UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

NutriMenu

Presentado por Álvaro Manjón Vara
en Universidad de Burgos — 7 de enero de 2024
Tutores: Raúl Marticorena Sánchez y Antonio Jesús Canepa Oneto

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. nombre tutor, profesor del departamento de nombre departamento, área de nombre área.

Expone:

Que el alumno D. Álvaro Manjón Vara, con DNI dni, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado título de TFG.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 7 de enero de 2024

Vº. Bº. del Tutor:	Vº. Bº. del co-tutor:
D. nombre tutor	D. nombre co-tutor

Resumen

En este primer apartado se hace una **breve** presentación del tema que se aborda en el proyecto.

Descriptores

Palabras separadas por comas que identifiquen el contenido del proyecto Ej: servidor web, buscador de vuelos, android ...

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Descripción del contenido del trabajo y del estrucutra de la memoria y del resto de materiales entregados.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

Conceptos teóricos

En aquellos proyectos que necesiten para su comprensión y desarrollo de unos conceptos teóricos de una determinada materia o de un determinado dominio de conocimiento, debe existir un apartado que sintetice dichos conceptos.

Algunos conceptos teóricos de LaTeX[1].

3.1. Secciones

Las secciones se incluyen con el comando section.

Subsecciones

Además de secciones tenemos subsecciones.

Subsubsecciones

Y subsecciones.

3.2. Referencias

Las referencias se incluyen en el texto usando cite [**?**]. Para citar webs, artículos o libros [**?**].

[1] Créditos a los proyectos de Álvaro López Cantero: Configurador de Presupuestos y Roberto Izquierdo Amo: PLQuiz

3.3. Imágenes

Se pueden incluir imágenes con los comandos standard de LaTeX, pero esta plantilla dispone de comandos propios como por ejemplo el siguiente:

Figura 3.1: Autómata para una expresión vacía

3.4. Listas de items

Existen tres posibilidades:

- primer item.
- segundo item.

1. primer item.
2. segundo item.

Primer item más información sobre el primer item.

Segundo item más información sobre el segundo item.

-

Herramientas	App AngularJS	API REST	BD	Memoria
HTML5	X			
CSS3	X			
BOOTSTRAP	X			
JavaScript	X			
AngularJS	X			
Bower	X			
PHP		X		
Karma + Jasmine	X			
Slim framework		X		
Idiorm		X		
Composer		X		
JSON	X	X		
PhpStorm	X	X		
MySQL			X	
PhpMyAdmin			X	
Git + BitBucket	X	X	X	X
MikTEX				X
TEXMaker				X
Astah				X
Balsamiq Mockups	X			
VersionOne	X	X	X	X

Tabla 3.1: Herramientas y tecnologías utilizadas en cada parte del proyecto

3.5. Tablas

Igualmente se pueden usar los comandos específicos de LATEXo bien usar alguno de los comandos de la plantilla.

Técnicas y herramientas

4.1. Patrón de diseño Modelo-Vista-Controlador (MVC)

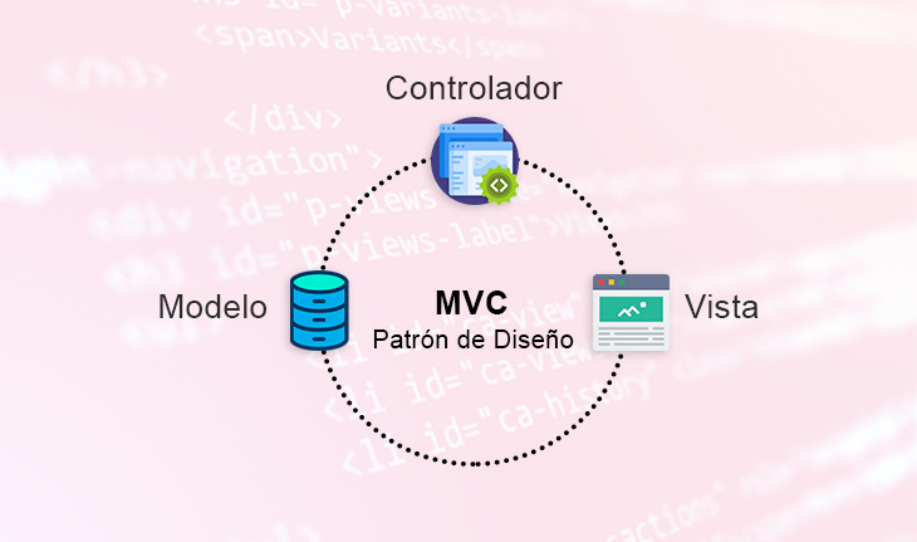

Figura 4.1: Diagrama del patrón Modelo-Vista-Controlador

La función del patrón de diseño conocido como **Modelo-Vista-Controlador** es la de organizar de forma estructurada los componentes fundamentales de un determinado *sistema de software*, estableciendo relaciones entre ellos.

Como metodología de trabajo, fue propuesta por Trygve Reenskaug en 1979, basándose su utilidad en tres ideas fundamentales [1]:

- La organización y clasificación de la información que se aporta al sistema.
- La unificación y gestión de la lógica del sistema.
- La presentación de datos al usuario mediante una interfaz comprensible y asequible.

Los tres componentes fundamentales de este patrón tienen misiones diferenciadas sobre la forma en cómo tratan la información [7]:

- **Modelo**: Es el componente cuya misión es únicamente la de gestionar el contenido de una base de datos, incluida su manipulación y utilización. Es la parte que se va a encargar de la **gestión de los datos**.
- **Vista**: Es el componente que sirve para acceder al contenido de la base de datos y presentar los datos al usuario. Va a ser **la parte con la que va a interactuar el usuario**.
- **Controlador**: Este componente es el que pone en conexión el modelo y la vista, gestionando y procesando las instrucciones que se reciben para obtener un resultado. Su forma básica de trabajar consiste en acceder a la base de datos, manipularlos con una determinada finalidad y presentar los resultados de esa manipulación de los datos. Se va a encargar de gestionar **la lógica y la gestión de los datos**.

El motivo por el cual he decidido utilizar este patrón es porque permite trabajar de una forma precisa y eficaz, haciendo que la estructura del sistema sea clara, ordenada y separada, lo que facilita su comprensión y posterior mantenimiento.

4.2. Contenerización

Se conoce como contenerización a la tecnología de virtualización que permite ejecutar **contenedores**, entornos completamente aislados del resto de la máquina que los está ejecutando, y que contienen todo lo necesario (bibliotecas, código, archivos de configuración, dependencias...) para que se pueda ejecutar una aplicación en cualquier entorno.[6]

A diferencia de entornos clásicos de virtualización como las **máquinas virtuales**, en los que se emula el hardware por completo y se debe disponer

de una instalación completa del sistema operativo para funcionar, en los contenedores se permite que múltiples instancias compartan un mismo sistema operativo, a pesar de estar usando espacios de ejecución distintos. Esto se consigue usando características de Linux como los *espacios de nombres del kernel* o los *cgroups* [15], puesto que los contenedores están basados en Linux, pero esto no impide su ejecución en distintas plataformas y sistemas operativos, incluido Windows.

Docker

Docker es una plataforma de código abierto que permite el desarrollo, ejecución y distribución de aplicaciones en contenedores.

Arquitectura de Docker

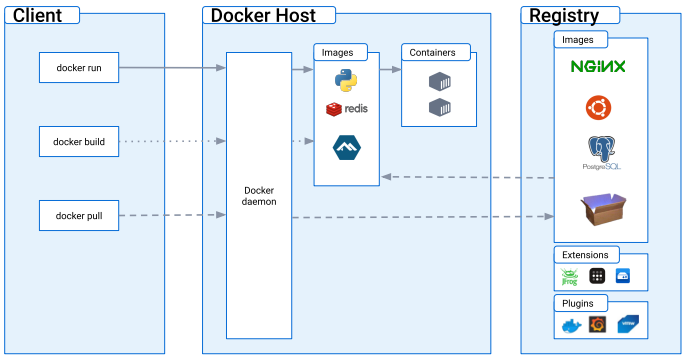

Figura 4.2: Arquitectura de Docker

La arquitectura de Docker es una arquitectura **cliente-servidor** [9]. El cliente de Docker, que es con lo que interactúa el usuario, habla con el *daemon* haciendo llamadas API REST, y este se encarga de gestionar las peticiones y los objetos de Docker, como son las imágenes, los contenedores, los volúmenes y las redes. Finalmente, para conseguir las imágenes, el *daemon* habla con *Docker Registry*, el repositorio que contiene todas las imágenes necesarias para el despliegue. Por defecto se usa **Docker Hub**, un repositorio público en el que cualquiera puede subir y descargar imágenes, pero se puede usar también un repositorio privado.

Imágenes

Las imágenes son plantillas que contienen instrucciones y parámetros para crear contenedores de Docker. Se puede crear un contenedor usando directamente una imagen del *registry*, reutilizando imágenes ya existentes como base para crear nuestras propias imágenes, o incluso creando nuestras propias imágenes desde cero. Las imágenes se crean usando archivos *Dockerfile*, añadiendo en él las distintas dependencias, instrucciones y parámetros, y cada uno de estos elementos es una capa dentro de la imagen. De esta manera, cada vez que se modifica el *Dockerfile* y se vuelve a construir la imagen, realmente sólo se vuelven a construir las capas en las que han habido cambios, acelerando de esta forma el proceso de despliegue. [10]

Docker Compose

Docker Compose es una herramienta que nos permite definir, orquestrar, desplegar y escalar **aplicaciones Docker multi-contenedor**. [8] Para ello, se define un archivo *YAML* en el que se van a definir las configuraciones necesarias para los distintos servicios pertenecientes a la aplicación. Esto nos va a permitir establecer relaciones y dependencias entre distintos contenedores que forman parte de un mismo aplicativo, permitiendo así que varios contenedores formen parte de una misma red o tengan la capacidad de compartir volúmenes de almacenamiento, por poner unos ejemplos.

He decidido usar esta herramienta ya que facilita bastante la conexión e integración entre los distintos servicios que componen la aplicación, como son la base de datos, el backend, y el frontend. Además de esto, a la hora de poner la aplicación en producción, obtenemos varias ventajas respecto a un entorno tradicional:

- **Automatización de despliegues**, ya que una vez hayamos definido en el archivo de configuración de Docker Compose todas las dependencias, configuraciones y relaciones necesarias, con un sólo comando Docker Compose se va a encargar de crear y desplegar los contenedores, configurar los elementos necesarios, e iniciar los servicios.

- **Creación de entornos aislados y consistentes**, lo que nos permite desplegar entornos de desarrollo locales, así como entornos CI/CD de validación y pruebas, y que estos repliquen la infraestructura de producción, que también puede ser desplegada usando Docker Compose.

- **Escalabilidad de infraestructura**, puesto que con un sólo comando podemos aumentar o disminuir el número de instancias de las que dispone un servicio.

4.3. Frameworks

Spring Boot

React

JPA

Bootstrap

4.4. Gestores de paquetes

Maven

Node

4.5. API

Nutritionix

4.6. Lenguajes de programación

En este apartado voy a introducir y hablar un poco sobre los lenguajes de programación que he usado para el desarrollo de este proyecto:

HTML

El lenguaje HTML (*HyperText Markup Language*) hoy en día es la base de internet, ya que es prácticamente imposible encontrar una página web que no tenga definido su contenido usando este lenguaje. Como **lenguaje de marcado** que es, utiliza marcas (etiquetas) que indican qué tipo de contenido se está representando en el navegador, lo que permite identificar y estructurar mejor este contenido, así como ser más precisos posteriormente a la hora de dar estilo a este contenido [12].

Otro de los valores fundamentales del lenguaje HTML es el **hipertexto**, que es lo que permite enlazar fácilmente contenidos, ya sea dentro de la

misma página web o con otras distintas. Esto permite mejorar la forma de acceso a la información e interconectar el conocimiento, algo muy importante puesto que este es uno de los pilares básicos de internet.

Un archivo HTML sólo contiene (o debería contener, según las buenas prácticas [19]) **información referente a la estructura y el contenido de la página web**, pero no información relacionada con el cómo se va a ver estructurada esta información o cómo se van a procesar los datos y se van a interactuar con ellos, puesto que eso es tarea de otros lenguajes, como CSS y JavaScript. Esto hace que, si visualizamos un archivo HTML desde el navegador que no contiene referencias a ningún otro archivo de otro lenguaje, veamos la información mostrada de forma cruda, sin ningún tipo de estilo ni distribución aplicadas mas allá que los que vienen por defecto en los navegadores.

CSS

Como complemento óptimo del lenguaje HTML, el lenguaje CSS (*Cascading Style Sheets*) define **la estructura y los estilos utilizados en la presentación** de documentos escritos con HTML.

CSS no es un lenguaje de programación propiamente dicho ni tampoco un lenguaje de marcado, sino que es lo que se ha dado en llamar un **lenguaje de hojas de estilo**. Su funcionalidad principal es la de definir y mejorar la presentación de páginas web creadas con HTML, creando estilos y formatos que les son de aplicación [11].

De la misma forma que HTML, CSS es uno de los lenguajes más usados hoy día en el diseño de los estilos de páginas web, ya sea de forma pura o mediante el uso de distintos frameworks que facilitan su uso.

JavaScript

La funcionalidad principal del lenguaje JavaScript es la de mejorar la experiencia de los usuarios de páginas web, haciendo que las páginas dejen de ser estáticas para pasar a ser **interactivas y dinámicas**, lo que indudablemente las hace más atractivas e incrementa exponencialmente su funcionalidad [5].

A diferencia de HTML, que es un lenguaje de marcado; y CSS, que es un lenguaje de hojas de estilo; JavaScript es un lenguaje de programación propiamente dicho con **compilación just-in-time**, es decir, compilación en tiempo de ejecución [13]. Este lenguaje consta de todo lo que podríamos

esperar a la hora de escribir código, desde funciones y estructuras de lógica hasta la implementación de programación orientada a objetos.

Al ser un lenguaje de programación con funcionalidad completa, no sólo se usa JavaScript para añadir funciones a las páginas web, sino que hoy en día hay aplicaciones y entornos construidos en JavaScript, como son Node.js, Visual Studio Code, o Adobe Acrobat. JavaScript también puede ser usado tanto en el **lado del cliente**, para añadir interactividad a la web; o en el **lado del servidor**, para manejar la lógica de la aplicación y modificar los datos de esta.

Java

Java es un **lenguaje de programación orientado a objetos basado en C y C++**, buscando ser similar a estos lenguajes pero haciéndolo más universal, ya que la idea de este lenguaje es que **se pueda compilar una vez y funcionar en cualquier sistema**, gracias a la implementación de la máquina virtual de Java.

Este lenguaje es uno de los más populares y utilizados hoy en día, ya que podemos encontrarlo en prácticamente cualquier campo y ámbito, gracias a su compatibilidad, robustez, y gran cantidad de librerías y *frameworks* de terceros. [4]

En mi caso he usado este lenguaje para el desarrollo del *backend*, puesto que al ser un lenguaje que ya conozco y con el que tengo experiencia, sabía que el usar un *framework* como **Spring Boot**, así como distintas librerías como **JPA**, no iba a suponer un problema, puesto que el ecosistema Java es de los más extensos y robustos que hay ahora mismo, por lo que no voy a tener problema a la hora de buscar documentación o investigar cómo añadir una funcionalidad que me interese.

4.7. Herramientas de desarrollo

A continuación voy a proceder a describir las herramientas que me han ayudado a la hora de desarrollar este proyecto:

Visual Studio Code

Visual Studio Code es un **editor de código fuente gratuito y de código abierto** desarrollado por Microsoft y disponible para macOS, Windows y Linux, así como recientemente también en versión web.

Al ser una herramienta bastante enfocada en el desarrollo web (aunque está adaptada para el desarrollo de cualquier lenguaje), puesto que la propia aplicación está desarrollada en JavaScript, y además dispone de gran soporte en cuanto a extensiones y personalización por parte de la comunidad de desarrolladores, es una herramienta perfecta para un desarrollo como el de este proyecto, ya que dispone de un montón de herramientas que pueden agilizar y facilitar el trabajo. En mi caso esto ha sido especialmente útil a la hora de desarrollar el **frontend**, puesto que gracias al alto número de extensiones que existen en el *marketplace* de Visual Studio Code he encontrado bastantes integraciones con React, como Simple React Snippets, una extensión que permite usar abreviaciones para insertar bloques de código comunes en el desarrollo de React.

Como ya he mencionado, este IDE dispone de un montón de opciones a la hora de personalizar el entorno de programación, lo cuál sumado a las propias características de la aplicación (integración con *Git*, autocompletado y resaltado de sintaxis con *IntelliSense*, herramientas de debugging...) la convierten en una aplicación muy versátil [18].

IntelliJ IDEA

Otro IDE que he usado para poder llevar a cabo este trabajo es IntelliJ IDEA, **un editor que permite escribir código Java de forma eficaz y rápida**.

Gracias a ser un editor de código principalmente enfocado en el lenguaje Java, presenta **características e integraciones que de normal no encontraríamos en otros editores** (aunque en editores como Visual Studio Code las podríamos llegar a asemejar mediante el uso de extensiones, pero en este editor tenemos la ventaja de que ya encontramos todo esto por defecto), como integración con los paquetes Maven, detección automática de errores, autocompletado inteligente del código... [16]

Esta aplicación dispone de una versión gratuita y otra de pago, pero al ser estudiante he tenido acceso a la versión completa sin coste adicional, disponiendo de todas las funcionalidades que ofrece esta herramienta. [17]

4.8. Sistema gestor de bases de datos

En esta sección voy a describir los sistemas que he usado para todo lo que conlleva la gestión y almacenamiento de los datos de la aplicación:

MySQL

MySQL es un sistema RDBMS (*Relational Database Management System*) gratuito (aunque dispone de una licencia empresarial) y de código abierto, y es **uno de los sistemas de administración y gestión de bases de datos más populares y utilizados**, motivo por el que lo he escogido como herramienta de trabajo.

Su funcionamiento se basa en las **bases de datos relacionales**, en las que los datos se organizan y almacenan en tablas que pueden relacionarse entre sí; y en el **modelo cliente-servidor**, en el que el servidor es el encargado de gestionar las bases de datos, y el cliente interactúa con el servidor para solicitar información, modificarla, insertarla o eliminarla mediante el uso del lenguaje SQL, permitiendo así que múltiples clientes puedan interactuar con una base de datos centralizada. [20]

Algunas de las ventajas que me han hecho decantarme por este sistema son:

- **Es un proyecto de código abierto, activo y con gran soporte de la comunidad**, lo que significa que va a disponer de prácticamente cualquier característica que necesite, y cualquier error va a ser solucionado lo más pronto posible.
- Este sistema es **conocido por su rendimiento y eficiencia**, por lo que va a poder soportar las cargas de mi aplicación sin problema alguno.
- **Es un sistema altamente escalable**, por lo que si en algún momento la aplicación creciese y necesitase de más potencia de procesamiento o de opciones de clusterización, sé que con este gestor no tendría ningún problema.

4.9. Herramientas de gestión

Metodología Scrum

Para el desarrollo de este proyecto he seguido la metodología Scrum, que es una forma de trabajo que permite **gran agilidad, flexibilidad y eficiencia en la gestión de proyectos de software** [3].

Un proceso Scrum **se ejecuta en base a la realización de sesiones de trabajo (*sprints*)** que suelen tener dos semanas de duración. Al finalizar

cada sprint, se realiza la entrega específica que se ha definido y marcado inicialmente.

Previamente al inicio del sprint, se realiza **una reunión de planificación** dirigida por el responsable del proyecto (*Scrum Master*), que en este caso serían los tutores del TFG, en la que se marca al *Equipo de Desarrollo*, es decir, a mí, **los objetivos a cumplir durante el sprint**, la metodología de trabajo a utilizar, o se analizan y solventan las dudas y obstáculos que puedan presentarse durante el mismo.

Durante el periodo que dura el sprint, **se pueden realizar breves reuniones diarias**. En ellas se analizan los obstáculos surgidos en el día anterior y se sincronizan las actividades a realizar en ese día.

Al finalizar el sprint, **se analiza si el trabajo realizado es óptimo y se ajusta a los objetivos** planteados inicialmente. Para ello se realiza una reunión retrospectiva en la que se analiza qué ha funcionado o no, y qué es mejorable.

La utilidad de la metodología Scrum radica en varios aspectos destacables:

- **Es flexible** y permite adaptarse a los cambios.
- Es transparente y **fomenta la comunicación entre los miembros del equipo**, lo que además redunda en una mayor colaboración entre ellos.
- Al final de cada sprint, **el producto final resulta mejorado e incrementado gracias a un proceso de retroalimentación**, lo que permite su mejora continua.

Git

Git es una herramienta inicialmente diseñada por *Linus Torvalds*, la cual permite el **control de versiones** de un proyecto de software. Este sistema de control gestiona los cambios y los diferentes estados y versiones por los que pasa el código de un proyecto, por lo que es una herramienta esencial para el desarrollo de cualquier proyecto, ya sea individual o colaborativo. [2]

La base de Git son **los repositorios**, el lugar donde *se va a almacenar todo el código y el histórico de cambios* que se ha producido sobre este.

Cada desarrollador va a disponer de su propio **repositorio local**, que no es más que *su copia local del código y sus cambios*, sumado a los cambios que este mismo desarrollador haya realizado.

Además de esto, va a haber un **repositorio central**, que va a ser *el lugar donde se unifican todos los cambios* que realicen los distintos desarrolladores. Los repositorios locales van a ser un *clon* de este repositorio principal. Como veremos en el siguiente punto, en el caso de este proyecto el repositorio central se trata de GitHub.

Los desarrolladores van a realizar los cambios e implementaciones pertinentes sobre el código, y una vez estén satisfechos con estos, van a realizar lo que se denomina como **commit**, *una instantánea del código* en ese preciso momento.

Para poder realizar modificaciones en el código sin pisar el desarrollo de otros compañeros, ni modificar el código principal de la aplicación, se da uso de lo que se denomina como **ramas o *branches***, que son *versiones independientes del código en un estado concreto* (para seleccionar este momento indicamos de qué *commit* partimos). Las ramas son un elemento fundamental a la hora de realizar implementaciones, experimentar con el código o simplemente hacer cambios, y son uno de los puntos clave en las buenas prácticas del desarrollo de software.

Una vez nuestros cambios estén listos, todo funcione correctamente y tengamos nuestros *commits* hechos, podemos subir el código de nuestro repositorio local al repositorio central haciendo ***push***. De la misma forma, si nuestros compañeros han estado realizando cambios y queremos descargarlos a nuestro repositorio local, podemos hacerlo mediante ***pull***.

En caso de que hayamos terminado con el trabajo de una rama concreta y queramos unificar estas modificaciones a la rama principal, usaremos la operación ***merge***, que se encargará de fusionar e integrar los cambios a la rama principal. [14]

GitHub

Aspectos relevantes del desarrollo del proyecto

Este apartado pretende recoger los aspectos más interesantes del desarrollo del proyecto, comentados por los autores del mismo. Debe incluir desde la exposición del ciclo de vida utilizado, hasta los detalles de mayor relevancia de las fases de análisis, diseño e implementación. Se busca que no sea una mera operación de copiar y pegar diagramas y extractos del código fuente, sino que realmente se justifiquen los caminos de solución que se han tomado, especialmente aquellos que no sean triviales. Puede ser el lugar más adecuado para documentar los aspectos más interesantes del diseño y de la implementación, con un mayor hincapié en aspectos tales como el tipo de arquitectura elegido, los índices de las tablas de la base de datos, normalización y desnormalización, distribución en ficheros3, reglas de negocio dentro de las bases de datos (EDVHV GH GDWRV DFWLYDV), aspectos de desarrollo relacionados con el WWW... Este apartado, debe convertirse en el resumen de la experiencia práctica del proyecto, y por sí mismo justifica que la memoria se convierta en un documento útil, fuente de referencia para los autores, los tutores y futuros alumnos.

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] José María Aguilar. ¿Qué es el patrón MVC en programación y por qué es útil? https://www.campusmvp.es/recursos/post/que-es-el-patron-mvc-en-programacion-y-por-que-es-util.aspx, Oct 2019. [Internet; descargado 20-septiembre-2023].

[2] Atlassian. Qué es Git. https://www.atlassian.com/es/git/tutorials/what-is-git, 2023. [Internet; descargado 20-septiembre-2023].

[3] Atlassian. Qué es scrum y cómo empezar. https://www.atlassian.com/es/agile/scrum, 2023. [Internet; descargado 20-septiembre-2023].

[4] AWS. ¿Qué es Java? https://aws.amazon.com/es/what-is/java/, 2023. [Internet; descargado 20-septiembre-2023].

[5] AWS. ¿Qué es JavaScript? https://aws.amazon.com/es/what-is/javascript/, 2023. [Internet; descargado 20-septiembre-2023].

[6] Microsoft Azure. ¿Qué es un contenedor? https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-a-container/, 2023. [Internet; descargado 11-junio-2023].

[7] Easy App Code. Patrón de diseño MVC. ¿Qué es y cómo puedo utilizarlo? https://www.easyappcode.com/patron-de-diseno-mvc-que-es-y-como-puedo-utilizarlo, Sept 2020. [Internet; descargado 20-septiembre-2023].

[8] Docker Docs. Docker Compose overview. https://docs.docker.com/compose/, 2023. [Internet; descargado 12-junio-2023].

[9] Docker Docs. Docker overview - Docker architecture. https://docs.docker.com/get-started/overview/#docker-architecture, 2023. [Internet; descargado 12-junio-2023].

[10] Docker Docs. Docker overview - Docker objects. https://docs.docker.com/get-started/overview/#docker-objects, 2023. [Internet; descargado 12-junio-2023].

[11] MDN Web Docs. CSS. https://developer.mozilla.org/es/docs/Web/css, 2023. [Internet; descargado 20-septiembre-2023].

[12] MDN Web Docs. HTML: Lenguaje de etiquetas de hipertexto. https://developer.mozilla.org/es/docs/Web/HTML, 2023. [Internet; descargado 20-septiembre-2023].

[13] MDN Web Docs. JavaScript. https://developer.mozilla.org/es/docs/Web/JavaScript, 2023. [Internet; descargado 20-septiembre-2023].

[14] Git. Git Basics. https://git-scm.com/book/en/v2, 2023. [Internet; descargado 20-septiembre-2023].

[15] Sascha Grunert. Demystifying Containers - Part I: Kernel Space. https://medium.com/@saschagrunert/demystifying-containers-part-i-kernel-space-2c53d6979504, Mar 2019. [Internet; descargado 11-junio-2023].

[16] JetBrains. IntelliJ IDEA. https://www.jetbrains.com/es-es/idea/, 2023. [Internet; descargado 20-septiembre-2023].

[17] JetBrains. Licencias educativas gratuitas. https://www.jetbrains.com/es-es/community/education/#students, 2023. [Internet; descargado 20-septiembre-2023].

[18] Microsoft. Visual Studio Code. https://code.visualstudio.com/, 2023. [Internet; descargado 20-septiembre-2023].

[19] Midudev. Curso COMPLETO de HTML GRATIS desde cero: SEO, semántica y más. https://www.youtube.com/watch?v=3nYLTiY5skU, Oct 2023. [Internet; descargado 20-septiembre-2023].

[20] Wikipedia. MySQL. https://en.wikipedia.org/wiki/MySQL, 2023. [Internet; descargado 20-septiembre-2023].