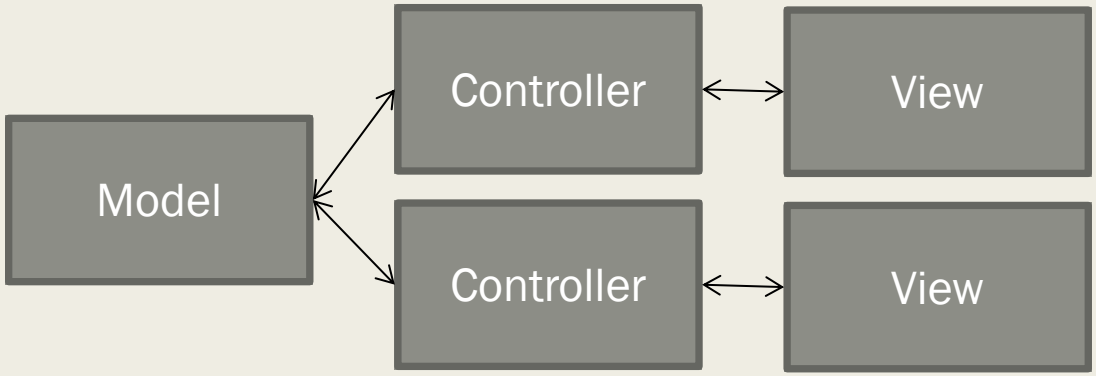
DESIGN PATTERNS

MVC

- Architecture level pattern that gives advice on how a solution to a problem should be structured.
- Model-View-Controller
- Layered design with each part in the solution having a single concern.
- Separation of concerns



Views

- A component that has the responsibility of interacting with a user.
- In general it is something that is visible.
  - *It is possible that we can have clients that are not people, but other systems.*
  - *Need an interface that allows interaction.*

Models

- A component that has the responsibility of keeping information.
- The model encapsulates the data
  - *Data is not directly accessible. (private)*
  - *Some operations are not allowed.*
  - *Implementation is hidden behind a well defined public interface.*
- Business logic should be the responsibility of the model, but in practice the lines can blur.
- Example: If I type in a deposit value into an input element, who should have the responsibility of enforcing that the value is positive?
  - *The element*
  - *Some JS in the page*
  - *Some JS in the server*
  - *Some code in our data model*

Controllers

- A component that has the responsibility of mediating between the view and the model..

CRUD

- Basic operations we can perform on a persistant data store.
  - *Create*
  - *Read*
  - *Update*
  - *Delete*
- Controller decides which view it should serve to allow interaction with the model. Controller updates model. Controller updates view as needed.

MVVM

- Model – View – View Model.
- Another architecture pattern that defines responsibilities. The goal is to reassign responsibilities to reduce the size of the controllers.
- View Models transform model data into items that can be displayed on a view.
- The model only interacts with the view models.
- Controllers exist, but have limited responsibility. May interact with view and view model.