Arquitectura de software en capaz

La idea de la arquitectura en capas ha estado presente en el diseño de sistemas de software desde hace varias décadas. Se cree que este concepto surgió por primera vez en la década de 1960 en el contexto de la arquitectura de sistemas de cómputo.

Como modelo de diseño de software se popularizó en la década de 1980 con la aparición de las aplicaciones cliente-servidor, ya que, en estas, la capa de presentación (cliente) y la capa de procesamiento (servidor) se separan en dos componentes distintos, lo que permite una mayor flexibilidad y escalabilidad.

La arquitectura en capas es un modelo de diseño de software, cuya base es la separación de las diferentes funcionalidades del sistema en capas o niveles, donde cada capa se encarga de un conjunto de tareas específicas y se comunica con los niveles adyacentes mediante interfaces bien definidas.

Cada capa se construye sobre la inferior inmediata, la cual proporciona los servicios y la funcionalidad necesarios para que la capa superior pueda funcionar correctamente. Este modelo permite una mejor organización y modularidad del sistema; además facilita su mantenimiento y evolución a largo plazo.

**Arquitectura por capas en una web: pasos básicos**

1. Identifica las responsabilidades de cada capa

En una web se pueden identificar tres capas principales: la capa de presentación, la capa de lógica de negocio y la capa de acceso a datos. La primera se encarga de mostrar la información al usuario; la segunda, de procesa la información y la tercera, de almacenar y recuperar la información.

|  |
| --- |
| // CAPA DE PRESENTACIÓN - HTML Y JAVASCRIPT  <!DOCTYPE HTML>  <HTML>  <HEAD>  <TITLE>MY PAGE</TITLE>  <SCRIPT SRC="MYSCRIPT.JS"></SCRIPT>  </HEAD>  <BODY>  <H1>WELCOME TO MY PAGE</H1>  <FORM>  <LABEL FOR="NAME">NAME:</LABEL>  <INPUT TYPE="TEXT" ID="NAME" NAME="NAME"><BR><BR>  <LABEL FOR="EMAIL">EMAIL:</LABEL>  <INPUT TYPE="EMAIL" ID="EMAIL" NAME="EMAIL"><BR><BR> <BUTTON ONCLICK="SUBMITFORM()">SUBMIT</BUTTON>  </FORM>  </BODY>  </HTML> |

|  |
| --- |
| // CAPA DE LÓGICA DE NEGOCIO - JAVA  PUBLIC CLASS USER {  PRIVATE STRING NAME;  PRIVATE STRING EMAIL;  PUBLIC USER(STRING NAME, STRING EMAIL) {  THIS.NAME = NAME;  THIS.EMAIL = EMAIL;  }  PUBLIC STRING GETNAME() {  RETURN NAME;  }  PUBLIC STRING GETEMAIL() {  RETURN EMAIL;  }  }  PUBLIC CLASS USERSERVICE {  PUBLIC USER CREATEUSER(STRING NAME, STRING EMAIL) {  // VALIDACIÓN Y PROCESAMIENTO DE DATOS AQUÍ  RETURN NEW USER(NAME, EMAIL);  }  }  // CAPA DE ACCESO A DATOS - JAVA  PUBLIC CLASS USERDAO {  PUBLIC VOID SAVEUSER(USER USER) {  // LÓGICA DE ALMACENAMIENTO EN BASE DE DATOS AQUÍ  }  } |

2. Define las interfaces entre capas

Para garantizar la separación de responsabilidades, se deben definir las interfaces entre las capas. Por ejemplo, la capa de presentación debe comunicarse con la capa de lógica de negocio por medio de una interfaz clara.

|  |
| --- |
| // CAPA DE PRESENTACIÓN - HTML Y JAVASCRIPT  FUNCTION SUBMITFORM () {  VAR NAME = DOCUMENT.GETELEMENTBYID("NAME").VALUE;  VAR EMAIL = DOCUMENT.GETELEMENTBYID("EMAIL").VALUE;  // LLAMADA A FUNCIÓN DE LA CAPA DE LÓGICA DE NEGOCIO  VAR USER = CREATEUSER (NAME, EMAIL);  // LLAMADA A FUNCIÓN DE LA CAPA DE ACCESO A DATOS SAVEUSER(USER);} |

|  |
| --- |
| // CAPA DE LÓGICA DE NEGOCIO - JAVA  PUBLIC CLASS USERSERVICE {  PUBLIC USER CREATEUSER (STRING NAME, STRING EMAIL) {  // VALIDACIÓN Y PROCESAMIENTO DE DATOS AQUÍ  RETURN NEW USER (NAME, EMAIL);  }  }  // CAPA DE ACCESO A DATOS - JAVA  PUBLIC CLASS USERDAO {  PUBLIC VOID SAVEUSER (USER USER) {  // LÓGICA DE ALMACENAMIENTO EN BASE DE DATOS AQUÍ  }  } |

3. Crea los servicios en cada capa

Cada nivel debe proporcionar servicios específicos para realizar su tarea. Por ejemplo, se pueden crear servicios en la capa de presentación para manejar la autenticación del usuario y la validación de los datos de entrada; en la capa de lógica de negocio, para procesar la información y realizar cálculos complejos, y en la capa de acceso a datos, para almacenar y recuperar información de la base de datos.

|  |
| --- |
| // CAPA DE PRESENTACIÓN - HTML Y JAVASCRIPT  FUNCTION SUBMITFORM () {  VAR NAME = DOCUMENT.GETELEMENTBYID("NAME").VALUE;  VAR EMAIL = DOCUMENT.GETELEMENTBYID("EMAIL").VALUE;  // LLAMADA A SERVICIO DE LA CAPA DE LÓGICA DE NEGOCIO  VAR USER = USERSERVICE.CREATEUSER(NAME, EMAIL);  // LLAMADA A SERVICIO DE LA CAPA DE ACCESO A DATOS USERDAO.SAVEUSER(USER);  } |

|  |
| --- |
| // CAPA DE LÓGICA DE NEGOCIO - JAVA  PUBLIC CLASS USERSERVICE {  PUBLIC STATIC USER CREATEUSER (STRING NAME, STRING EMAIL) {  // VALIDACIÓN Y PROCESAMIENTO DE DATOS AQUÍ  RETURN NEW USER (NAME, EMAIL);  }  }  // CAPA DE ACCESO A DATOS - JAVA  PUBLIC CLASS USERDAO {  PUBLIC STATIC VOID SAVEUSER (USER USER) {  // LÓGICA DE ALMACENAMIENTO EN BASE DE DATOS AQUÍ  }  } |

4. Implementa las funciones en cada capa

Las funciones se implementan en cada capa para proporcionar la funcionalidad específica de esa capa. Por ejemplo, en la capa de presentación se pueden implementar funciones para mostrar la información en una página web y para manejar la entrada del usuario; en la capa de lógica de negocio, para procesar la información y realizar cálculos complejos, y en la capa de acceso a datos, para almacenar y recuperar información de la base de datos.

|  |
| --- |
| PUBLIC CLASS PRODUCT {  PRIVATE INT ID;  PRIVATE STRING NAME;  PRIVATE DOUBLE PRICE;  // CONSTRUCTOR, GETTERS AND SETTERS  PUBLIC VOID SAVE() {  // CODE TO SAVE THE PRODUCT IN THE DATABASE }  PUBLIC STATIC PRODUCT FINDBYID(INT ID) {  // CODE TO FIND A PRODUCT BY ID IN THE DATABASE  }  PUBLIC STATIC LIST<PRODUCT> FINDALL() {  // CODE TO RETRIEVE ALL PRODUCTS FROM THE DATABASE  }  PUBLIC VOID UPDATE() {  // CODE TO UPDATE THE PRODUCT IN THE DATABASE  }  PUBLIC VOID DELETE() {  // CODE TO DELETE THE PRODUCT FROM THE DATABASE  }  } |

WebGrafia:

<https://blog.hubspot.es/website/que-es-arquitectura-en-capas#:~:text=La%20arquitectura%20en%20capas%20es,adyacentes%20mediante%20interfaces%20bien%20definidas>.

<https://chat.openai.com/>

<https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/capas>