MODEL-VIEW-CONTROLLER

Ariadna Gamez, Carlota Corcuera, Adrián Patiño

ÍNDEX

01. Què és MVC?

- Patró de disseny arquitectònic de software que organitza la lògica d'una aplicació en diferents capes. Les capes interactuen entre elles.

- Originat en la dècada de 1970.

- Separa la lògica de negoci i la gestió de dades de la interfície d'usuari -> codi més:
  - Modulable
  - Reutilitzable
  - Testejable
  - Mantenible

- Utilitzat en aplicacions amb interfície d'usuari com aplicacions d'escriptori, web, mòbils, aplicacions de jocs, etc.

- Llenguatges i entorns utilitzats: Java -> Spring, Python, JavaScript, C# -> ASP.NET, PHP -> Laravel, etc.

02. Problemes que intenta resoldre

01. MANTENIMENT I ESCALABILITAT

En aplicacions complexes, el codi es barreja dificultant el manteniment i l'escalabilitat.

02. SEPARACIÓ DE RESPONSABILITATS

Manca de separació de responsabilitats: Lògica de negoci, gestió de dades i presentació de la interfície solen estar acoblats.

03. REUTILITZACIÓ LIMITADA

Si la lògica està fortament vinculada a una interfície concreta, resulta difícil reutilitzar aquest codi en altres aplicacions o interfícies.

04. TESTEJABILITAT

És complicat provar el comportament del sistema de forma aïllada quan la lògica de negoci i la presentació estan combinades.

03. Solució al problema

3 CAPES:

01. Model

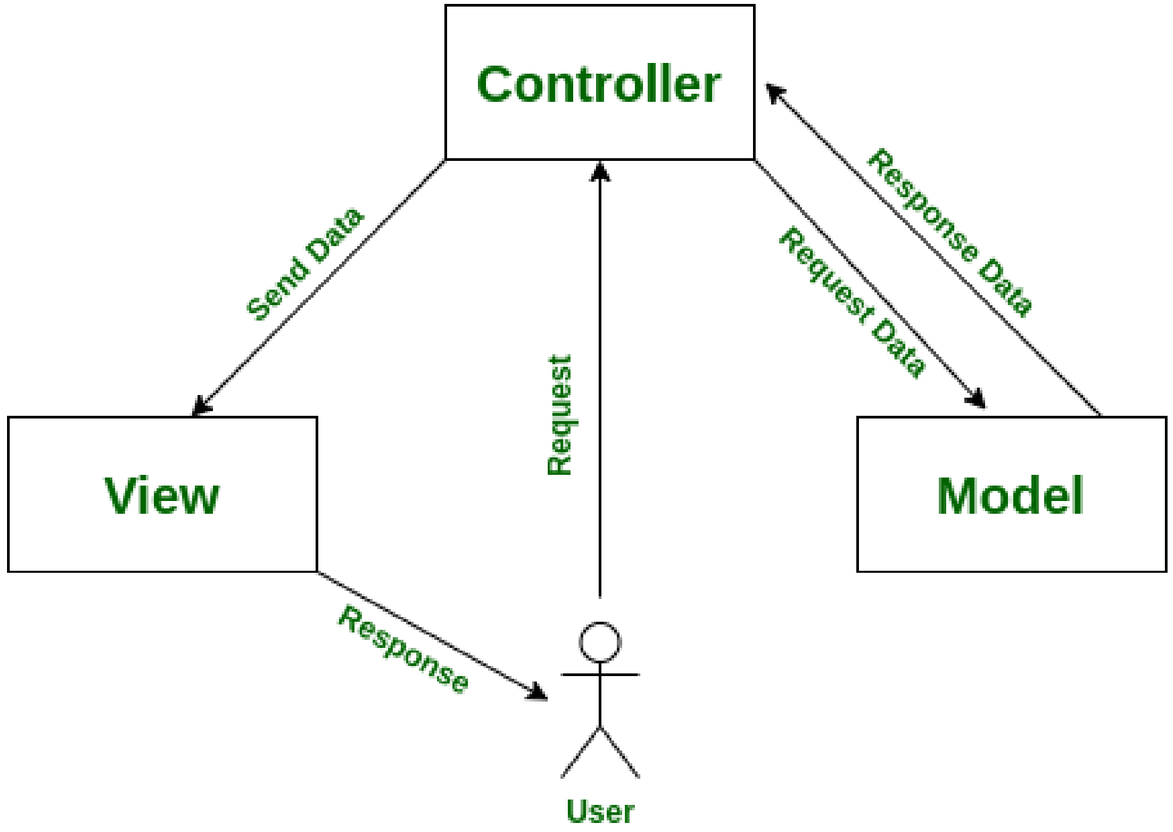
Responsable de la lògica de dades de l'aplicació i d'emmagatzemar i recuperar dades.

02. View

Interfície d'usuari necessària per interactuar amb l'aplicació. Presentació visual de les dades i interacció dels usuaris amb aquestes.

03. Controller

Actua com a intermediari entre el Model i la Vista. Cervell de l'aplicació, mantenint-ho tot en moviment i sincronitzat.



04. Avantatges i inconvenients

AVANTATGES

- Desenvolupaments de les aplicacions més ràpids i eficients -> aplicacions grans i complexes.
- Divisió de la carga de treball entre les capes -> productivitat augmenta
- Compatible amb el treball de diferents desenvolupadors sobre diferents capes simultàniament. Reutilitzar components.
- Desenvolupament basat en proves: es poden provar i solucionar problemes de les capes individualement.
- Permet utilitzar diferents tecnologies per cada capa.

INCONVENIENTS

- Augmenta la complexitat i sobrecàrrega del seu codi.
- Interdependència: les capes poden acabar depenent l'un de l'altre tot i que el principal objectiu es separar-los.

05. Exemple d'aplicació

Cada component realitza una tasca específica

PÁGINA WEB D'UN E-COMMERCE

MODEL (INVENTARI)

Gestiona els productes, preus i stock.

VISTA (INTERFÍCIE D'USUARI)

Mostra els productes disponibles, permet agregar articles al carretó.

CONTROLADOR (GESTOR DE COMPRES)

Procesa les entrades del usuari (ex: afegir un producte al carretó)
Solicita al model informació de l'inventari i actualitza la vista mostrant l'estat actualitzat del carretó.

06. Bibliografia

https://www.techtarget.com/whatis/definition/model-view-controller-MVC
https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm
https://www.interserver.net/tips/kb/mvc-advantages-disadvantages-mvc/
https://codigofacilito.com/articulos/mvc-model-view-controller-explicado
https://www.linkedin.com/advice/3/what-benefits-drawbacks-using-model-view-controller