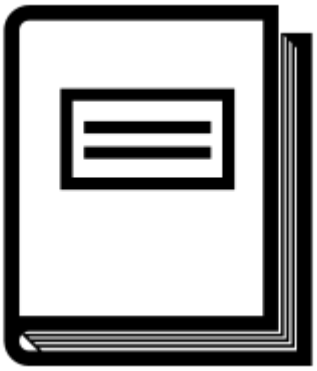
Propuesta, definición y análisis del proyecto

myReader



Trabajo realizado por:

Iván Garzón Martín

ÍNDICE

Propuesta, definición y análisis del proyecto.

1.1 Definición del proyecto

1.1.1 Obtención de la información necesaria para la definición del proyecto 3

1.1.2 Descripción detallada del proyecto 3

1.1.3 Ámbito del proyecto 3

1.1.4 ¿Adaptación o creación? 3

1.1.5 Conceptos básicos del proyecto software 4

Planificación del proyecto web.

1.1.6 Definición de tareas 5

1.1.7 Estimación de tiempos 5

Análisis del proyecto web

1.1.8 Requisitos técnicos Web 6

1.1.9 Elementos del contenido 6

1.1.10 Diseño gráfico 6

1.1.11 Herramientas para Web 6

Ingeniería del proyecto web.

1.1.12 Diseño del software 7

1.1.13 Diseño de datos 8

1.1.14 Diseño del sistema 8

1.1.15 Diseño de la interfaz de usuario 8

1.1.16 Diseño del contenido 8

Identificación y cuantificación de contingencias 9

1.2 Aseguramiento de la calidad 9

Definición del proyecto

- **Obtención de la información necesaria para la definición del proyecto.**

Para obtener toda la información necesaria del proyecto, me he guiado por ciertas páginas existentes como podrían ser FilmAffinity o GoodReads y he intentado crear, a raíz de ello, una plataforma para los amantes de la lectura.

- **Descripción detallada del proyecto**

Para dar una información más detallada del proyecto, digamos que se trata de una plataforma en la que, tanto el usuario registrado como el anónimo, podrá disfrutar de un gran catálogo con libros de los cuales podrán informarse sobre cualquier datos del mismo.

Además, como usuario registrado, éste tendrá la opción de añadir los libros que haya leído a su colección y así llevar un recuento a lo largo de su día a día como lector.

- **Ámbito del proyecto**

En cuanto al ámbito del proyecto, he decidido escoger Eclipse para el desarrollo del back, ya que en mi opinión te ofrece muchas facilidades a la hora de desarrollar código.

Además de esto, Eclipse ofrece un análisis de código más claro y una forma de selección de dependencias sencilla y fácil.

Para la parte de front, he escogido Angular en Visual Studio Code, ya que es un framework ya conocido y estoy bastante familiarizado con él. Y por último, el SGBD que he escogido ha sido MySQL, debido a la previa experiencia con él.

- **¿Adaptación o creación?**

Se podría considerar que myReader ha sido más una adaptación que una creación, ya que se han utilizado ciertas ideas existentes de otros proyectos aplicándolas en un ámbito menos reconocido que el de las películas o series, en este caso el de la lectura.

También se le ha incluido alguna nueva funcionalidad, por lo cual el aplicativo también tiene un punto de creación.

- **Conceptos básicos del proyecto software**

Para definir ciertos conceptos de un proyecto software, lo dividiremos en tres puntos:

- Invisibilidad
  El avance dentro de un proyecto de construcción de una carretera puede ser visto, en cambio, en el software no es inmediatamente visible.

- Complejidad
  Los proyectos software contienen mayor complejidad que otros tipos de proyectos respecto a los recursos necesarios.

- Flexibilidad
  El software puede cambiarse más fácilmente que otros productos.

Además de esos tres puntos, podemos decir que muchas técnicas aplicables a la administración de proyectos son aplicables a la administración de proyectos software.

Planificación del proyecto web

- **Definición de tareas**

1. Moodboard: El moodboard se trata de recopilar imágenes e información relacionada con el proyecto que se va a desarrollar.

2. Wireframe: Hecho con un programa de diseño, se trata de un esbozo para reflejar la distribución de los elementos del proyecto web.

3. MockUp: Al igual que el Wireframe, se trata de un boceto pero ya más complejo y con todo el estilo necesario para representar cómo se vería la web final.

4. Desarrollo y maquetación: Como su propio nombre indica, se trataría de la fase de desarrollo del código fuente para hacer este aplicativo funcional.

5. Pruebas: Una vez finalizado el desarrollo, este es el momento para identificar los posibles fallos y poder solventarlos.

- **Estimación de tiempos**

Para la estimación de tiempos, hablaremos en % del tiempo total empleado en las tareas realizadas en el punto anterior.

1. 5%
2. 10%
3. 10%
4. 55%
5. 20%

Análisis del proyecto web

- **Requisitos técnicos Web**

    - Tipo de servidor: Es el lenguaje con el que queremos programar la página y que el servidor necesita para ser compatible. En este caso Java, aunque el servidor puede contener más de un lenguaje a la vez sin problemas.

    - Base de datos: El sistema en el que se almacena la información de la web. En este caso es MySQL, ya que es gratuito y muy potente. Existen otros como Oracle, SQLserver o MariaDB.

    - Capacidad y potencia del servidor: Debemos calcular el espacio que necesitaremos para almacenar la página tanto en el momento de finalizar el proyecto como durante su posterior duración.

- **Elementos del contenido**

Los principales elementos que encontramos en nuestra aplicación son: texto, imágenes, enlaces, animaciones, formulario, cabecera, pie de página, botones interactivos, etc.

- **Diseño gráfico**

Para esto, tendríamos que encargarnos de dejar todo el contenido gráfico (imágenes, textos, etc) subido en un repositorio externo, en este caso Firebase. Dicho contenido se extrae de Internet, previamente seleccionado y estudiado para evitar problemas con el copyright.

- **Herramientas para Web**

Con ciertas herramientas, como podrían ser páginas online de edición de imágenes o generadores de iconos para usuarios, nos facilita mucho la labor de generar el contenido necesario en nuestro aplicativo. Además de esto, la herramienta de desarrollador del propio navegador nos ayudará también a hacer todo de una forma más sencilla.

Ingeniería del proyecto web

- **Diseño del software**

Para disponer de un diseño óptimo en el backend, he escogido el framework Spring ya que ofrece unas características que me llaman la atención por encima de los demás, en los cuales podemos destacar el soporte integrado de Tomcat y una fácil gestión de las dependencias. Las dependencias elegidas para este caso han sido las siguientes:

- ***Lombok:*** Ya que me ofrece una librería de anotaciones la cual ayudará a reducir la cantidad de código escrito.
- ***Spring Web***: Esencial para construir una web. Incluyendo RESTful, MVC y usando Tomcat como contenedor predeterminado.
- ***Spring Security***: Ofrece una autenticación y control de acceso customizado.
- ***Spring Data JPA*** y ***MySQL Driver***: Para controlar la persistencia de datos en almacenamiento SQL.
- ***Spring Boot:*** Me garantiza reinicios rápidos de la aplicación, LiveReload y una mejor configuración para la experiencia de desarrollo.

Por otro lado, para el manejo de datos desde el controlador hasta la vista, he utilizado Thymeleaf debido a las siguientes ventajas respecto a JSP:

- Podemos prototipar sin necesidad de ejecutar la aplicación.

- Si nuestra aplicación está corriendo y solo modificamos las plantillas, no es necesario volver a desplegarla, cambiando la propiedad cache a false.

- El dialecto estándar, o el dialecto de Spring si trabajamos con Thymeleaf conjuntamente con Spring, es mucho más potente y rico en funcionalidades que la librería de etiquetas de JSTL.

Para el frontend he decidido utilizar lenguaje HTML con su correspondiente estilo en CSS y el añadido del kit de herramientas Bootstrap, debido a la opción del desarrollo web responsive para dar una mejor forma a la web y utilizar el estilo propio que nos ofrece. Por otro lado, se ha utilizado también JavaScript para dar cierto dinamismo a la página web y añadir alguna que otra funcionalidad extra.

- **Diseño de datos**

En cuanto al sistema de gestión para la Base de Datos he escogido MySQL, ya que existe una gran variedad y cantidad de información para poder realizar cualquier tipo de desarrollo o extracción de información, lo cual esto ayuda increíblemente en la mejora de tiempos de creación de cualquier proyecto de software. Además, SQL tiene unos estándares bien definidos y mantiene una sencillez en su escritura.

- **Diseño del sistema**

El objetivo para tener un correcto diseño del sistema ha sido proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de planificación temporal. Estas estimaciones se han hecho dentro de un marco de tiempo limitado al comienzo de un proyecto de software y se han ido actualizando a la vez que progresaba el proyecto. Además las estimaciones han definido los escenarios del mejor caso, y peor caso, de modo que los resultados del proyecto hayan podido limitarse.

- **Diseño de la interfaz de usuario**

En cuanto al diseño de la interfaz del usuario, se han tenido en cuenta varios aspectos muy importantes, como pueden ser los colores usados, el contraste de los mismos, estilo a los elementos enfocados, la disminución del ruido visual y la eliminación de elementos innecesarios.

- **Diseño del contenido**

Para el diseño del contenido, hemos optado por un estilo minimalista, con un contenido sencillo y fácil de ver y entender. Se han utilizado colores claros y con un contraste alto entre ellos. También hemos implementado una intuitiva navegación por el aplicativo, ayudándonos con un menú superior de navegación.

Por último, en cuanto a la administración del contenido, cabe destacar que tanto las imágenes como las hojas de estilo del aplicativo están organizadas por carpetas, lo que hace un fácil manejo por el repositorio de myReader.

Identificación y cuantificación de contingencias

En cuanto a la identificación de contingencias, se han tenido en cuenta varias supuestas problemáticas que podrían haber surgido durante la realización del proyecto.

La más importante, la reestructuración de los modelos y la base de datos debido a la adición de nuevos campos según la necesidad.

Otra contingencia que se ha tenido en cuenta es el despliegue de la aplicación en Proxmox, teniendo preparada otra posibilidad de despliegue en el caso de que algún error o problema hubiese surgido en esa situación. Por lo demás, se han considerado como contingencias de un nivel mínimo supuestos cambios en el diseño del aplicativo, organizando todo para un correcto diseño y funcionamiento del mismo.

Aseguramiento de la calidad

Para el aseguramiento de calidad del software, se han incluido procedimientos para la aplicación eficaz de métodos y herramientas, supervisar las actividades de control de calidad, tales como revisiones técnicas y las pruebas del software, procedimientos para la administración de cambio y elaboración de reportes.

Para llevar a cabo este aseguramiento de calidad, se han recabado y evaluado datos durante el proceso de realización del software. Ayudándonos del ciclo PDCA, nos ha permitido el aprendizaje organizacional y el logro de mejores estándares. Este ciclo se puede entender como:

Plan (Planificar), énfasis en la planificación del proyecto.

Do (Hacer), corresponde a realizar, fabricar o trabajar el producto planificado.

Check (Revisar o Comprobar), para confirmar si el resultado es el esperado.

Act (Actuar), si se presenta algún reclamo, se actúa sobre el problema sin tener que esperar que finalice el proceso, luego se vuelve a la fase de planificación, volviendo al ciclo PDCA, para obtener un mejoramiento.