del inglés). En vez de: es desarrollado para cumplir... mejor: se desarrolla para cumplir...

Se evitarán los párrafos largos.

Se utilizarán las viñetas para facilitar la lectura del documento.

**Formato**

El documento se generará en formato pdf.

**Entrega**

Todo el material del módulo Proyecto (documentos, ficheros fuentes, herramientas...) entregará en formato electrónico, en una carpeta comprimida.

**Lenguajes de Programación:**

**Backend (API):** Java.

**Backend y Frontend (Aplicación de Escritorio):** C#.

**Frontend (Aplicación Web):** TypeScript y React

**Frameworks/Librerías:**

**API (Backend):** Spring Boot y Maven (Desarrollo de la API RESTful).

**Escritorio (Frontend):** .NET Core con WPF/Windows Forms

**Web (Frontend):** React.js, CSS.

**Base de Datos:** MySQL (Gestión y almacenamiento de datos relacionales).

**Contenerización:** Docker (Creación y gestión de contenedores de la API y la base de datos MySQL), y Docker Compose (Orquestación y despliegue de los servicios contenerizados).

**Exposición de API Externa:** Ngrok (Establecer un túnel seguro que permita el acceso remoto a la API desde entornos con IP dinámica).

**Plataforma de Despliegue Web:** Vercel (Despliegue y hosting de la aplicación web).

**Entornos de Desarrollo Integrado (IDE):**

**Aplicación de escritorio:** Visual Studio.

**API:** Eclipse.

**Aplicación web:** Vercel y Visual Studio Code

**Control de Versiones:** Git (Gestión de cambios en el código fuente) y GitHub (Plataforma de alojamiento de repositorios).

**Otras Herramientas:**

Postman (Pruebas y depuración de la API).

MySQL Workbench (para la gestión y administración de la base de datos MySQL).

Herramienta de ingeniería inversa de MySQL Workbench (Diagramas de arquitectura).