MEMORIA ESCRITA DEL PROYECTO

CFGS Desarrollo de Aplicaciones Multiplataforma

**Clever Help Desk**

**Autor:** Victor Garcia Velasco

**Tutor:** Mario Gago

**Fecha de entrega:** dd/mm/aaaa\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Convocatoria:** Segundo Semestre – 2022

**GitHub del proyecto:** <https://github.com/victorgv/CleverHelpDesk>

**Documentos del proyecto:**Enlace a la carpeta del Drive\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Contenido

[1. Introducción 3](#_Toc98882242)

[1.1. Motivación 3](#_Toc98882243)

[1.2. Abstract 3](#_Toc98882244)

[1.3. Objetivos propuestos (generales y específicos) 3](#_Toc98882245)

[2. Metodología usada 4](#_Toc98882246)

[Otros modelos evaluados 4](#_Toc98882247)

[3. Tecnologías y herramientas utilizadas en el proyecto 6](#_Toc98882248)

[4. Estimación de recursos y planificación 7](#_Toc98882249)

[5. Análisis del proyecto 8](#_Toc98882250)

[Requisitos funcionales 8](#_Toc98882251)

[Requisitos no funcionales 9](#_Toc98882252)

[Diagrama Entidad-Relación 9](#_Toc98882253)

[Diagramas Casos de Uso 10](#_Toc98882254)

[CU\_01 LOGIN 10](#_Toc98882255)

[CU\_02 Creación TICKET 11](#_Toc98882256)

[6. Diseño del proyecto 13](#_Toc98882257)

[7. Despliegue y pruebas 14](#_Toc98882258)

[8. Conclusiones 15](#_Toc98882259)

[9. Vías futuras 16](#_Toc98882260)

[10. Bibliografía/Webgrafía 17](#_Toc98882261)

[Anexo A. Glosario 18](#_Toc98882262)

[Anexo N. Manual estilos proyectos JAVA 18](#_Toc98882263)

[IntelliJ IDEA, atajos de teclado 18](#_Toc98882264)

[Anexo N. Manual estilos proyectos DELPHI 18](#_Toc98882265)

***En la normativa de proyectos vigente encontrarás una breve descripción de cada uno de estos apartados para saber qué información debes incluir en ellos***

# Introducción

ss

## Motivación

## Abstract

## Objetivos propuestos (generales y específicos)

# Metodología usada

Debido al tiempo limitado y a que los requisitos están claros desde un principio he decidido utilizar el *Modelo en Cascada* para la realización de este proyecto de software. En este modelo de desarrollo clásico dividimos el ciclo de vida del proyecto en estas 5 fases:

Diagrama

Descripción generada automáticamente

Las principales motivaciones que me han hecho elegir este modelo son:

* Tener poco tiempo para realizar este proyecto (apenas 6 semanas)
* Los requerimientos ya están cerrados y claros
* Fácil de entender, planificar y realizar seguimiento

## Otros modelos evaluados

En la evaluación de qué metodología iba a usar me he interesado por la utilización de las metodologías agiles: *Scrum* y *Programación Extrema (XP)*, en ambos casos es necesario una retroalimentación por parte del cliente e ir realizando ciclos incrementales y refinando en cada uno los requisitos (tanto de ciclos anteriores como añadiendo nuevos requisitos). Evidentemente no voy a tener un cliente que me revise las entregas de cada ciclo, y la realización de varios ciclos aportando en cada entrega valor funcional al producto me habría llevado más tiempo, este ha sido mi principal motivo para descartar estas metodologías.

# Tecnologías y herramientas utilizadas en el proyecto

Herramientas desarrollo:

* IntelliJ IDEA 2021.2.3 (Community Edition)
* Delphi 10.4 Communuty Edition
* Github
* Jira

Herramientas apoyo:

* Microsoft Word
* Microsoft Excel
* [Diagrams.io](https://www.diagrams.net/)

Equipo desarrollo:

* Lenovo T490 Windows 11

# Estimación de recursos y planificación

# Análisis del proyecto

En el proceso de análisis detectamos y plasmamos los requisitos funcionales y no funcionales, así como realizamos el diagrama Entidad-Relación que modelará nuestra BBDD y otros diagramas que pueden nos permitirán ver el alcance de los componentes internos del sistema y las interacciones que va a tener el sistema.

## Requisitos funcionales

1. Se deberá hacer LOGIN para poder utilizar la aplicación CLIENTE.
2. Hay tres niveles acceso ADMIN, AGENTE, USUARIO, se deberá tener el acceso coherente al nivel asignado
3. Se podrán consultar los tickets según el nivel acceso:
   1. ADMIN y AGENTE, pueden ver todos
   2. USUARIO, solo los tickets que haya abierto
4. Creación tickets
5. Asignar un agente a un ticket
6. Cambiar estado/proyecto/tipo ticket
7. Poder adjuntar comentarios de texto al ticket
8. Guardar un pequeño histórico cronológico del ticket, cuando se creo/cambios estados/etc
9. Generación de email con el número de ticket cuando se abra y se cierre
10. Posibilidad de crear un ticket enviando un email a una dirección genérica de la aplicación
    1. Solo creará el ticket si la dirección de email del remitente está registrada como usuario
    2. En caso de no reconocer el email como usuario devolverá un email indicando que no se ha podido crear el ticket indicando que se debe poner en contacto con el administrador
11. Poder dar de alta/baja usuarios (mantenimiento de usuarios)
12. Poder crear proyectos para agrupar los tickets por temática (mantenimiento de proyectos)

## Requisitos no funcionales

1. Poder utilizar la aplicación en S.O. Windows y Android mediante aplicación nativa, realizada con DELPHI/FIREMONKEY
2. Seguridad básica, guardado passwords encriptados y utilizar JWT para autentificación en el servidor REST
3. Creación servidor REST/JSON en JAVA utilizando el framework SPRING BOOT
4. Como BBDD se utilizará ORACLE, pero el lado servidor utilizará el estándar JPA con su implementación HIBERNATE de forma que permita la migración a cualquier otro gestor de BBDD relacional que lo soporte

## Diagrama Entidad-Relación

A continuación se plasma el modelo entidad-relación para facilitar la representación de datos, entidades y relaciones que formarán la base de datos.

Diagrama

Descripción generada automáticamente

Entidades:

* ***User***, usuarios que tendrán acceso a la aplicación
* ***Role***, perfiles de usuario, inicialmente serán: USUARIO, AGENTE y ADMINISTRADOR, aunque permitiría añadir nuevos roles
* ***Ticket***, son las peticiones/incidencias/etc que han abierto los usuarios
* ***Comment***, posibles comentarios que pueda ir introduciendo el usuario que abre el ticket o el agente de lo atiende
* ***Ticket History***, es un histórico cronológico que guarda los eventos más relevantes del ticket. Por ejemplo almacenará cuando se hizo un cambio de estado y quien lo hizo
* ***Project***, diferentes proyectos o áreas en las que luego podremos agrupar los diferentes tickets
* ***Master type***, diferentes tipos de ticket, por ejemplo: consulta, incidencia, extracción datos BD, etc.
* ***Master status***, los posibles estados que puede tener un ticket.

## Diagramas Casos de Uso

A través de las especificaciones y Diagramas de Casos de Uso describiremos el comportamiento que tiene el sistema desde el punto de vista del usuario o los procesos que van a interactuar con él.

### CU\_01 LOGIN

|  |  |  |  |
| --- | --- | --- | --- |
| **Identificador de Caso Uso** | | CU\_01 | |
| **Nombre** | | LOGIN | |
| **Descripción** | | Proceso de validación y acceso en el sistema | |
| **Actores** | | Usuario, Agente, Administrador | |
| **Secuencia normal** | | | |
| Actor | | | Software |
| 1. Introduce su email, password y pulsa “login” | | |  |
|  | | | 2. Se valida que el email/password son válidos |
|  | | | 3. Se accede a la aplicación |
| Excepciones | | | Software |
| Usuario no registrado | | | Muestra mensaje |
| Usuario/password incorrecto | | | Muestra mensaje |
| Usuario/password en blanco | | | Muestra mensaje y posiciona en el campo vacío |
| **CU relacionados** | **[CU\_XX]** | | |
| Precondición | 1. Debe estar dado de alta en la aplicación.  2. Debe rellenar los campos email y password | | |
| Post condición | El usuario puede acceder al sistema | | |

Diagrama

Descripción generada automáticamente

### CU\_02 Creación TICKET

|  |  |  |  |
| --- | --- | --- | --- |
| **Identificador de Caso Uso** | | CU\_02 | |
| **Nombre** | | Creación de un TICKET | |
| **Descripción** | | Proceso de creación tickect | |
| **Actores** | | Usuario, Agente, Administrador y Email | |
| **Secuencia normal** | | | |
| Actor | | | Software |
| 1. EMAIL: se recibe un email  USUARIO: crea un ticket desde el formulario | | |  |
|  | | | 2. Se emite un email con el número de tickect como confirmación |
| Excepciones | | | Software |
| Email no existe, ocurre cuando recibe un email y este no está registrado como usuario | | | Se emite un email de respuesta indicando que no está registrando y debe ponerse en contacto con el administrador |
| **CU relacionados** | **CU\_01** | | |
| Precondición | 1. debe estar registrado el email  2. Si es usuario, agente o administrador debe estar logeado | | |
| Post condición | El usuario puede acceder al sistema | | |

CU\_03 Consulta TICKET/Adjuntar comentarios

CU\_04 Gestión del TICKET

CU\_05 Administración del sistema

Diagrama de Clases

# Diseño del proyecto

Diseño detallado:

* Diagramas de clases
* Diagramas de secuencia
* Especificación de métodos
* Modelos relacionales
* Diseño interfaces

Elementos a tener en cuenta:

* Como se hará la persistencia de los datos
* Seguridad
* Manejo de errores
* Iteracción y presentación

# Despliegue y pruebas

# Conclusiones

# Vías futuras

# Bibliografía/Webgrafía

* <https://docwiki.embarcadero.com/>
* <http://spring.io/>
* <https://openwebinars.net> Curso de Spring Core

# Anexo A. Glosario

A continuación se incluye la terminología y siglas utilizadas en la elaboración de esta memoria.

# Anexo N. Manual estilos proyectos JAVA

dd

## IntelliJ IDEA, atajos de teclado

|  |  |  |
| --- | --- | --- |
| Acción | Atajo teclado | Comentario |
| Formatear código | CTRL+ALT+L | También botón derecho en navegador ficheros seleccionando *Reformat code*  Otra opción más avanzada CTRL+ALT+SHIFT+L |
| Acciones sobre código | ALT+ENTER | Para añadir imports o resolver errores |
| Generar getters, setters, constructors, etc | ALT+INSERT | También botón derecho sobre el código opción *Generate* |
| Duplicar línea | CTRL+D |  |

# Anexo N. Manual estilos proyectos DELPHI

Ss

