

CIFP N°1 CEUTA

Proyecto Final de Grado

Gestor REI (Recursos E Inventario)

Álvaro García Salvatierra

2º Desarrollo Aplicaciones Web

CURSO 2024/2025

Resumen:

La idea principal de este proyecto es crear un sistema de inventariado, tareas, gestión de usuarios al mismo tiempo, la idea de este proyecto llega desde la poca optimización de estos procesos a nivel local en varias instituciones publicas. Este proyecto ayudaría a optimizar los recursos y la agilización de las incidencias.

Palabras Clave: Gestor, Inventario, Tareas, Inventariado, Técnicos, Usuarios, Gestión Administrativa, Incidencias

Abstract:

The main idea of this project was to create a Inventory system that could assign tasks to technicians and manage users all at the same time. This whole idea comes from the lack of optimization in public institutions. This project would help optimize the resources and overall facilitate troubleshooting.

Keywords: Gestor, Inventory, Tasks, Technician, Users, Administrative Gestion, Incidences, Troubleshooting, Manager, Manage

Indice

- 1. Introducción**
- 2. Objeto del proyecto**
- 3. Objetivos**
- 4. Alternativas**
- 5. Analisis DAFO**
- 6. Requisitos Funcionales**
- 7. Requisitos No Funcionales**
- 8. Casos de uso**
- 9. Diagrama de Gantt**
- 10. Diseño de la base de datos**
- 11. Diseño de interfaces**
- 12. Desarrollo en entorno servidor**
- 13. Desarrollo web en entorno cliente**
- 14. Conclusión**

Introducción

La idea de este proyecto proviene de la falta de optimización de recursos y tratamiento de incidencias y otros problemas en las instituciones publicas que hemos notado a lo largo del tiempo.

Gracias a este proyecto tenemos la idea de poder crear un programa con el que agilizar un poco este proceso, así ayudando a que las incidencias se resuelvan mas fácilmente y tener un inventariado limpio y organizado.

Objeto del proyecto

El “Gestor REI” es un sistema de gestión utilizado principalmente para levantar incidencias, gestionar inventario y poder recoger datos sobre las tareas técnicas del trabajo (como el rendimiento de cada técnico en el trabajo).

Gracias a este sistema cualquier usuario puede levantar una incidencia y asignarla a sus empleados/subordinados o cualquier técnico de la institución.

Objetivos

- Agilizar proceso de creación de incidencias.
- Crear objetos en inventario.
- Manejar inventariado general.
- Crear un sistema de jerarquías entre usuarios para distinguir jefes y empleados.
- Crear un sistema de asignación para asignar incidencias a los técnicos.

ALTERNATIVAS

La alternativa principal es GLPI es un sistema bastante completo.

Ventajas:

- Nuestro sistema es más modificable en su enteridad.
- Nuestro sistema incluye un sistema de jerarquías para declarar que usuario es jefe de que técnico.
- Nuestro sistema trabaja a nivel local, por lo cual los administradores del sistema a nivel local pueden tener libre albedrío para modificar lo que quieran cuando quieran
- Recogida de datos de técnicos. Gracias a estos datos podemos ver el rendimiento de los técnicos.

Desventajas:

- Nuestro un sistema basado en permisos, por lo cual tiene más restricciones a la hora de asignar incidencias.
- Nuestro sistema no tiene distintos tipos de objetos, sino que usa una plantilla a la hora de crearlos.

Análisis DAFO

Debilidades		Amenazas
<p>Alta cantidad de información entrante, y necesidad de sincronización constante, lo que puede generar</p>		<p>Ya existe un competidor a nivel estatal.</p>
Fortalezas		Oportunidades
<p>Un sistema con mayor comunicación a nivel local que nuestros competidores.</p> <p>Un sistema más modificable y al gusto del administrador de cada centro.</p>		<p>El principal competidor no <u>actúa</u> a nivel local, solo a nivel estatal, por lo cual este sistema se podría usar por los institutos locales.</p>

REQUISITOS FUNCIONALES

RF-1 Gestión de usuarios

- RF1.1 Registro Usuarios
- RF1.2 Dar baja usuarios Usuarios
- RF1.3 Modificar datos del usuario
- RF1.4 Autenticar la sesión del usuario
- RF1.5 Cierre de sesión
- RF1.6 Sistema de privilegios
- RF1.7 Panel de administrador
- RF1.8 Gestión de los datos de los usuarios en BD
- RF1.9 Crear jerarquía varios usuarios

RF-2 Tareas

- RF2.1 Registrar tareas
- RF2.2 Eliminar tareas
- RF2.3 Cambiar estados tarea
- RF2.4 Modificar y responder a la tarea
- RF2.5 Asignar tarea a usuario
- RF2.6 Dar nombre a la tarea
- RF2.7 Dar una descripción a la tarea
- RF2.8 Dar un tiempo estimado de la tarea

RF-3 Institución

- RF3.1 Crear instituciones (o centros)
- RF3.2 Dar de baja instituciones

RF-4 Inventario

- **RF4.1** Asignar Inventario a Institución

RF-5 Gestión Objetos

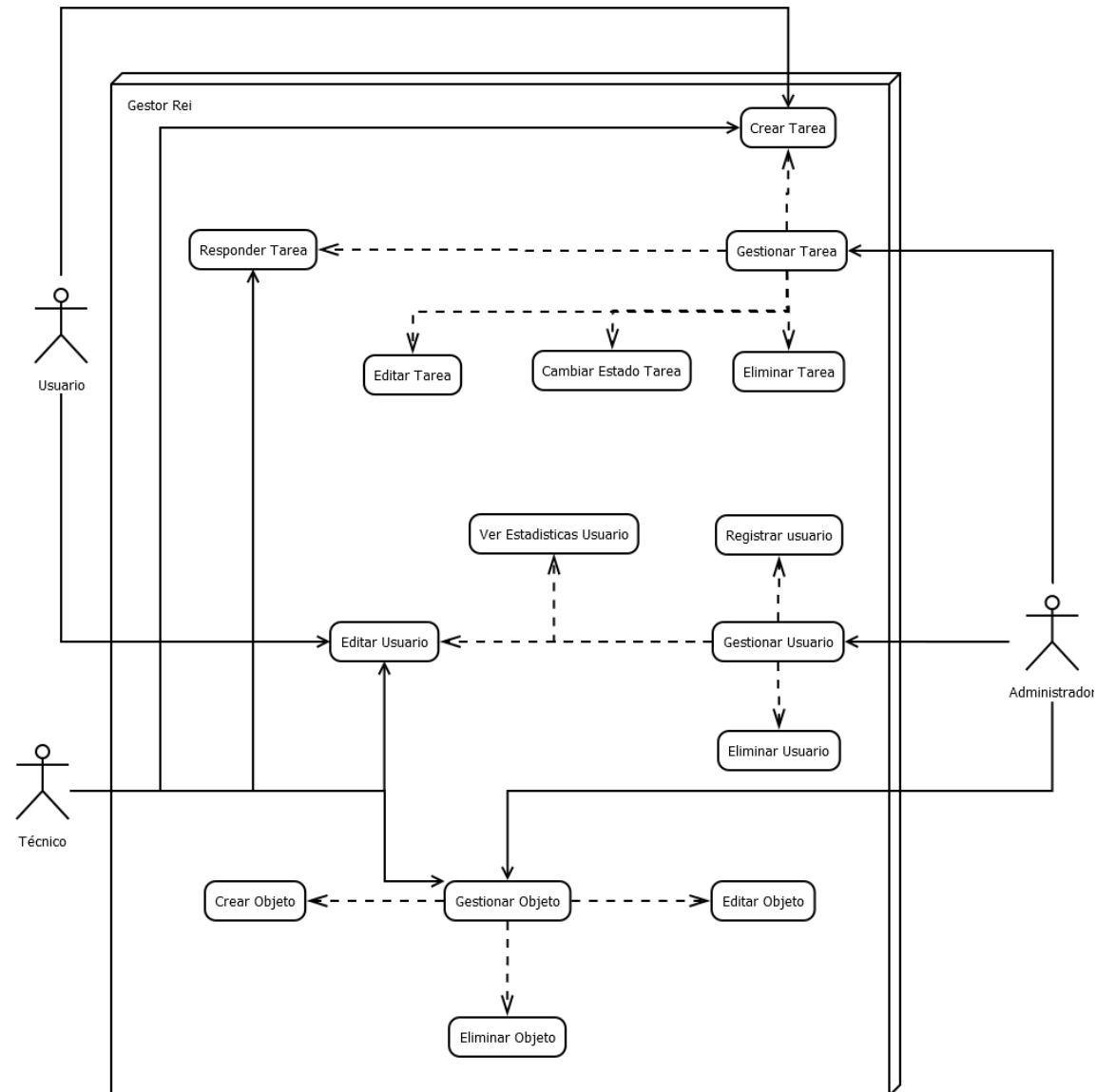
- **RF5.1** Crear Objetos
- **RF5.2** Dar de baja Objetos
- **RF5.3** Dar estado a objeto (En inventario, de baja, desactivado, etc..)
- **RF5.4** Dar nombre a Objetos
- **RF5.5** Dar descripción a objeto
- **RF5.6** En caso de avería dar descripción de la propia

REQUISITOS NO FUNCIONALES

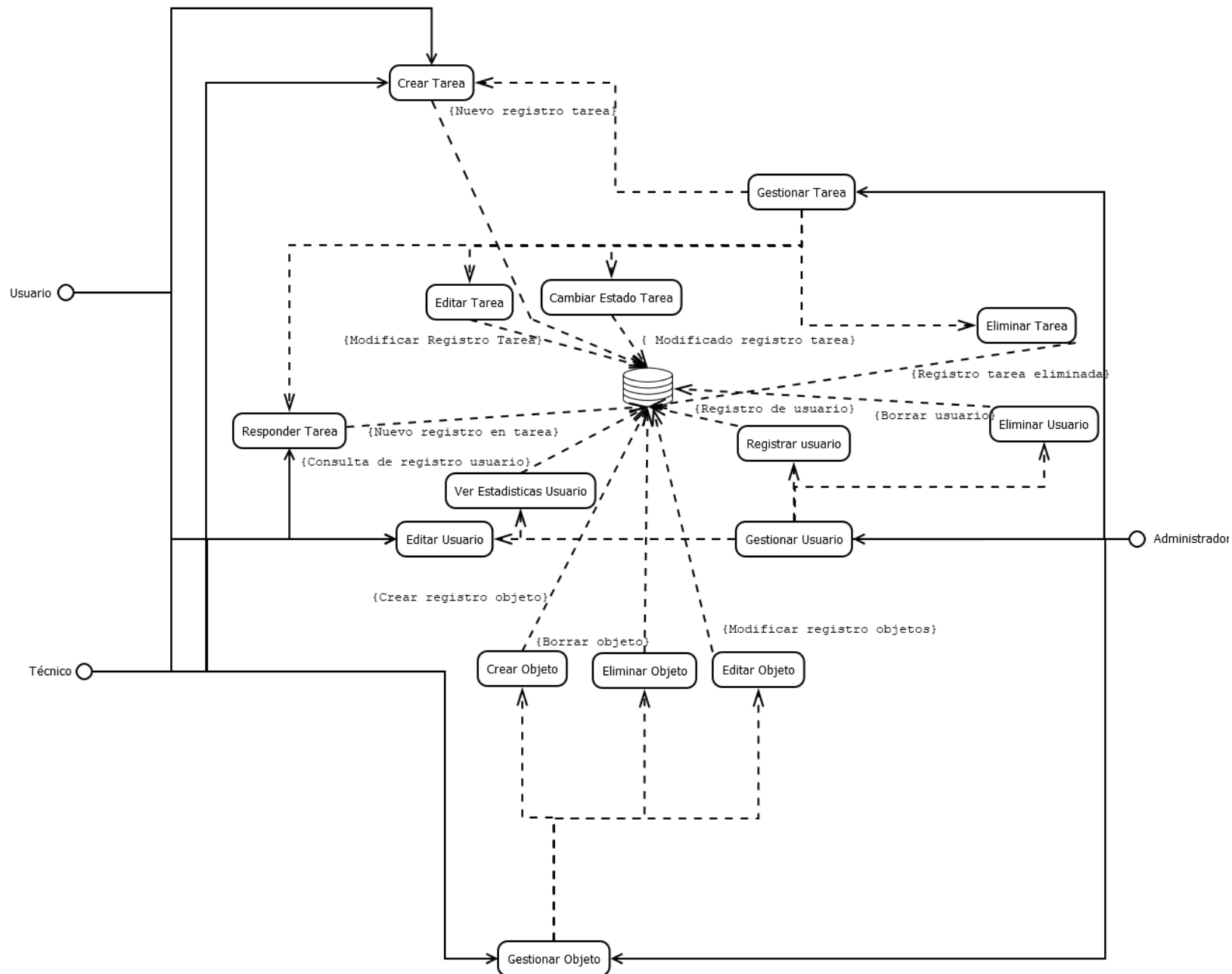
- **RNF1.1** Crear Promedios de los datos del usuario (tareas completadas, etc)
- **RNF1.2** Gestión de expresiones regulares.
- **RNF1.2** Gestión seguridad del usuario.
- **RNF2.1** Crear sistema de respuesta de asignado a creador

CASOS DE USO

Diagrama UML:

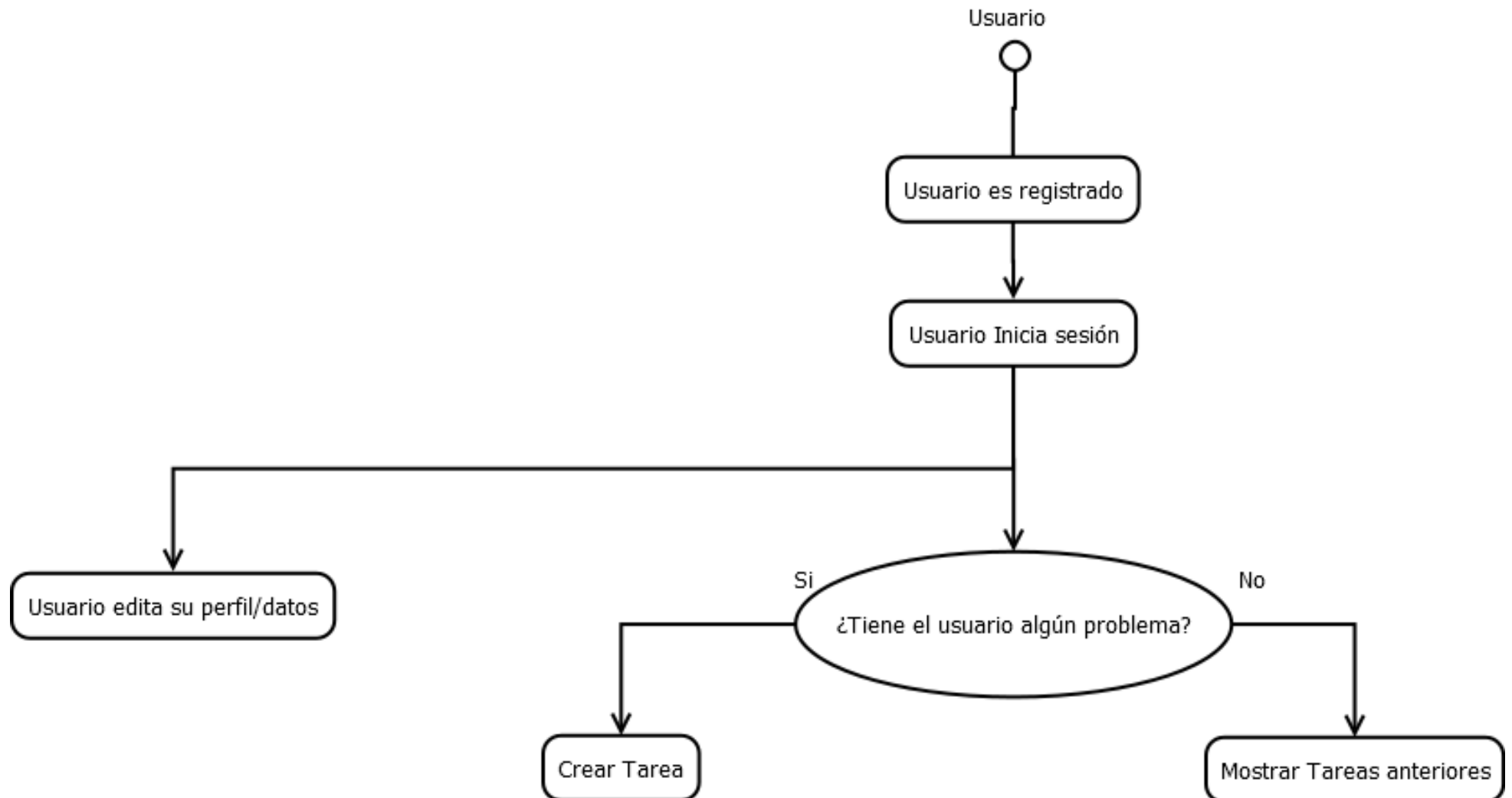


Diagramas de flujo de datos:



Diagramas de caso de uso

En este caso hay 3 diagramas, 1 por cada tipo de usuario que va a tener la aplicación.



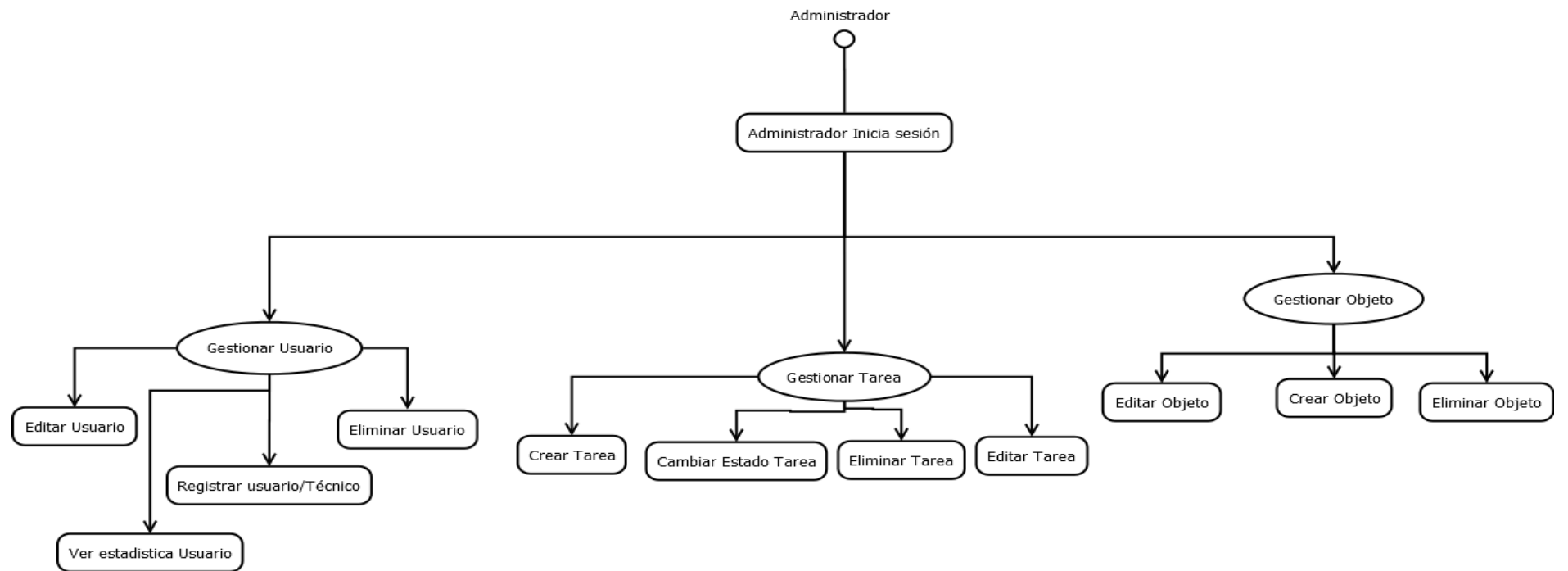
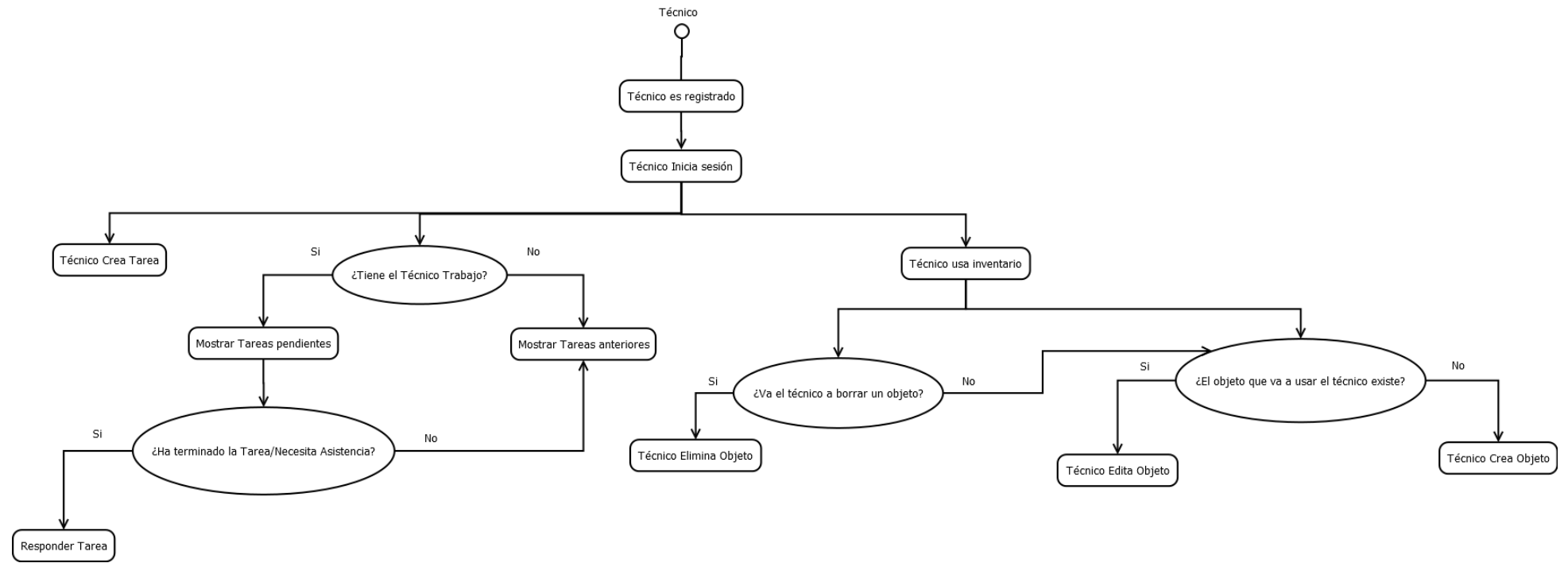
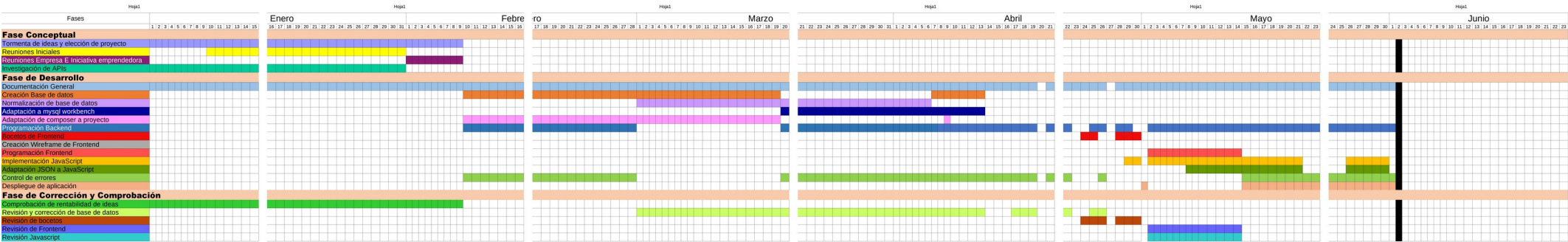
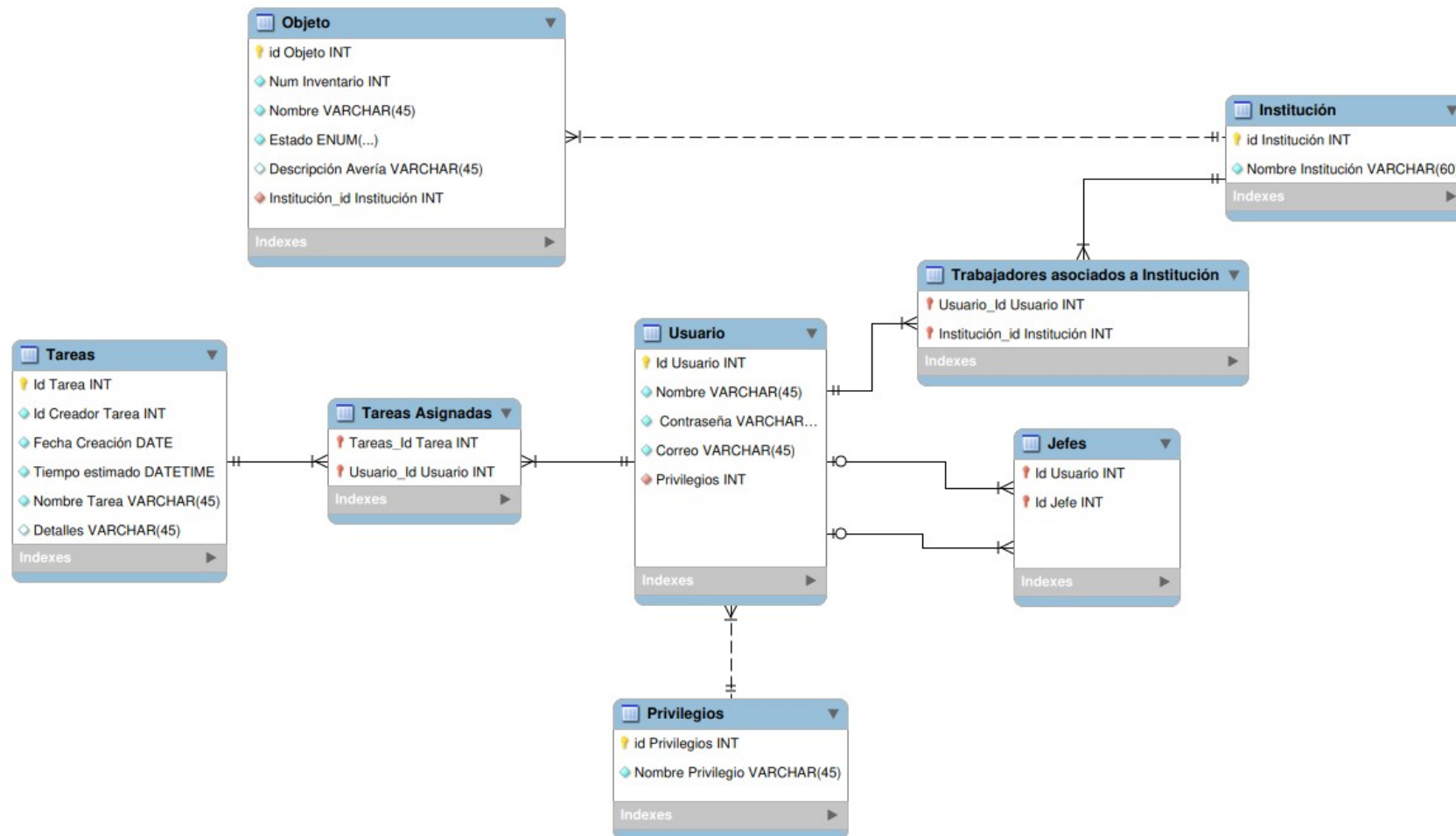


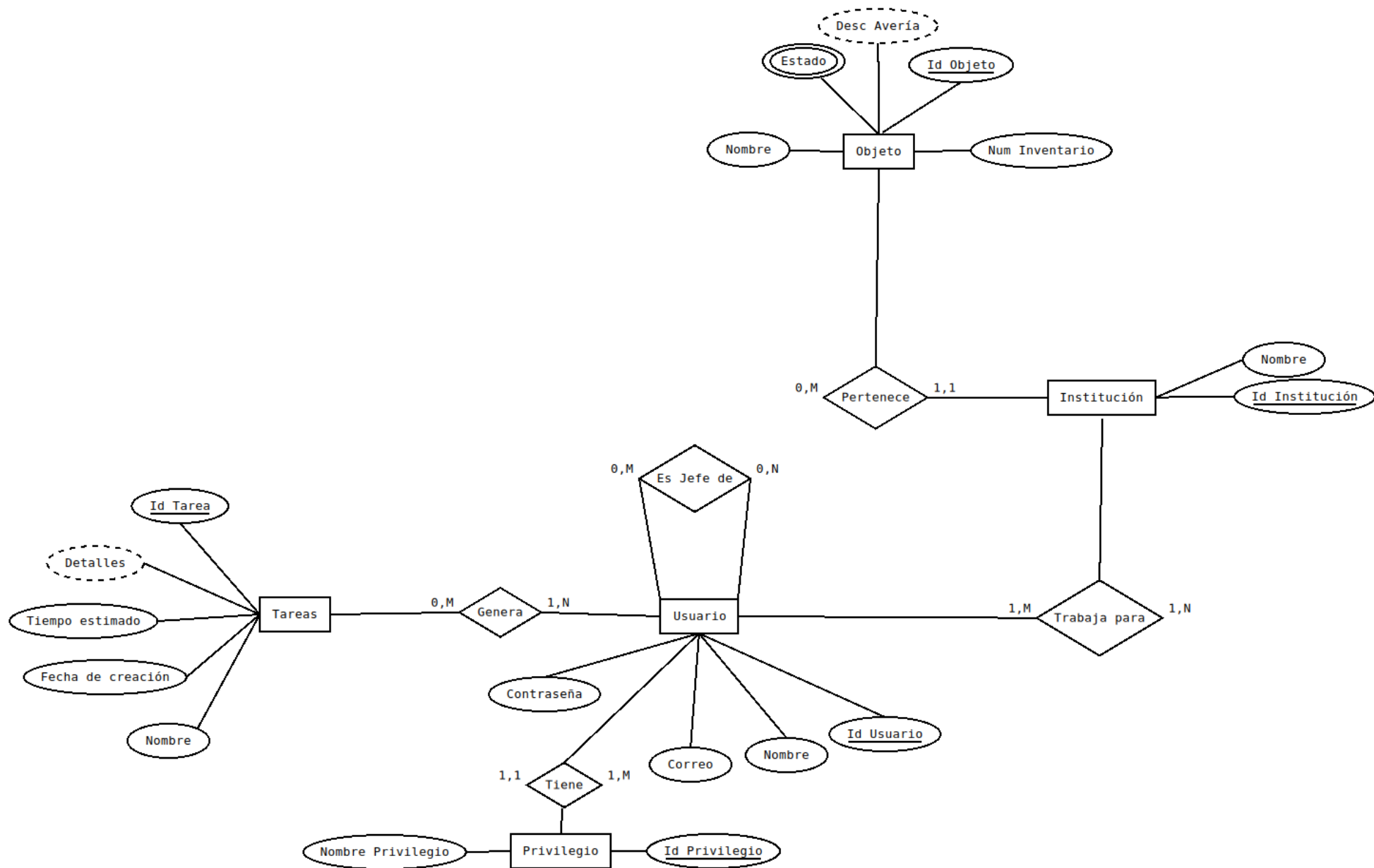
DIAGRAMA DE GANTT

Este documento es extremadamente grande, consultar foto conjunta o pdf adjunto en la documentación.



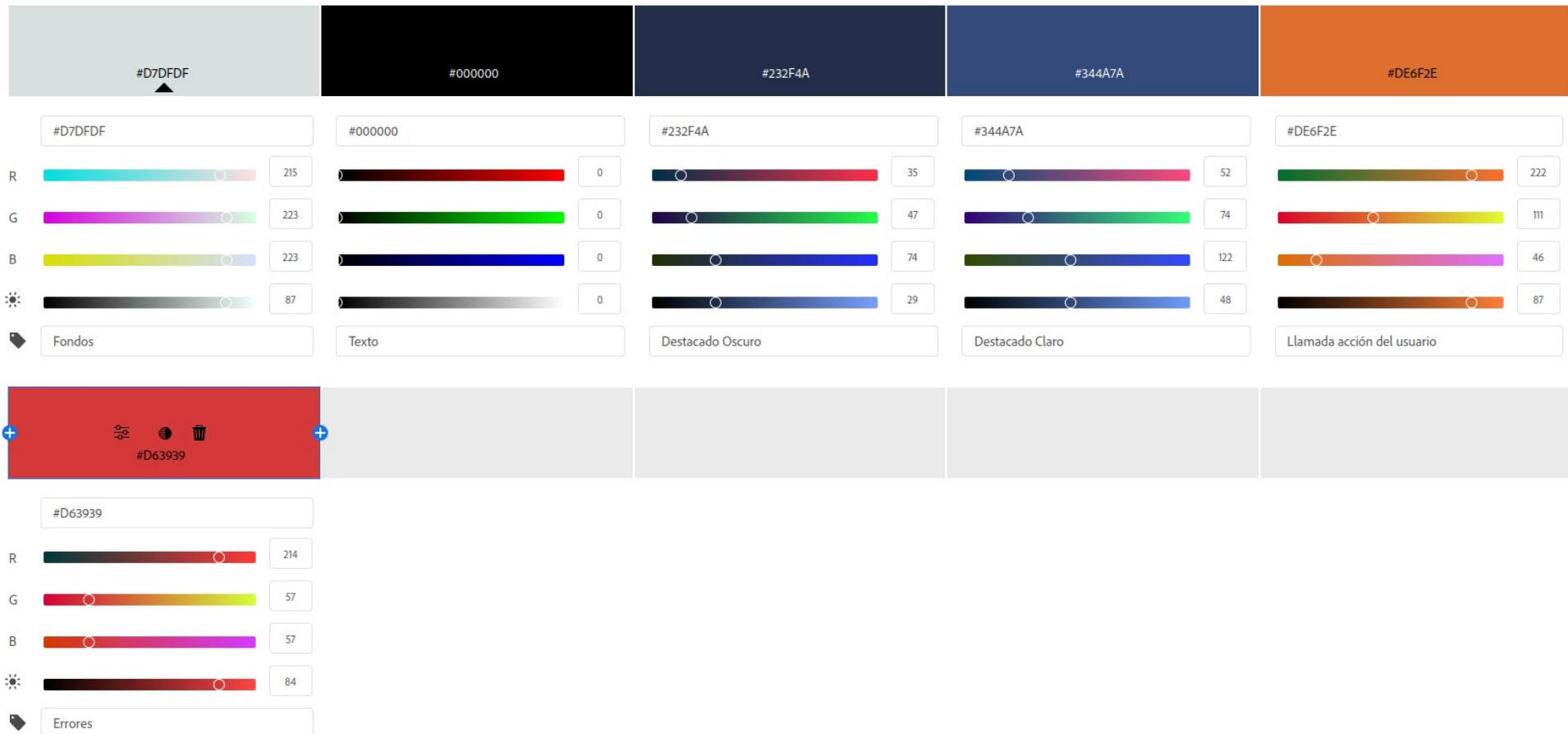
DISEÑO DE LA BASE DE DATOS





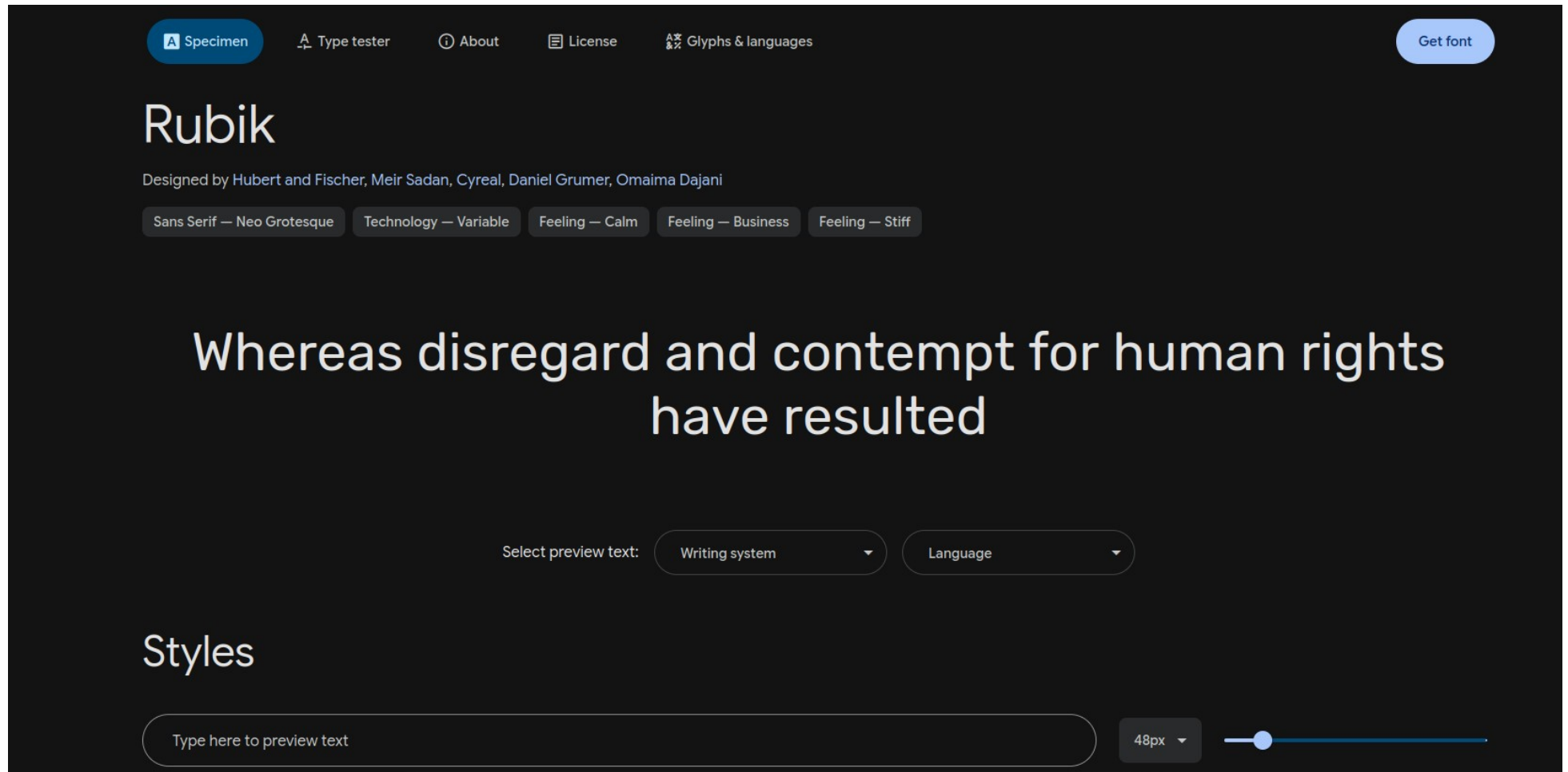
DISEÑO DE INTERFACES

Paleta de colores



El gris tiene un par de tonos oscuros para evitar la carga visual, los 2 azules se utilizan para destacar los elementos de las interfaces y la versión más clara para resaltar los elementos importantes al usuario, el naranja para resaltar los elementos extremadamente importantes del usuario y el rojo para los errores o las acciones negativas (como errores o eliminar cosas).

Tipografía



He escogido la letra Rubik debido a la apariencia de letra “Informativa” que tiene a la par de ser sencilla a la vista.

Iconos y otros elementos:

Todos los iconos son importados de Fontawesome.com.

Wireframes:



Título

Detalles de Tarea

Nombre Tarea

Nombre de Tarea

Estado:

Estado Tarea

Descripción Avería:

Descripción de la avería

Empleados Asignados:

Usuario 1

Usuario 2

Usuario 3

Título

Creación de Tarea

Nombre Tarea:

Nombre de Tarea

Estado:

Elige un estado

Descripción de la Tarea:

Empleados:

<input type="checkbox"/> Usuario 1	<input type="checkbox"/> Usuario 7	<input type="checkbox"/> Usuario 13
<input type="checkbox"/> Usuario 2	<input type="checkbox"/> Usuario 8	<input type="checkbox"/> Usuario 14
<input type="checkbox"/> Usuario 3	<input type="checkbox"/> Usuario 9	<input type="checkbox"/> Usuario 15
<input type="checkbox"/> Usuario 4	<input type="checkbox"/> Usuario 10	<input type="checkbox"/> Usuario 16
<input type="checkbox"/> Usuario 5	<input type="checkbox"/> Usuario 11	<input type="checkbox"/> Usuario 17
<input type="checkbox"/> Usuario 6	<input type="checkbox"/> Usuario 12	<input type="checkbox"/> Usuario 18

Crear Tarea

Título

Creación de Objeto

Nombre Objeto:

Nombre de Objeto

Estado:

Elige un estado

Descripción Avería:

Descripción de la avería

Crear Objeto

Título

Usuarios

Buscar Usuario

Crear Usuario

Usuario 1	Tipo Usuario	Correo Usuario	Editar	Eliminar
Usuario 2	Tipo Usuario	Correo Usuario	Editar	Eliminar
Usuario 3	Tipo Usuario	Correo Usuario	Editar	Eliminar
Usuario 4	Tipo Usuario	Correo Usuario	Editar	Eliminar
Usuario 5	Tipo Usuario	Correo Usuario	Editar	Eliminar
Usuario 6	Tipo Usuario	Correo Usuario	Editar	Eliminar

Título

Registro de Usuario

Nombre Usuario:

Nombre de Usuario

Tipo de usuario

NO EDITAR TIPO USUARIO

Contraseña Usuario:

Contraseña Usuario

Repetir Contraseña Usuario:

Repetir Contraseña

Correo usuario:

NO EDITAR CORREO

Registrar Usuario

Título

Registro de Usuario

Nombre Usuario:

Nombre de Usuario

Tipo de usuario

▼ Elige el tipo de usuario

Contraseña Usuario:

Contraseña Usuario

Repetir Contraseña Usuario:

Repetir Contraseña

Correo usuario:

Correo usuario

Registrar Usuario



Título

Asignar Empleados

Encargado:

Buscar Usuario

Elegir un encargado

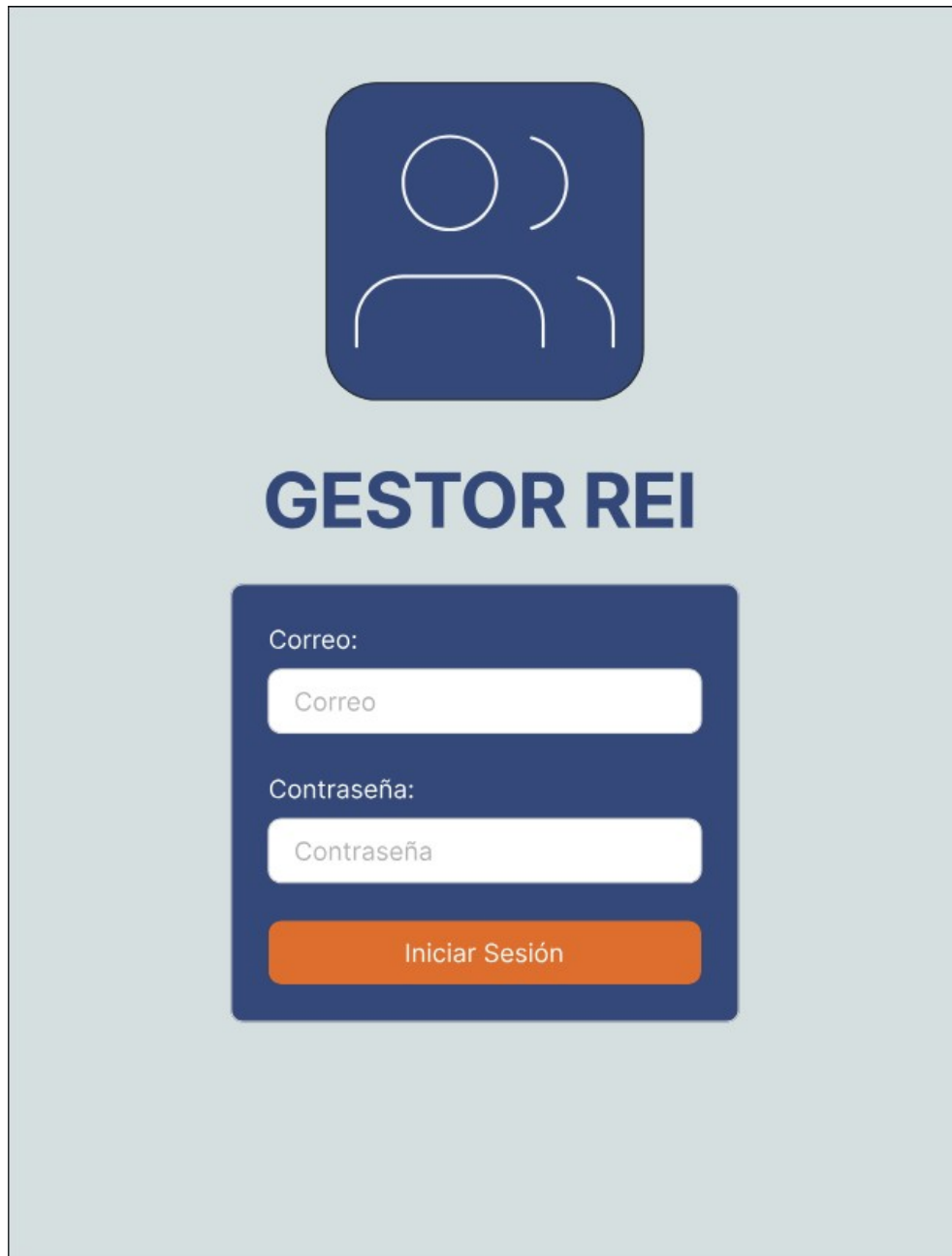
Buscar Usuario

Empleados:

<input type="checkbox"/> Usuario 1	<input type="checkbox"/> Usuario 7	<input type="checkbox"/> Usuario 13
<input type="checkbox"/> Usuario 2	<input type="checkbox"/> Usuario 8	<input type="checkbox"/> Usuario 14
<input type="checkbox"/> Usuario 3	<input type="checkbox"/> Usuario 9	<input type="checkbox"/> Usuario 15
<input type="checkbox"/> Usuario 4	<input type="checkbox"/> Usuario 10	<input type="checkbox"/> Usuario 16
<input type="checkbox"/> Usuario 5	<input type="checkbox"/> Usuario 11	<input type="checkbox"/> Usuario 17
<input type="checkbox"/> Usuario 6	<input type="checkbox"/> Usuario 12	<input type="checkbox"/> Usuario 18

Asignar Empleados

Mockup:



A login screen for 'GESTOR REI'. At the top is a dark blue rounded square icon containing a white outline of a person. Below the icon is the text 'GESTOR REI' in a bold, dark blue font. Underneath is a dark blue rounded rectangle containing two white input fields. The first field is labeled 'Correo:' and the second is labeled 'Contraseña:'. Both fields have placeholder text 'Correo' and 'Contraseña' respectively. At the bottom of this rectangle is an orange button with the text 'Iniciar Sesión'.



A dashboard for 'Gestor REI'. The top header is dark blue with a white star icon in an orange circle on the left and the text 'Gestor REI' in white. Below the header is a light blue section titled 'Tareas' in dark blue. In the top right of this section is an orange button with the text 'Crear Tarea'. Below the title is a dark blue bar with a white downward arrow and the text 'Tareas Pendientes'. Underneath is a search bar with the text 'Buscar Tarea' and an orange magnifying glass icon. Below the search bar are two task cards. Each card has a title 'Tarea 1' and 'Tarea 2', a label 'Descripción:' followed by a white input field with placeholder text 'Descripción', and a label 'Asignado Por:' followed by the text 'Usuario'. At the bottom of each card is an orange button with the text 'Revisar'. Below the task cards is a dark blue bar with a white rightward arrow and the text 'Tareas Completadas'.



Gestor REI



Crear Tarea

Nombre de la tarea:

Estado de la tarea:

Descripción de Tarea:

Asignar Tarea:



Crear Tarea



Gestor REI



Objetos

Buscar Objeto



Crear Objeto

Objeto	Estado	Descripción Avería	Editar	Eliminar
Objeto	Estado	Descripción Avería	Editar	Eliminar
Objeto	Estado	Descripción Avería	Editar	Eliminar
Objeto	Estado	Descripción Avería	Editar	Eliminar
Objeto	Estado	Descripción Avería	Editar	Eliminar



Gestor REI



Detalles de Objeto

Nombre del objeto:

Estado del objeto:

Descripción Avería:



Gestor REI



Creación de Objeto

Nombre del objeto:

Estado del objeto:

Descripción Avería:

Crear Objeto



Gestor REI

Usuarios

Buscar Usuario



Registrar Usuario

Usuario	Tipo Usuario	Correo Usuario	Ver Stats.	Editar	Eliminar
Usuario	Tipo Usuario	Correo Usuario	Ver Stats.	Editar	Eliminar
Usuario	Tipo Usuario	Correo Usuario	Ver Stats.	Editar	Eliminar
Usuario	Tipo Usuario	Correo Usuario	Ver Stats.	Editar	Eliminar
Usuario	Tipo Usuario	Correo Usuario	Ver Stats.	Editar	Eliminar



Gestor REI

Editar Usuario

Nombre del Usuario:

Nombre

Tipo de Usuario:

No editable

Contraseña del Usuario:

Contraseña

Repetir Contraseña:

Contraseña

Correo Usuario:

No editable

Editar Usuario





Gestor REI



Crear Usuario

Nombre del Usuario:

Tipo de Usuario:

Contraseña del Usuario:

Repetir Contraseña:

Correo Usuario:

Registrar Usuario



Gestor REI



Asignar Empleados

Encargado:



Empleados:



Asignar Empleados



Gestor REI



Gestor REI





Gestor REI



Gestor REI

Tareas



Objetos



Usuarios



Permisos



Cerrar Sesión





Gestor REI

Tareas



Objetos



Usuarios



Permisos



Cerrar Sesión



DESARROLLO EN ENTORNO SERVIDOR

USO POO:

En el ámbito de objetos he desarrollado 4 clases con sus respectivos controladores, además de esto también he empleado un emptymodel con las funciones básicas de todas las clases (como hacer consultas a la base de datos), además de otros complementos como traits, herencias y métodos estáticos.

USO PDO:

El pdo utiliza una clase independiente principal “Database” la cual está vinculada con el emptymodel, cada modelo se encarga de usar una función del emptymodel usada para realizar consultas configurable con las tablas del modelo en cuestión y consultas preparadas, así haciendo que cada clase realiza la consulta, la envía usando el modelo de la base de datos que está vinculado a una función del emptymodel y luego el modelo que realizó la consulta recoge los objetos y los formatea para su envío a la función necesaria.

USO MVC:

El modelo vista controlador funciona utilizando un sistema de controlador frontal con un sistema de modelo vista controlador tradicional, esto ayuda al procesamiento de la página, facilita la programación y ayuda a la detección de errores y al aislamiento de fallos.

Modelos Planteados:

Los modelos planteados son Usuario (se encarga de toda la parte del usuario, como el registro de los propios, recogida, dar de baja), Objeto (Objetos del inventario, estados en el que se encuentran, dar de alta y de baja), Tarea (Creación de tarea, modificación de la propia, formateado de la fecha para registrarlas en la base de datos) y las Instituciones (Dar de alta y de baja las instituciones).

Los usuarios son las personas que usarán la plataforma, desde el usuario promedio, al técnico, al admin e incluso al dueño/creador del sistema. Las tareas son labores que los usuarios de mayores privilegios (cualquiera que sea admin [no cuenta el creador del sistema u owner] y los técnicos), normalmente estas constan de labores técnicas en su mayoría. Los objetos son “pertenencias” de la institución, normalmente usadas en el ámbito de trabajo. Las instituciones son la entidad en la que están registrados los objetos y los usuarios, siendo estos cuerpos privados u públicos.

Controlador frontal y controladores:

Cada modelo consta de su propio controlador en caso de recogida de datos y modificación del formato de los propios para mostrarlos en la vista, además de funcionar como capa extra de seguridad. Y el controlador frontal consta del índice, que se dedica a recoger otros controladores, modelos y extras (como la clase de seguridad), para procesar información y mostrar el resultado de la lógica.

USO DAO:

El DAO en este caso consta de una mezcla del modelo “database” que se encarga de establecer las conexiones a cada tabla necesaria en el momento de la consulta y a la propia base de datos, y el emptymodel que actúa como la base de todos los modelos, encargándose éste de cosas como registros en la base de datos, eliminaciones de registros y consultas generales.

Uso de clases Helpers: Seguridad y validaciones:

Las validaciones se preparan principalmente a la hora de procesar datos en con el uso de Javascript, por lo cual esa parte se maneja con los modelos o Javascript, y en el ámbito de los helpers tengo una clase enrutador, que se encarga de enrutar las vistas, además de recoger los métodos de las clases necesarias antes de usarlas en las propias vistas, también cuento con una clase de seguridad con métodos estáticos para asegurar las rutas del enrutador, para iniciar y cerrar sesión y generar mensajes de error en caso de excepción.

VISTAS:

En cuanto a las vistas todas funcionan con buffer, cargando los elementos en buffer y con una plantilla mostrando dicho contenido, además de esto cada vista se utiliza para 1 función principal por cada modelo, por ejemplo hay 1 vista para cada función, estas siendo listar elementos, crear elementos, editar elementos, después hay un grupo de excepciones que son vistas diferentes como ver estadísticas del usuario, vista de índice, el inicio de sesión, revisar una tarea y manejar permisos de usuario.

Partials y/o plantillas:

Como se ha comentado en el apartado anterior se utiliza un sistema de plantillas junto a cargas de buffer para quitar carga del servidor, las plantillas principales son header y main, Header siendo donde se colocan los metadatos necesarios para el funcionamiento de la aplicación (cosas como la API, JQuery y los iconos y fuentes importados), main es principalmente la vista general, constando de una barra de cabecera que actúa como enlace al índice con el logotipo y nombre de la aplicación, el cuerpo diviéndose en 2, el contenido principal (donde se carga el contenido de buffer) y una barra lateral con links que puede usar el usuario para ir a las vistas que necesite usar.

Estructura del proyecto:

El proyecto utiliza una estructura estándar de proyecto web, tengo un directorio para la documentación y una para la aplicación. En la de la aplicación hay 2 carpetas “public” que es la carpeta que tiene todos los “assets” o recursos accesibles por cualquier usuario (siendo estos el índice y las carpetas que contienen los estilos y el procesamiento por parte de cliente/Javascript) y la carpeta “app” que es donde se encuentra todo el procesamiento y programación del proyecto, siendo estas la carpeta “controllers” donde se alojan los controladores que realizan los métodos, la carpeta “models” que contiene los modelos necesitados de cada entidad relacionada con el proyecto, la carpeta “views” donde se encuentran las vistas de todo el proyecto y por último la carpeta “core” que contiene los esenciales del proyecto (cosas como la conexión de la base de datos o el modelo padre que usan todas las clases, la clase de seguridad [también puede tener su propio directorio], y la clase enrutador).

USO API:

La API de mi elección fue Google Charts, debido a su fácil implementación y buen “Feedback”/Retroalimentación visual sobre varios datos. Su uso principalmente es en la parte de usuarios para que el admin U owner puedan ver las estadísticas al respecto de las tareas que se encuentran asignadas al usuario en cuestión.

CONTROL DE PERFILES DE USUARIO Y SESIONES:

Las sesiones se utilizan principalmente para almacenar varios datos del usuario, como por ejemplo sus identificadores, permisos y nombre.

CONTROL DE EXCEPCIONES:

El control de excepciones se ha manejado de 2 maneras principales, limitando el uso de funcionalidades según el permiso del usuario (por ejemplo un usuario promedio no debería poder ver al resto de usuarios más allá de los que necesite asignar alguna tarea, en este caso admins y técnicos) y con el try catch común pero usando la clase seguridad junto a Javascript para recoger el fallo y mostrarlo de manera interactiva en un cartel con una animación al usuario.

EXPLICACIÓN DE FLUJO EN CONTROLADORES:

Los controladores funcionan todos igualmente en su mayoría, a la hora de hacer ediciones o registros se encargan de comprobar que se esté pasando algún tipo de información por POST, después de validarla la formatea y la procesa registrandola en la base de datos, si no se ha pasado nada por POST llamará a la vista necesaria en ese momento para poder rellenar el formulario pertinente y así crear la información a procesar.

Todas constan también de algún tipo de consulta preparada, siendo estas registros, eliminaciones y recogidas de datos. Estas van desde una búsqueda puntual hasta búsquedas dinámicas de cualquier dato usando ajax.

Uso de composer:

Composer se encuentra instalado y utiliza namespaces en TODAS las clases, metodos y cualquier tipo de PHP del proyecto, además de eso también se usa Alias de los namespaces y clases en todas las partes del proyecto.

Uso de Ajax:

El ajax creado es dinámico y es uno de los pilares del proyecto, las peticiones son creadas y formateadas por Javascript, el cual usando un GET se las envía a nuestro PHP, el cual recibe 2 parámetros, la consulta y la tabla, usa un switch con la tabla y llama al controlador necesario para hacer la consulta, en cada modelo hay una consulta preparada para recoger la búsqueda del ajax, tras realizar la consulta recoge los objetos y se los devuelve al Javascript en formato JSON.

DESARROLLO WEB EN ENTORNO CLIENTE

VALIDACIÓN DE DATOS:

La validación de datos se encuentra en un script que revisa las expresiones regulares pasándolas por un case y devolviendo si están correctas o no, de no estar correctas regresará un mensaje de error.

DOM:

El uso principal del dom es en las vistas, se encarga de crear los resultados del ajax, junto a los enlaces pertinentes, este proceso se utiliza principalmente en la gran mayoría de las páginas, esto funciona recogiendo en el ajax la tabla buscada y la búsqueda del ajax, de haber un resultado anterior lo borra, tras recoger la información del ajax y hacer todo el proceso se crea un cuerpo donde van los resultado del ajax, este siendo invisible, mientras uno vacío en blanco se ve delante de él y después utiliza un case con la tabla a buscar para modificar los resultados y crear un div con la información recogida y los enlaces con los identificadores necesarios para ello, y al terminar de crearlo todo le da una clase para hacerlo visible, así evitando parpadeos y fallos visuales al cambiar rápidamente de búsqueda en el campo asignado para las búsquedas y haciendo aparecer los resultados lentamente gracias a una mezcla de modificaciones de estilo con DOM y clases modificadas de CSS ,de no encontrarse resultados crea un aviso diciendo que no se ha encontrado nada y lo coloca en el div de los resultados.

También se utiliza para la generación de errores, recogiénolos de PHP y metiendo el mensaje en un div fuera de pantalla y tras esperar unos segundos usando una función asíncrona cambia la clase del div para que gracias al CSS se vea como aparece el cartel desde la parte superior de la pantalla, baja, se mantiene unos 5 segundos aproximadamente para que el usuario pueda leer el mensaje y luego cambia la clase usando la misma función asíncrona la devuelve al estado original.

El DOM también es muy utilizado a la hora de modificar las clases para que CSS pueda cambiar el aspecto y dar transiciones y animaciones.

Y se usa también en un multiselect con checkboxes utilizados en un par de formularios, crea un elemento con las opciones especificadas desde PHP, después coge dichas opciones y las crea como checkboxes con labels en un div con scroll y búsqueda interna, cada uno con su capacidad de chequeo, a no ser que se haya limitado de antemano con una de las opciones, que limita

cuantos elementos chequeados hay, y luego usa el label para crearlo y ponerlo en el elemento que clickamos originalmente para saber visualmente que elementos hemos seleccionado sin tener que comprobar las checkboxes otra vez.

La API también usa el DOM, pasándole información desde una cabecera modificada generará un div con los elementos

TRY CATCH:

En cuanto al control de errores debido a que el código de los diversos scripts están muy limitados y comprobados pues los errores está limitados y las excepciones controladas. El único caso en el que se puede soltar un error por consola es en el ajax en caso de un fallo grave de conectividad u recogida de datos (cada uno con su mensaje modificado).

EVENTOS:

Los eventos más utilizados han sido window.onload y window.onresize. Onload se ha utilizado para automatizar la ejecución de scripts a partir de que cargue la página, usado principalmente para la función de la API que recoge la información del usuario y genera un grafico con ello y con el ajax, para generar una búsqueda de todos los elementos. Onresize se ha usado en la función de la API para regenerar el grafico cuando se modifique el tamaño de la ventana, esto es debido a que el gráfico que devuelve la API tiene un tamaño fijo y no cambia, así haciendo que la pagina falle a nivel visual al cambiar de tamaño la ventana, es por esto que gracias a esta función se regenera el gráfico en el máximo de la ventana actual para adaptarlo.

PÁGINA MAIN:

La página inicial se procesa principalmente con PHP al igual que los permisos, pero si que se usa JQuery para realizar los menús hamburguesa de la versión móvil.

AJAX:

El ajax va sujeto a un input en todas las páginas que recoge el valor del propio campo y se lo pasa a la función del ajax junto a la tabla en la que se va a buscar, tras recogerlo se envían ambos al ajax PHP por GET usando un FETCH, recogemos el resultado, de no salir nada generamos un mensaje de error, de haber un fallo de conectividad generamos un error por consola, uno en la conexión al archivo, otro si al búsqueda falla, tras recoger los datos del ajax PHP depuramos las claves (debido a que dependiendo del parámetro de recogida de PHP las claves se pueden recoger con su nombre indexado u ordenado como si fuera un array), esto lo realizamos descartando las claves puramente numéricas y quedándonos solo con las que tengan un nombre indexado, tras hacer esto hacemos un foreach de cada resultado recogido para filtrarlo, creamos un elemento (principalmente un div), en el que meteremos los datos recogidos en orden, esto yo lo he realizado recogiendo todos los datos que no sean 0 contenga la palabra ID y juntándolos en un solo hilo de texto, el cual inyectaremos en el div creado, junto a este texto también crearemos con la ID un par de enlaces, principalmente uno para editar y otro para eliminar.

Dependiendo de la vista crearemos u modificaremos otros elementos, por ejemplo los enlaces (href) de los enlaces (a) que hemos creado anteriormente, en el caso de los usuarios también crearemos un botón de ver estadística que nos llevará a la página en la que se encuentra la API.

En el caso de las tareas este proceso se complica ya que son 2 ajax conjuntos pero que realizan consultas parecidas, esto nos lleva a modificar casi todo el ajax, en mi caso recogí la tabla que me paso para saber cual buscar y las pasé con una letra P para tareas pendientes y C para tareas completas, cada una con su campo de búsqueda y de recogida de resultados. En la parte PHP ocurre el mismo filtro pero solo cambia la consulta preparada para filtrar entre tareas completadas y pendientes, en el caso de no ser ninguna de las 2 realiza una consulta de todas las tareas.

JQUERY:

Jquery se ha utilizado unicamente para crear un toggle de las clases de un objeto, siendo este el menú desplegable de la versión móvil, gracias a esto y un par de clases de CSS podemos hacer que el desplegable aparezca y desaparezca a voluntad tras darle a cierto div con el icono del menú hamburguesa.

CONCLUSIÓN

Este proyecto a comparación de otros ha sido muy básico e incluso puede que algo pobre, pero eso no deja de lado el esfuerzo que le he puesto.

Gracias a esta labor he aprendido a programar de nuevo usando objetos y herencias en PHP, he aprendido a usar librerías y namespaces y alias con Composer, me ha ayudado a refinar mis habilidades con JavaScript, he vuelto a aprender como hacer Mockups y revisado mis animaciones de CSS, refrescado mi análisis de Bases de datos, e incluso he vuelto a revisar mis esquemas de previos años de análisis DAFO. Además de esto he intentado hacer un despliegue propio el cual me ha consumido muchas horas y al final he optado por un servidor común, pero el intento no ha sido en vano ya que me ha ayudado a comprender como hacer mejores despliegues en docker.

Este proyecto ha sido arduo para mi y lamento mucho no haber podido echar más horas, pero por varias cuestiones mi tiempo ha sido limitado y además algunos hemos empezado con algunas pequeñas desventajas pero con otros puntos fuertes, me hubiera gustado hacer más énfasis en partes técnicas de PHP, Diseño y Empresa.

A pesar de todo esto he cumplido con todos los requisitos funcionales que he antepuesto y creo que tras un par más de mejoras este proyecto podría incluso tener un uso real más allá de ser un proyecto a nivel educativo.