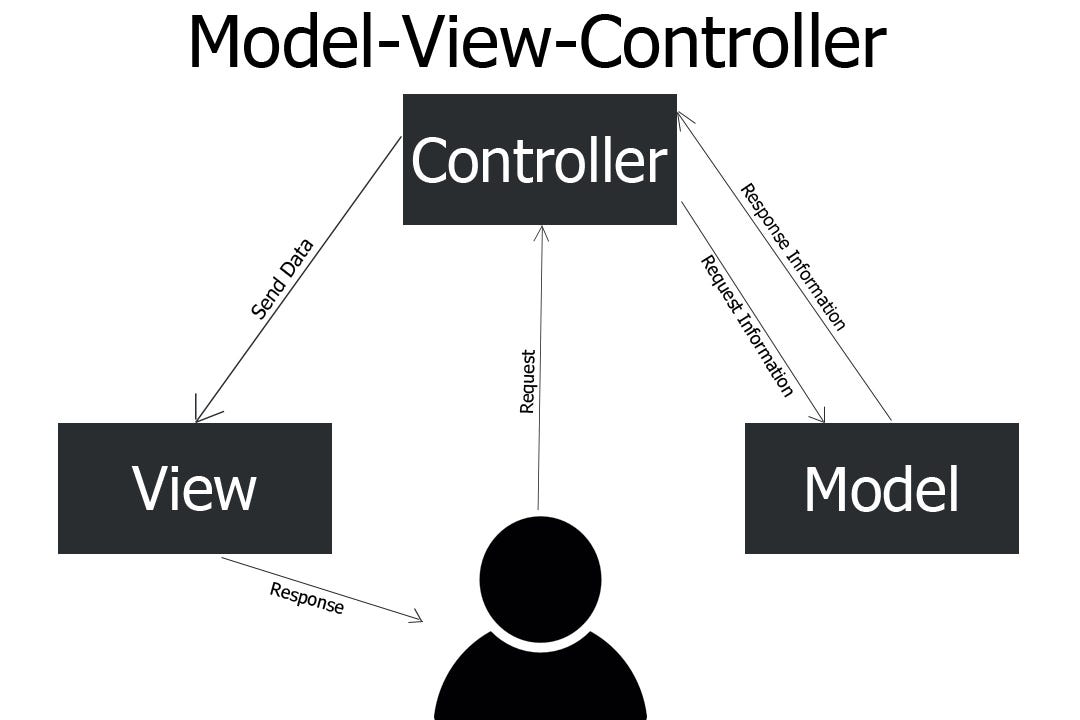
1. **MVC 🡺 Model , View , Control**

* **عبارة عن Architectural Pattern في السوفت وير بستخدمه علشان انظم الكود بتاعي وافصل الصفحات عن بعضها عشان الكود يكون اسهل في القراءة وسهل التعديل عليه**
* Model :
* بيتعامل مع الداتا بيزي ياخد request من controller ويشوف الداتا بيز ويرجع البيانات المطلوبة
* View :
  + الصفحات اللي المستخدم بيشوفها في البرنامج او الموقع
* Controller :
* بيربط view مع model بياخد action من user يبعته لل modelبعدين ياخد الرد من model يرجعه لل view

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. **SOILD :**

* **S - Single-responsibility Principle** 
  + **بخلي الكلاس مسئول عن حاجة واحد ويكون ليه سبب واحد اني اغير فيه**
  + يعني مثلا لو عندي كلاس فيه معلومات عن student محطش فيه بيانات عن منتجات في سوبر ماركت
  + او مثلا مينفعش يكون عندي دكتور في كل التخصصات الأفضل كل دكتور يكون ليه تخصص منفصل
* **O - Open-closed Principle** 
  + **اقدر اضيف حاجة جديدة في الكلاس من غير ما اعدل في الأساس**
  + مثال : الخلاص بيكون ثابت واقدر اغير الحاجة اللى تركب عليه بسهوله وكل حاجة بيكون ليها وظيفة مختلفة
* **L - Liskov Substitution Principle** 
  + **لو عندي كلاس وارث من كلاس تاني لازم الابن يقدر يعمل كل حاجة الاب بيعملها**
* **I - Interface Segregation Principle**
  + **معتمدش علي حاجة مش محتاجها**
  + لو عندي interface بيعمل كذا حاجة الأفضل اعمل كذا interface وكل واحد بينفذ حاجة معينة وقتها استخدم interface اللى انا محتاجه فقط
* **D - Dependency Inversion Principle**
  + **لما اعتمد على حاجة اعتمد عليها نفسها مش على الحاجة اللى بتقدمها**
  + مثال : لو عندي فيشة في الحيطة اعتمد علي الفيشة نفسها مش على نوع الكابل ولا الفيشة اللى بركبه فيها