**MVC**

Model - мозг приложения. Она выполняет все вычисления и создает для себя виртуальный мир, в котором может существовать сама, без вычислений видов и контролеров. Иными словами, вы можете считать модель виртуальной копией приложения без интерфейса.

View (вид) - окно, через которое пользователь взаимодействует с вашим приложением. В большинстве случаев вид отображает содержимое модели, но, кроме того, он же воспринимает и действия пользователя. Любые контакты между пользователем и вашим приложением отправляются в вид. После этого они могут уже быть перехвачены controller и переданы в Model.

Controller - контролеры видов (ViewControllers), которые упоминались выше. Контролер является переходным звеном между model & view. Он интерпретирует события, происходящие с одной стороны, и по мере необходимости использует эту информацию для внесения изменений на другой стороне. Например: если пользователь изменяет какое-либо поле в виде, то контроллер гарантирует, что и модель измениться соответствующим образом. А если модель получит новые данные (например, из интернета), то котроллер прикажет виду отобразить их.

**Жизненный цикл UIView**

1. init(coder:) и init(frame:)

* init(frame:) вызывается при создании объекта вида программно, когда вы явно создаете экземпляр вида в коде.
* init(coder:) вызывается при создании объекта вида из Interface Builder или при декодировании из архива.

1. awakeFromNib вызывается после завершения загрузки всех объектов из nib-файла (если вид был загружен из Interface Builder).
2. layoutSubviews вызывается, когда размеры или расположение подвижных элементов (subviews) изменяются. Этот метод также вызывается при первом добавлении вида в иерархию представлений.
3. willMove(toSuperview:) и didMoveToSuperview

* willMove(toSuperview:) вызывается перед тем, как вью *будет* добавлено или удалено из иерархии представлений. Параметр указывает на новое представление (если добавляется) или nil (если удаляется).
* didMoveToSuperview вызывается после того, как вью *было* добавлено или удалено из иерархии представлений.

1. willMove(toWindow:) и didMoveToWindow

* willMove(toWindow:) вызывается перед тем, как вью *будет* добавлено или удалено из окна. Параметр указывает на новое окно (если добавляется) или nil (если удаляется).
* didMoveToWindow вызывается после того, как вью *было* добавлено или удалено из окна.

1. didAddSubview и willRemoveSubview

* didAddSubview: вызывается *после* того, как подвижный элемент (subview) был добавлен в иерархию представлений.
* willRemoveSubview: вызывается *перед* тем, как подвижный элемент будет удален из иерархии представлений.

1. didReceiveMemoryWarning вызывается, когда система сообщает о нехватке памяти. Здесь можно освобождать ресурсы, которые можно восстановить позже.
2. layoutIfNeeded заставляет немедленно выполнить расчет и применение раскладки (layout), если это необходимо. Например, если у вас есть изменения, которые не вызывают автоматического обновления раскладки.
3. setNeedsLayout и layoutIfNeeded

* setNeedsLayout *сообщает* системе о том, что требуется пересчет и обновление раскладки в следующем цикле отображения.
* layoutIfNeeded *принудительно вызывает* пересчет и обновление раскладки, если они были отложены.

**Жизненный цикл UIViewController**

1. Init() – инициализация. XCODE собирает данные о приложении, выделяет память;
2. loadView() – создание и инициализация иерархии представлений, загрузка всех View приложения;
3. viewDidLoad() – начальная настройка интерфейса, загрузка данных и инициализация, которые могут быть выполнены после загрузки UIView, но до отображения на экране, завершает загрузку всех View;
4. viewWillAppear() – вызывается для выполнения дополнительных настроек перед отображением контроллера на экране, View начинает появляться;
5. viewDidappear() – вызывается после отображения контроллера на экране / для выполнения задач, которые требуют видимости контроллера на экране, View заканчивает появляться;
6. viewWillDisappear() – вызывается перед скрытием контроллера;
7. viewDidDisappear() – для выполнения доп.задач, связанных с тем, что пользователь больше не видит приложения, View не отображается, но выделенные для него ресурсы доступны;
8. deinit() – деинициализация - освобождение ресурсов и выполнения очистки, таких как отмена подписок или освобождение памяти.