GESTIÓN DE OBJETIVOS

Bea Couchoud Ruiz

Memoria del Proyecto de DAM

IES Abastos

Curso 2020-2021

Grupo 7U

Valencia, X de junio de 2021

Tutor individual: Eduardo González Sanz

Tabla de contenido

1. Introducción

En el siguiente documento se pretende presentar la aplicación ideada, comentar el origen y la motivación que llevó a hacer realidad este proyecto, los objetivos iniciales y posteriores del mismo, y detallar todas las características técnicas que permitan entender qué hace la aplicación y cómo lo hace.

1.1 Justificación

Casi todo el mundo ha aprovechado el confinamiento vivido durante el 2020 para plantearse nuevos retos, mejores hábitos de vida o retomar objetivos que llevaban tiempo en segundo plano por falta de tiempo. Ahora que cada vez está más cerca la vuelta a la normalidad es posible que gran parte de los logros también vuelvan a su estado inicial. Para evitar perder todo ese esfuerzo realizado se ha decidido desarrollar una aplicación que permita de forma ágil e intuitiva tomar nota de casi cualquier reto u objetivo que queramos iniciar o mantener de la manera mas personalizada posible, para poder llevar un control de las mejoras realizadas y crear mayor adherencia a los nuevos hábitos que se deseen.

Por tanto, este proyecto no busca innovar, ya que hay muchas maneras, físicas y virtuales, de llevar un registro diario. Lo que se desea conseguir con este proyecto es crear una herramienta sencilla y útil que permita llevar un control de objetivos de manera adaptada y personalizable, y sobre todo, que permita ver la evolución de los mismos con un simple vistazo.

1.2 Objetivos

El objetivo principal del proyecto es crear una aplicación multiplataforma, que permita registrar y visualizar los avances realizados desde distintas plataformas (móvil, escritorio y web), a ser posible con sincronización de datos entre las mismas. También sería interesante poder compartir tus retos y logros con amigos y familiares, para motivarse y apoyarse mutuamente.

Como objetivo secundario y personal, se ha buscado realizar esta aplicación híbrida empleando tecnologías y frameworks que no se han visto a lo largo del curso en combinación con otras que si que se han utilizado, aprovechando así la realización de este proyecto para reforzar lo aprendido durante los últimos meses y obtener también nuevas habilidades y conocimientos.

1.3 Idea inicial

En un principio las necesidades mínimas que se buscan cubrir con esta aplicación multiplataforma son:

- Los posibles usuarios deben poder crear su cuenta e iniciar sesión en ella.
- Los usuarios deben poder crear, editar, eliminar y hacer públicos o privados nuevos logros u objetivos, a los que de ahora en adelante nos referiremos, tanto en el presente documento como durante el desarrollo de la aplicación, como *ítems*.

- Dentro de los ítems se debe poder crear y editar registros diarios.
- Los usuarios deben poder ver de manera rápida sus avances de manera gráfica.
- Los usuarios deben poder ver su perfil y modificar sus datos si fuese necesario.
- La aplicación debe poder lanzar notificaciones.

Si hubiese tiempo suficiente otras características deseables serian:

- Poder añadir o eliminar amigos.
- Poder consultar los ítems públicos de los amigos añadidos y que a su vez los amigos del usuario puedan ver los ítems públicos del mismo.

En caso de que no diese tiempo debido a la limitación de horas se dejaría el código desarrollado y la aplicación preparados para añadir esta nueva funcionalidad de la manera más sencilla posible.

2. Diseño

Este apartado está dedicado a la fase de diseño del proyecto. Basándose en las necesidades de la aplicación y la información de la etapa de análisis, se detallarán los módulos y subsistemas que forman la totalidad del software, con el fin de llevar a cabo su desarrollo.
El objetivo de la etapa de diseño, es la definición de la arquitectura de la aplicación y la especificación detallada de sus distintos componentes, así como del entorno tecnológico necesario para funcionamiento.

2.1 Análisis

Aquí se recoge toda la información relativa a la fase de análisis de la aplicación. Quedan definidos tanto las tecnologías y herramientas necesarias para llevar a cabo este proyecto como los requisitos de software necesarios para usar la aplicación y todo ello sirve de base para la siguiente etapa del proyecto.

2.1.1 Tecnologías y Herramientas

Las tecnologías y herramientas que se han elegido para realizar este proyecto son las siguientes:

Aplicación Híbrida

Las aplicaciones híbridas, a diferencia de las nativas, son aquellas capaces de funcionar en distintos sistemas operativos móviles. Entre ellos: Android, iOS y Windows Phone. De esta manera, una misma app puede utilizarse en cualquier smartphone o tablet, indistintamente de su marca o fabricante.

Para ello, estas aplicaciones tienen componentes que permiten la adaptabilidad de un mismo código a los requerimientos de cada sistema.

Entre las ventajas de las aplicaciones híbridas sobre las nativas destacan:

- El desarrollo es más ágil y sencillo, por lo tanto, más económico.
- Sus actualizaciones son más fáciles de desarrollar e implementar y el mantenimiento es más sencillo.
- Un mismo código se puede utilizar en todos los sistemas y plataformas.
- No se requieren permisos externos para distribuir la app en las tiendas *online*.
- Permite abordar a un mercado de usuarios mucho más amplio.

Así mismo, aunque no aprovechan todo el rendimiento que ofrece cada sistema operativo, pueden tener acceso a los componentes nativos a través de plugins como Capacitor y Apache Cordova.

Ionic

Ionic es un SDK de front-end de código abierto basado en tecnologías web (HTML, CSS y JS) cuyo propósito es el de crear aplicaciones híbridas, por lo que ya de base estamos hablando de un proyecto multiplataforma.

Ionic es compatible y permite el desarrollo de aplicaciones para Android, iOS, Windows, WebApps y Amazon FireOS.

Además, al tratarse de un software multiplataforma, los equipos de desarrollo pueden trabajar también desde diferentes sistemas (Windows, MacOS, Linux..).

Al basar toda la maquetación en HTML y, en este caso concreto, Angular, nos permite el diseño de interfaces interactivas de manera sencilla y totalmente MVC, de forma que la lógica se traslada a los modelos y controladores y las vistas se diseñan por separado.

Ionic cuenta con decenas de componentes prediseñados con los que se puede crear de forma rapidísima tarjetas, alertas, botones, menús, formularios…

Todos estos componentes están adaptados a las interfaces nativas de los diferentes sistemas, por lo que si por ejemplo usamos un componente «Alert», este se mostrará de manera diferente en Android que en dispositivos iOS.

Por último, hay que tener en cuenta que Ionic es un proyecto open source, por lo que es posible su uso libre y gratuito, por lo que, como ya hemos visto, además de ahorrar tiempo ahorraremos dinero en costes de licencias, etc.

Angular / TypeScript

Angular es un framework que permite crear aplicaciones web progresivas y aplicaciones multiplataforma. Es de código abierto, mantenido por Google.

Angular se basa en clases tipo "Componentes", contenidos en “Modulos”, cuyas propiedades son las usadas para hacer el enlace o binding de los datos. En dichas clases hay propiedades (variables) y métodos (funciones a llamar). Algunas de las ventajas que ofrece son:

- Mejora de la productividad: el planteamiento de arquitecturas estándar repercute directamente en la productividad del proyecto. Angular proporciona controladores, servicios y directivas para organizar el proyecto.

- Generación de código: Angular convierte tus plantillas en código altamente optimizado para las máquinas virtuales de JavaScript de hoy en día, ofreciéndote todas las ventajas del código escrito a mano con la productividad de un framework.

- División del código: Las aplicaciones de Angular se cargan rápidamente gracias al nuevo enrutador de componentes. Éste ofrece una división automática de códigos para que los usuarios sólo carguen el código necesario para procesar la vista que solicitan.

- Reutilización de código: El diseño de Angular adopta el estándar de los componentes web. Un componente en Angular es una porción de código que es posible reutilizar en otros proyectos de Angular sin apenas esfuerzo, en otras palabras, te permiten crear nuevas etiquetas HTML personalizadas, reutilizables y auto-contenidas, que luego se pueden utilizar en otras páginas y aplicaciones. Esto facilita el desarrollo de aplicaciones de manera mucho más ágil, pasando de un "costoso" MVC a un juego de puzles con los componentes.

- Angular CLI: Las herramientas de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.

Se ha decidido usar Angular porque, aunque la versión mas reciente de Ionic permite usar cualquier framework de front-end, con Angular tiene más recorrido y por tanto la documentación es más sólida y la comunidad de usuarios más grande.

Aunque Angular no te obliga a usar TypeScript, se ha decidido usarlo para la realización del proyecto por varias razones, una de las primeras es que la base de datos será MongoDB, que trabaja en formato JSON y por tanto TypeScript facilita la organización de los datos. Otra es la consistencia en la documentación. Por ejemplo, ES6 (o sea, ECMAScript 2015) ofrece varias formas diferentes de declarar un objeto, lo cual puede confundir a muchos. Con TypeScript esto no pasa, y toda la sintaxis y la manera de hacer las cosas en el código es la misma, lo que añade coherencia a la información y a la forma de leer el código. Además, la búsqueda de errores de runtime en JavaScript puede ser una tarea imposible. TypeScript proporciona detección temprana de errores (en tiempo de compilación), y tipado fuerte de clases, métodos, así como de objetos y APIs JavaScript ya existentes. TypeScript es un superset de ECMAScript 6, es decir, incluye todas las funcionalidades de ES6, y además incorpora una capa por encima con funcionalidades extra.

Esto, entre otras cosas, significa que puedes mezclar código TypeScript con código ES6 estándar sin problema, y el compilador de TypeScript seguirá pasando el código a ES5 (tanto ES6 como TypeScript se transpilan a ES5).

Capacitor

Capacitor es el puente de Ionic hacia lo nativo. Con capacitor podemos acceder desde las tecnologías de desarrollo web y Javascript a los recursos nativos de los dispositivos, permitiendo una comunicación sencilla y la utilización de todas las características necesarias para realizar aplicaciones asombrosas, consiguiendo un elevado rendimiento.

Gracias a Capacitor es muy sencillo acceder al SDK nativo de cada plataforma, con una interfaz de desarrollo unificada y optimizada para su uso con el framework Ionic. Capacitor es potente, sencillo y extensible vía plugins creados por el propio equipo de

Ionic y la enorme comunidad de este framework. Los principales motivos por los que se ha elegido usar Capacitor frente a, por ejemplo, Cordova, son las siguientes:

- Excepcional integración con Frameworks Javascript por medio de Ionic: React, Angular y VueJS.
- Crear APPs PWA o escritorio (gracias a Electron).
- Mantenimiento por parte del equipo de Ionic.
- No existe incompatibilidad con las tiendas oficiales.
- Sin necesidad de utilizar plugins puedes acceder a capacidades como: cámara, geolocalización, notificaciones, acelerómetro, accesibilidad…
- Dispone de sus extensiones propias, y además es compatible con las de Cordova.
- Y tan importante como todo lo anterior: Rápido de instalar y ejecutar sin complicados procesos.

Electron

Electron es un framework para JavaScript que permite el desarrollo de aplicaciones enriquecidas de escritorio mediante el uso de tecnologías web. Esta desarrollado por GitHub (lo que garantiza revisiones constantes), es de código abierto y multiplataforma (funciona bajo Linux, Mac y Windows). Utiliza Node.js del lado del servidor y Chromium como interfaz. El motivo principal para usarlo es que, además de estar detrás de muchos proyectos open source importantes (VSCode, Slack, Discord…), se ha utilizado a lo largo del curso en la asignatura Diseño de Interfaces.

Express / NodeJS

Node (o más correctamente: *Node.js*) es un entorno que trabaja en tiempo de ejecución, de código abierto, multiplataforma, que permite a los desarrolladores crear toda clase de herramientas de lado servidor y aplicaciones en JavaScript. El entorno omite las APIs de JavaScript específicas del explorador web y añade soporte para APIs de sistema operativo más tradicionales que incluyen HTTP y bibliotecas de sistemas de ficheros. Algunas de sus ventajas son:

- El código está escrito en "simple JavaScript", lo que significa que se pierde menos tiempo ocupándose de las "conmutaciones de contexto" entre lenguajes cuando estás escribiendo tanto el código cliente como del servidor.
- JavaScript es un lenguaje de programación relativamente nuevo y se beneficia de los avances en diseño de lenguajes cuando se compara con otros lenguajes de servidor tradicionales (Python, PHP, etc.).
- El gestor de paquetes de *Node* (NPM del inglés: Node Packet Manager) proporciona acceso a cientos o miles de paquetes reutilizables. Tiene además la mejor en su clase resolución de dependencias y puede usarse para automatizar la mayor parte de la cadena de herramientas de compilación.

Express es un framework web, escrito en JavaScript y alojado dentro del entorno de ejecución NodeJS. Es, de hecho, el framework web más popular de Node, y es la librería subyacente para un gran número de otros frameworks web de Node populares. Proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Con miles de métodos de programa de utilidad HTTP y middleware a su disposición, la creación de una API sólida es rápida y sencilla.

A pesar de que *Express* es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más.

En contraposición, esta flexibilidad es una espada de doble filo. Hay paquetes de middleware para abordar casi cualquier problema o requerimiento, pero deducir cuáles son los paquetes adecuados a usar algunas veces puede ser muy complicado. Tampoco hay una "forma correcta" de estructurar una aplicación, y muchos ejemplos que puedes encontrar en Internet no son óptimos, o sólo muestran una pequeña parte de lo que necesitas hacer para desarrollar una aplicación web.

Se han elegido estas tecnologías porque, además de lo arriba descrito, han sido empleadas a lo largo del curso en la asignatura Diseño de Interfaces.

MongoDB

MongoDB es una base de datos distribuida, basada en documentos y de uso general que ha sido diseñada para desarrolladores de aplicaciones modernas y para la era de la nube. Es un sistema de base de datos NoSQL, orientado a documentos y de código abierto.

MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida. Las propias consultas son también JSON, por lo que se programan fácilmente.

Además, ofrece un modelo de datos flexible, es decir, los campos pueden variar de un documento a otro; no es necesario declarar la estructura de los documentos al sistema; los documentos se describen por sí mismos. Si se necesita agregar un nuevo campo a un documento, entonces el campo se puede crear sin afectar a todos los demás documentos del sistema, sin actualizar un catálogo central del sistema y sin desconectar el sistema.

Se ha decidido usar MongoDB porque, por un lado, es idóneo para proporcionar a la aplicación que se va a desarrollar la flexibilidad requerida y por otro, ha sido utilizado durante el curso en la asignatura Acceso a Datos.

Material

Material es un sistema de diseño creado por Google para ayudar a los equipos a crear experiencias digitales de alta calidad para Android, iOS, Flutter y la web.

Los componentes de material son bloques de construcción interactivos para crear una interfaz de usuario e incluyen un sistema de estados integrado para comunicar los estados de enfoque, selección, activación, error, desplazamiento, presionar, arrastrar y deshabilitar. El color, la tipografía y la forma de los componentes del material, como los botones, se pueden modificar fácilmente para que coincidan con el branding escogido.

En general, se puede decir que da un aspecto uniforme y un enfoque jerarquizado a una aplicación de manera sencilla. Se ha escogido porque, además de funcionar muy bien en combinación con Angular, se ha visto en el desarrollo del curso durante las clases de Programación Multimedia y Dispositivos Móviles.

Visual Studio Code

Visual Studio Code es un potente editor de código fuente multiplataforma Windows, Mac, Linux con reconocimiento de sintaxis de código y coloreado de una multitud de lenguajes e integración con Git.

VSC es un editor ligero, no un IDE completo, es decir no es un sustituto del Visual Studio, sino que más bien es una herramienta básica para cubrir las necesidades de edición de código simple. Lo más importante de este editor es su gratuidad, que es multiplataforma y que ofrece muchas posibilidades gracias a múltiples extensiones.

Git

Git es el software de control de versiones más utilizado. Entre otras muchas cosas, Git te permite subir y actualizar el código de tu página web a la nube de GitHub. De esta forma siempre puedes disponer de él cuando lo necesites. Pero, además, puedes:

- Conocer quién y cuándo ha realizado una determinada modificación.
- Realizar comparaciones entre versiones de una aplicación.
- Observar la evolución del proyecto con el paso del tiempo.
- Contar con una copia del código para poder retroceder ante cualquier imprevisto.
- Estar al tanto de los cambios en el código fuente.

- Tener una copia de seguridad del proyecto al completo.
- Disponer de un historial en el que se detallen las modificaciones realizadas en el código del sitio web.
- Git es software libre y open-source.

Trello

Trello es un software de administración de proyectos con interfaz web y con cliente para iOS y Android. Emplea el sistema Kanban para el registro de actividades que, mediante tarjetas virtuales, organiza tareas, permite agregar listas, adjuntar archivos, etiquetar eventos, agregar comentarios y compartir tableros.

Trello es un tablón virtual en el que se pueden colgar ideas, tareas, imágenes o enlaces. Es versátil y fácil de usar pudiendo usarse para cualquier tipo de tarea que requiera organizar información. Es perfecta para la gestión de proyectos ya que se pueden representar distintos estados y compartirlas con diferentes persona que formen el proyecto. Con ella se intenta mejorar las rutinas de trabajo de un equipo generando prioridades, tiempos y avisos entre otras muchas funcionalidades.

Microsoft Office

Microsoft Office es una suite ofimática muy popular, de software privativo. Los programas de Office empleados para la realización del proyecto son:

- Word: Con el procesador de texto se ha realizado la totalidad de la memoria.
- Excel: Con el programa de hoja de calculo se ha realizado el diagrama de Gantt.
- PowerPoint: Se ha empleado para realizar la presentación final.

2.2 Planificación

En esta fase, el objetivo prioritario es identificar cada una de las necesidades que deben ser satisfechas al desarrollar el sistema final. Esto se hará efectivo mediante la obtención de requisitos que deben estar presentes en el sistema de información final, con la intención de dar solución a las necesidades identificadas con anterioridad. Al finalizar esta primera etapa en el desarrollo del producto se consiguen los requisitos, tanto funcionales como no funcionales, y su modelado mediante los casos de uso, desarrollados estos últimos en diagramas. Todo esto facilita la comprensión del sistema haciendo que pueda ser diseñado cumpliendo con todos los objetivos y cubriendo las necesidades de forma satisfactoria.

2.2.1 Requisitos

En esta sección la finalidad es detallar de una manera clara y concisa cada uno de los objetivos que presenta el proyecto, así como todas sus características.

Es de suma importancia para que el desarrollo del proyecto concluya con éxito que antes de empezarlo se tenga una completa y plena compresión de los requisitos del software.

Para organizar de la forma más clara posible todos los requisitos se utilizará esta tabla. El formato del identificador será RF o RNF (requisito funcional o requisito no funcional respectivamente), seguido de un número (un índice de dos dígitos). Estos identificadores podrán ser después usados para organizar tareas con mayor facilidad. Por ejemplo: “RF - 01”.

IDENTIFICADOR
NOMBRE:
PRIORIDAD:
DESCRIPCIÓN:

2.2.1.1 Requisitos funcionales

2.2.1.2 Requisitos no funcionales

2.2.2 Gestión del proyecto

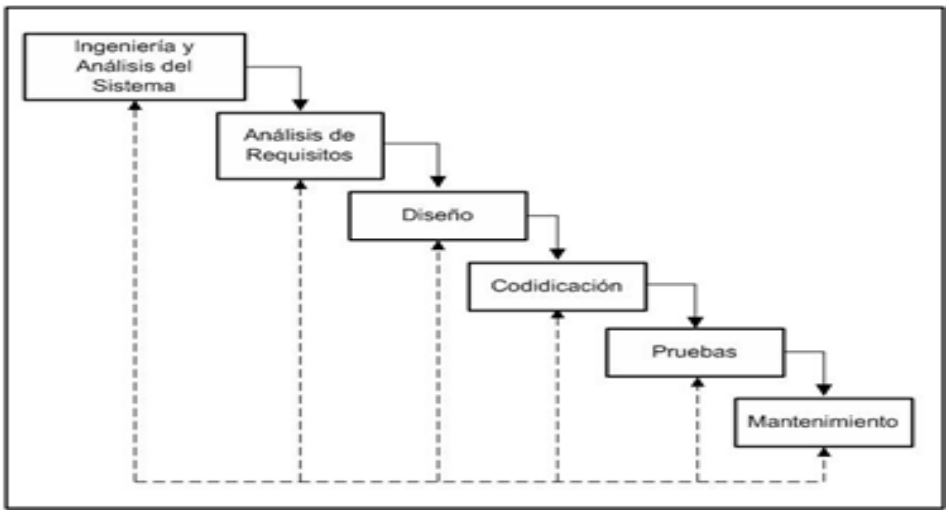
Para gestionar de manera correcta el proyecto se necesita elegir el ciclo de vida más apropiado para el proyecto en función de nuestras necesidades. Otra parte muy importante de la gestión del proyecto es la estimación de tiempo de las diferentes tareas, para lo cual se realiza un diagrama de Gantt.

2.2.2.1 Ciclo de vida del proyecto

El proceso para el desarrollo de software, también denominado ciclo de vida del desarrollo de software, es una estructura aplicada al desarrollo de un producto de software. Hay varios modelos a seguir para el establecimiento de un proceso para el desarrollo de software, cada uno de los cuales describe un enfoque distinto para diferentes actividades que tienen lugar durante el proceso.

El paradigma elegido es el desarrollo en cascada con retroalimentación. Este modelo admite la posibilidad de hacer iteraciones, es decir, durante las modificaciones que se hacen en el mantenimiento se puede ver, por ejemplo, la necesidad de cambiar algo en el diseño, lo cual significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas, es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas.

Después de cada etapa se realiza una revisión para comprobar si se puede pasar a la siguiente.

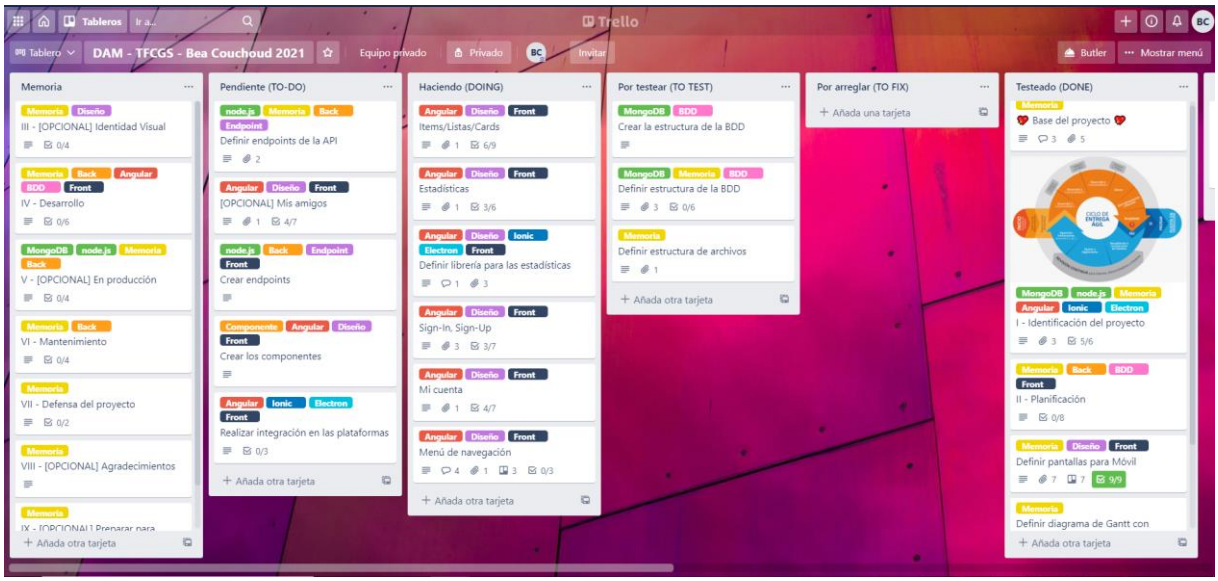


Los motivos principales para su elección han sido que la planificación es muy sencilla, la calidad del producto resultante es alta, permite retroceder en cualquier momento si fuera necesario y es el más sencillo de utilizar cuando el desarrollador tiene poca experiencia.

Una vez elegido el modelo de ciclo de vida del proyecto se ha procedido a planificar cada una de sus fases con el objetivo de estimar el tiempo que va a ser utilizado para la realización completa del proyecto.

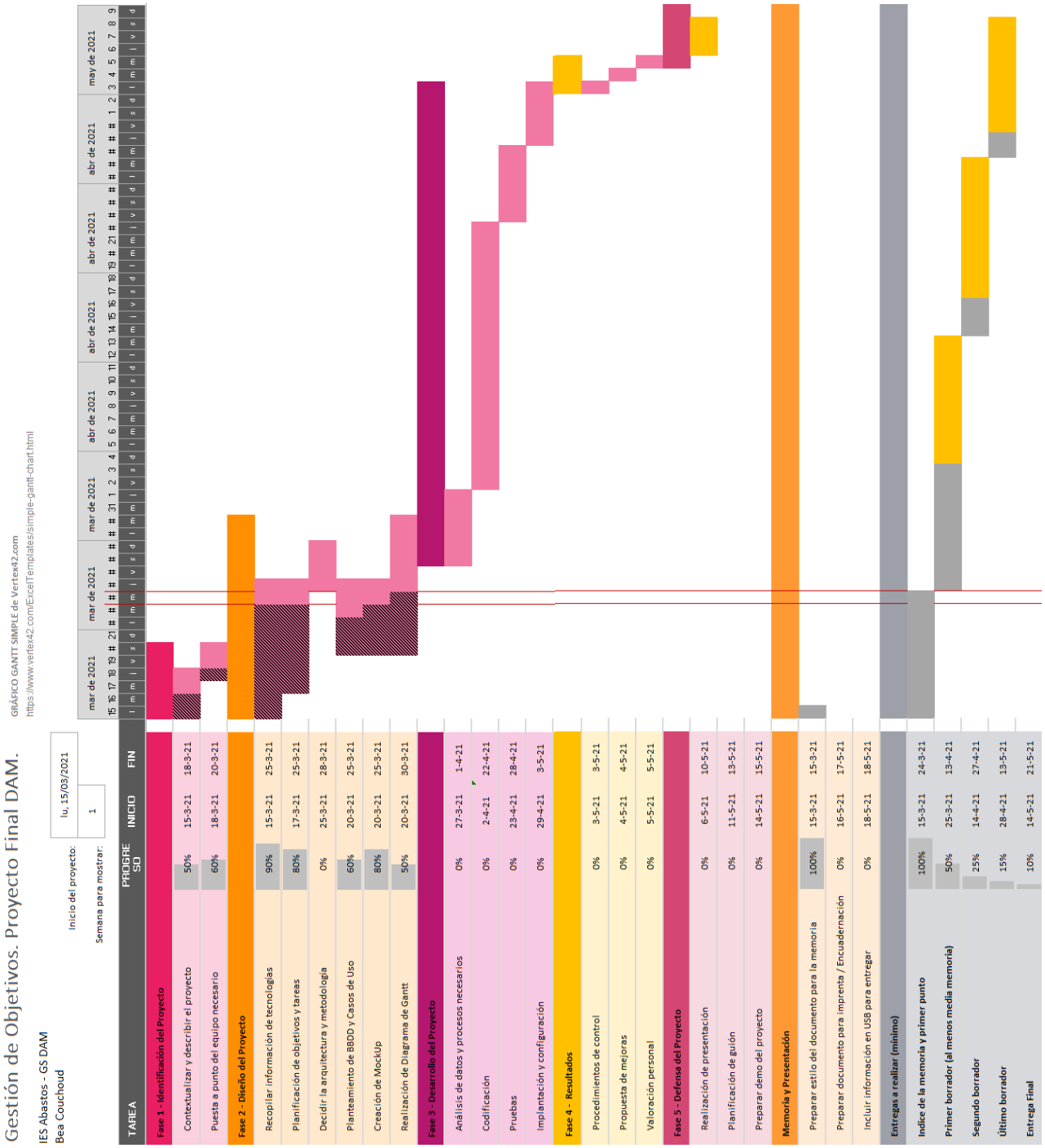
2.2.2.2 Trello

En el tablero de Trello podemos observar las tareas a realizar y el estado actual de cada una de ellas. Permite mejor organización.

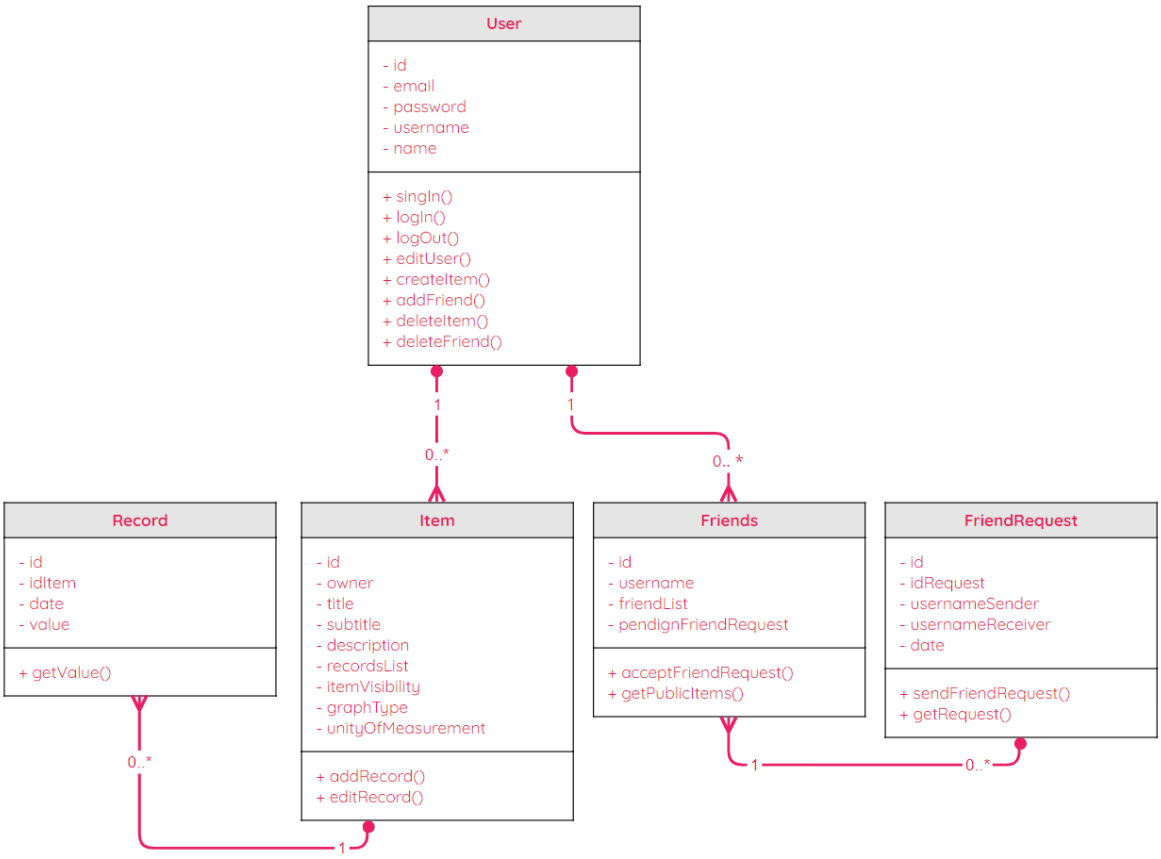


2.2.2.3 Diagrama de Gantt

En el diagrama de Gantt se puede observar la duración de cada uno de los plazos de realización de tareas, separados por las fases anteriormente descritas. Se puede observar que algunas de las tareas se solapan puesto que pueden ser desarrolladas de forma simultánea, siendo independientes unas con otras, mientras que otras necesitan que las anteriores hayan sido finalizadas para poder iniciarse. La estimación realizada plantea una duración de, aproximadamente, dos meses.



2.3 Base de Datos, UML y Casos de Uso



- Documento de MongoDB perteneciente a la colección User:

{

"_id" : ObjectId("5462ee906994db65eeab3075"),

"username": “bcouchoud”,

"name": "Bea Couchoud",

"email": "beacouchoud@gmail.com",

"password": "DAM2021"

}

- Documento de MongoDB perteneciente a la colección Item:

{

"_id" : ObjectId("5462ee906994db65eeab3076"),

"owner": “bcouchoud”,

“title”: “Example Item”,

“subtitle”: “This is a exmple”,

“description”: “This is a description of the item.”

"recordList":

[

“idItem1”,

“idItem2”,

“idItem3”,

“idItem4”

],

"itemVisibility": False,

“graphType”: 2,

"unityOfMeasurement": "vasos de agua"

}

- Documento de MongoDB perteneciente a la colección Record:

{

"_id" : ObjectId("5462ee906994db65eeab3077"),

"idItem": “idItem1”,

"date": "25/03/2021",

"value": 6

}

- Documento de MongoDB perteneciente a la colección Friends:

{

"_id" : ObjectId("5462ee906994db65eeab3078"),

"username": “bcouchoud”,

"friendsList”:

[

“usernameFriend1",

“usernameFriend2”,

```
                “usernameFriend3”,
                “usernameFriend4”
        ],
        “pendingFriendRequest”:
        [
                “idRequest1”,
                “idRequest2”
        ]
}
```

- Documento de MongoDB perteneciente a la colección FriendRequest:

```
{
        "_id" : ObjectId("5462ee906994db65eeab3077"),
        "idRequest": “idRequest1”,
        "usernameSender": "usernameUser1",
        "usernameReceiver": “bcouchoud”,
        “date”: “25/03/2021”
}
```

Queda pendiente desarrollar los casos de uso.

2.4 MockUp, User Interface y User Experience

Vista general de todas las pantallas de la aplicación y como se navega entre ellas. Más adelante se añadirá una breve explicación del detalle de cada pantalla y como interacciona con el usuario. También se incluirá una breve guía de estilos de la aplicación y cual será la organización de las distintas pantallas para la versión de escritorio.