УДК 004.4

Методологии и механизмы Auto Layout в разработке пользовательских интерфейсов

А.П. Сазонов

Kaseya Poland LLC, Краков, Польша

Статья посвящена анализу технологии Auto Layout, введённой в 2012 году, как фундаментального события в развитии интерфейсного дизайна. Исследование рассматривает ключевые аспекты оптимизации процесса верстки интерфейса с помощью Auto Layout в контексте споров и дебатов, охватывающих применение технологии в графическом интерфейсе и через программный код. В работе освещается проблематика выбора методов верстки и влияние этого выбора на разработку интерфейсов. Автор анализирует различные подходы к созданию пользовательского интерфейса, обсуждая преимущества и недостатки использования Auto Layout по сравнению с традиционными методами на основе фреймов и сторонних решений. Работа подробно рассматривает механизмы Auto Layout для динамического вычисления размера и позиции элементов интерфейса, а также управления внутренними и внешними изменениями в макете приложения. Особое внимание уделяется методологии настройки и обновления ограничений, необходимых для адаптивного дизайна интерфейсов под различные устройства и размеры экранов.

Ключевые слова: Autolayout, современные технологии, цифровизация, IT, верстка интерфейса.

Auto Layout methodologies and mechanisms in user interface development

A.P. Sazonov

Kaseya Poland LLC, Krakow, Poland

The scientific article is devoted to the analysis of Auto Layout technology, introduced in 2012, as a fundamental moment in the development of interface design. The study examines the key aspects of optimizing the interface layout process using Auto Layout in the context of disputes and debates covering the application of technology in the interface builder and through the program code. The paper highlights the problems of choosing layout methods and the impact of this choice on the development of interfaces. The author analyze various approaches to creating a user interface, discussing the advantages and disadvantages of using Auto Layout compared to traditional frame-based methods and third-party solutions. The work examines in detail the Auto Layout mechanisms for dynamically calculating the size and position of interface elements, as well as managing internal and external changes in the application layout. Special attention is paid to the methodology of configuring and updating the constraints necessary for the adaptive design of interfaces for various devices and screens.

Keywords: Autolayout, modern technologies, digitalization, IT, interface layout.

Введение

Разработка пользовательских интерфейсов (UI) – важный аспект разработки программного обеспечения, служащий мостом между пользователем и приложением. Эффективность, удобство использования и эстетическая привлекательность пользовательского интерфейса напрямую влияют на удовлетворенность и вовлеченность пользователей. Таким образом, методологии и механизмы,

используемые при проектировании и реализации пользовательского интерфейса, имеют первостепенное значение. Среди них Auto Layout – система, которая динамически вычисляет размер и положение всех представлений в иерархии представлений на основе ограничений, наложенных на эти представления, – стала важной областью внимания.

Проблемное пространство, решаемое методологиями Auto Layout, вращается вокруг необходимости создания гибких, адаптивных и масштабируемых пользовательских интерфейсов для широкого диапазона размеров, ориентаций и разрешений устройств. Традиционные подходы к проектированию пользовательского интерфейса часто включают статическое позиционирование, что может привести к таким проблемам, как перекрытие элементов, плохое использование пространства и отсутствие реакции на изменения размера или ориентации экрана. Эти проблемы препятствуют разработке приложений, обеспечивающих оптимальное взаимодействие с пользователем на нескольких устройствах, что становится все более важным в современной разнообразной экосистеме вычислительных устройств [1, 2].

Мотивация для решения проблемы статического дизайна пользовательского интерфейса двоякая. Во-первых, повышение адаптивности пользовательских интерфейсов позволяет сделать использование приложений более инклюзивным и доступным, удовлетворяя более широкую аудиторию с различными предпочтениями в отношении устройств. Во-вторых, эффективные и масштабируемые методологии проектирования пользовательского интерфейса способствуют сокращению времени и ресурсов разработки, поскольку они сводят к минимуму необходимость ручной настройки и перепроектирования для разных размеров и ориентаций экрана.

В данной статье представлены решения вышеупомянутых проблем путем изучения методологий и механизмов Auto Layout при разработке пользовательского интерфейса. Целью статьи является предоставление всестороннего понимания Auto Layout, включая его принципы, лучшие практики и стратегии реализации.

1. Материалы и методы

Auto Layout основан на концепции системы макетов на основе ограничений, что представляет собой сдвиг парадигмы по сравнению с традиционными макетами на основе фреймов. В этой системе пространственные отношения между элементами пользовательского интерфейса определяются ограничениями, а не их абсолютными позициями. Этот подход облегчает разработку интерфейсов, которые быстро реагируют и адаптируются к экранам различных размеров и ориентаций.

В основе Auto Layout лежит понятие ограничений. Ограничение – это математическое представление связи между двумя элементами пользовательского интерфейса или между элементом пользовательского интерфейса и родительским представлением. Эти отношения определяют такие аспекты, как размер, положение и выравнивание. Ограничения могут быть простыми, например «Элемент А находится на расстоянии 20 точек от верхнего края элемента В», или более сложными, включающими соотношения и множители, например: «Ширина элемента А равна половине ширины элемента В».

Auto Layout представляет несколько ключевых элементов и инструментов, которые помогают разработчикам определить эти ограничения (табл. 1).

Таблица 1

Элементы и инструменты Auto Layout

Название	Описание	Применение
Атрибуты ограничений	Включают размеры, такие как ширина и высота, а также позиции (впереди, сзади, сверху, снизу, по центру по оси X, по центру по оси Y). Ограничения определяются путем установления отношений между атрибутами различных элементов пользовательского интерфейса.	Используются для определения взаимного расположения и размеров элементов пользовательского интерфейса.
Уровни приоритета	Не все ограничения равны; некоторые могут конфликтовать с другими. Auto Layout позволяет разработчикам назначать уровни приоритета ограничениям, от обязательных до необязательных, с числовыми значениями, указывающими их важность.	Позволяет системе Auto Layout разрешать конфликты, игнорируя или изменяя ограничения с более низким приоритетом.
Сопротивление растяжению и сжатию	Эти свойства определяют, насколько плотно представление должно прилегать к его внутреннему размеру контента. Сопротивление растяжению предотвращает рост представления сверх его естественного размера, в то время как сопротивление сжатию предотвращает его сжатие до слишком малого размера.	Критически важны для управления представлениями с динамичным содержимым, например, текстом.
Внутренний размер контента	Некоторые элементы пользовательского интерфейса имеют естественный размер, определяемый их содержимым (например, размер кнопки на основе текста её метки). Auto Layout использует этот внутренний размер контента как фактор при расчете компоновок.	Облегчает управление представлениями с динамичным содержимым.
Язык визуального форматирования (VFL)	VFL предоставляет текстовое представление ограничений, позволяя разработчикам определять ограничения в компактной, удобочитаемой форме.	Ценен для программного создания ограничений, особенно когда графические интерфейсы недоступны.
Руководства по компоновке и безопасные зоны	Auto Layout включает руководства по компоновке и вставки безопасной зоны, помогающие разработчикам избегать размещения контента в областях, закрытых другими элементами (например, вырез на iPhone) или за пределами видимой области экрана.	Используются для обеспечения адекватного расположения контента в пределах видимой области экрана, особенно на устройствах с особенностями дизайна, как вырез на экране.
Стековые представления	Введены для упрощения процесса управления группой представлений, стековые представления автоматически управляют компоновкой своих дочерних элементов на основе их оси, распределения, выравнивания и свойств отступов.	Упрощает управление группами представлений, автоматически настраивая их компоновку в зависимости от заданных параметров.

Механизм Auto Layout обрабатывает эти ограничения для вычисления размера и положения всех представлений в пользовательском интерфейсе. Эти вычисления происходят в многопроходной системе макета, которая учитывает внутренние размеры, уровни приоритета и внешние изменения (например, поворот экрана или изменение размера окон).

Для упрощения понимания представим, что существует система Auto Layout, которая принимает на вход ограничения (ограничения, созданные в Xcode) и размер экрана приложения (который зависит от разрешения экрана устройства iOS и ориентации).

В таблице 2 представлены официальные разрешения экранов различных устройств iOS.

Таблица 2

Официальные разрешения экранов последних устройств iOS

Модель	Логическое разрешение, пикс	Физическое разрешение, пикс	Соотношение сторон	Размер экрана, пикс	Плотность пикселей, пикс/дюйм
iPhone 15 Pro Max	430 × 932	1290 × 2796	9:19.5	6,7″	460
iPhone 15 Pro	393 × 852	1179 × 2556	9:19.5	6,1″	460
iPhone 15 Plus	430 × 932	1290 × 2796	9:19.5	6,7″	460
iPhone 15	393 × 852	1179 × 2556	9:19.5	6,1″	460

Для корректной разработки пользовательского интерфейса с использованием Auto Layout в iOS, необходимо понимать работу движка Auto Layout Engine, который рассчитывает позицию и размер элементов на основе ограничений (constraints) и размеров экрана.

Как показано на рисунке 1, Auto Layout Engine принимает на вход ограничения и размер экрана, чтобы вычислить положение и размер представлений, которые затем отображаются на экране.

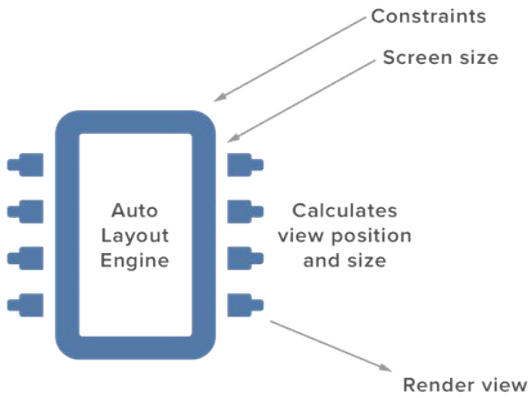

Рисунок 1. Auto Layout Engine

Для корректного отображения позиции и размеров элемента интерфейса системой Auto Layout, система должна знать или быть в состоянии рассчитать следующие параметры:

1. Горизонтальное положение элемента.
2. Вертикальное положение элемента.
3. Ширину элемента.
4. Высоту элемента.

Определение ограничений позволяет движку Auto Layout определить все вышеперечисленные параметры. Ниже представлено несколько примеров того, как Auto Layout рассчитывает позицию и размер элемента, используя простую математику (рис. 2).

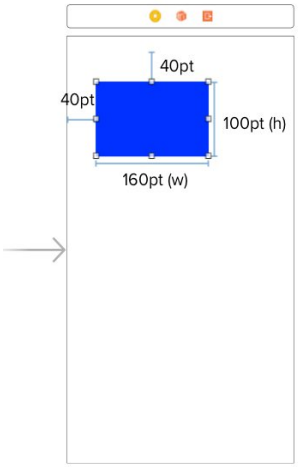

Рисунок 2. Пример расположения объекта

В данном примере определены четыре ограничения:

1. Расстояние от левого края экрана до левого края синего элемента составляет 40 пунктов (pt).
2. Расстояние от верхнего края экрана до верха синего элемента составляет 40 пунктов (pt).
3. Ширина синего элемента равна 160 пунктам (pt).
4. Высота синего элемента равна 100 пунктам (pt).

Исходя из первого ограничения, можно сделать вывод, что горизонтальная координата (x) элемента составляет 40. Исходя из второго ограничения, можно сделать вывод, что вертикальная координата (y) элемента составляет 40. Поскольку известны координаты x и y, а также ширина и высота элемента, система Auto Layout корректно отобразит его на экранах различного размера и ориентации, как представлено на рисунке 3.

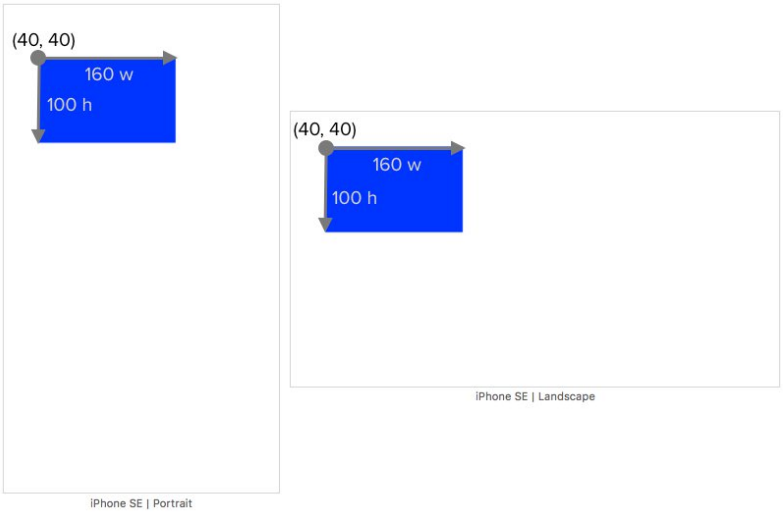

Рисунок 3. Принцип Auto Layout

Auto Layout динамически вычисляет размер и позицию всех элементов в иерархии представлений, основываясь на ограничениях, наложенных на эти элементы. Например, можно задать ограничение для кнопки таким образом, чтобы она была горизонтально центрирована относительно изображения, а верхний край кнопки всегда находился на расстоянии 8 пунктов ниже нижнего края изображения. Если размер или положение изображения изменяются, положение кнопки автоматически корректируется соответственно.

Такой подход к дизайну на основе ограничений позволяет создавать пользовательские интерфейсы, которые динамически реагируют как на внутренние, так и на внешние изменения.

Если говорить о внешних изменениях, то они происходят, когда изменяется размер или форма супервью. С каждым таким изменением необходимо обновлять макет иерархии представлений для наилучшего использования доступного пространства. Вот некоторые общие источники внешних изменений:

- пользователь изменяет размер окна (OS X);
- пользователь активирует или выходит из режима Split View на iPad (iOS);
- устройство меняет ориентацию (iOS);
- появляются или исчезают панели активного вызова и аудиозаписи (iOS);
- необходимо поддерживать различные размеры классов;
- необходимо поддерживать различные размеры экранов.

Большинство этих изменений может происходить во время выполнения, и они требуют динамического ответа от приложения. Другие, как поддержка различных размеров экранов, представляют адаптацию приложения к различным средам. Хотя размер экрана обычно не изменяется во время выполнения, создание адаптивного интерфейса позволяет приложению работать одинаково хорошо как на iPhone 11, так и на iPhone 15 Pro Max или даже на iPad. Auto Layout также является ключевым компонентом для поддержки функций Slide Over и Split Views на iPad.

Внутренние изменения происходят, когда изменяется размер элементов или управления в пользовательском интерфейсе.

Некоторые общие источники внутренних изменений:

- содержимое, отображаемое приложением, меняется;
- приложение поддерживает интернационализацию;
- приложение поддерживает Dynamic Type (iOS).

Когда содержимое приложения меняется, новое содержимое может требовать другой компоновки. Это часто встречается в приложениях, отображающих текст или изображения. Например, новостное приложение должно адаптировать свой макет в зависимости от размера отдельных новостных статей. Аналогично фотоколлаж должен обрабатывать широкий спектр размеров изображений и соотношений сторон.

Интернационализация – это процесс адаптации приложения к различным языкам, регионам и культурам [7]. Макет интернационализированного приложения должен учитывать эти различия и корректно отображаться на всех языках и регионах, которые поддерживает приложение.

Интернационализация оказывает три основных эффекта на макет.

Во-первых, при переводе пользовательского интерфейса на другой язык метки требуют разного количества пространства. Например, для немецкого языка, как правило, требуется значительно больше места, чем для английского. Японский часто требует гораздо меньше.

Во-вторых, формат, используемый для представления дат и чисел, может изменяться от региона к региону, даже если язык остается неизменным. Хотя эти

изменения обычно более тонкие, чем изменения языка, пользовательский интерфейс все равно должен адаптироваться к небольшому изменению размера.

В-третьих, изменение языка может повлиять не только на размер текста, но и на организацию макета. Разные языки используют разные направления компоновки. Например, английский использует направление компоновки слева направо, а арабский и иврит – справа налево. В целом, порядок элементов пользовательского интерфейса должен соответствовать направлению компоновки. Если кнопка находится в нижнем правом углу представления на английском языке, она должна быть в нижнем левом углу на арабском.

Наконец, если iOS-приложение поддерживает динамический тип, пользователь может изменить размер шрифта, используемого в приложении. Это может изменить как высоту, так и ширину любых текстовых элементов в пользовательском интерфейсе. Если пользователь изменяет размер шрифта во время работы вашего приложения, как шрифты, так и макет должны адаптироваться [3].

В контексте разработки под iOS, задача выравнивания содержимого и определения межкомпонентных интервалов может стать времязатратным процессом. Важно обратить внимание на методологию настройки ограничений при помощи UIKit, их последующее обновление, а также решение возникающих конфликтов между ограничениями. Ограничение (constraint) является сущностью, которая предоставляет возможность операционной системе определенным образом позиционировать элементы пользовательского интерфейса в рамках экрана. В рамках UIKit представлены две принципиальные стратегии установки таких ограничений: использование графического интерфейса и программное определение. Настоящий анализ сфокусирован на программном подходе к управлению ограничениями.

Создание нового ограничения начинается с деактивации AutoresizingMask, который активирован по умолчанию. Это действие является критическим шагом, поскольку обеспечивает основу для дальнейшей настройки и гибкого управления компонентами пользовательского интерфейса в пределах системы UIKit.

После создания нам нужно активировать ограничения. Для этого можно переключить ограничение на isActive по одному, либо использовать activate() функцию для одновременного обновления пакета ограничений. Так же можно деактивировать их таким же образом.

Ограничения можно обновлять после применения. Например, для изменения интервала или размера (рис. 4).

```
let heightConstraint = view.heightAnchor.constraint(equalToConstant: 100)
heightConstraint.constant = 50
```

Рисунок 4. Пример кода, для управления ограничениями

Переходя к вопросу урегулирования конфликтов между ограничениями, стоит отметить, что после освоения процессов добавления, модификации и активации ограничений, разработчики часто сталкиваются с необходимостью решения проблем, связанных с взаимоисключающими ограничениями. В таких ситуациях инструментарий Xcode обычно предоставляет указания о невозможности корректного разрешения ограничений через вывод в консоль и раздел Reports. Для детального анализа и идентификации корней проблемы предлагается использование инструмента Debug View Hierarchy, который позволяет визуально осмотреть каждое из ограничений и выявить источник конфликта. Этот метод дает возможность разработчикам глубже понять взаимодействие между элементами пользовательского интерфейса и наиболее

эффективно применять стратегии разрешения конфликтов ограничений в своих проектах.

При разработке интерфейсов, которые должны корректно отображаться на разнообразных устройствах, возникает необходимость учитывать специфические особенности отображения элементов управления, такие как статусная панель (status bar) и панель навигации (navigation bar). Различия в размерах и разрешениях экранов устройств могут существенно влиять на расположение и визуальное восприятие элементов пользовательского интерфейса.

UIKit предлагает решение этой проблемы через механизм safe area – безопасной зоны, которая автоматически адаптируется под размеры и особенности отображения различных устройств. Свойство safeAreaLayoutGuide позволяет разработчикам задавать ограничения (constraints) относительно этой безопасной зоны, обеспечивая тем самым, что важные элементы интерфейса не будут перекрыты системными элементами управления или выходить за пределы экрана.

Использование safeAreaLayoutGuide дает возможность обеспечить консистентное и корректное отображение интерфейса на разных устройствах, таких как iPhone 11 или iPhone 15 Pro Max, не вдаваясь в детали расчета специфических отступов и размеров для каждого устройства в отдельности. Например, при разработке можно определить ограничения для главного содержимого экрана так, чтобы оно автоматически адаптировалось и оставалось видимым и доступным на всех устройствах, не затрагивая при этом области, занятые системными элементами управления.

Таким образом, Auto Layout выступает не только как средство для упрощения разработки, но и как ключевой элемент для создания гибких и адаптивных интерфейсов, способных удовлетворять требованиям современного многообразия устройств и экранов [4].

В рамках развития интерфейсной разработки под iOS и macOS, система Auto Layout представляет собой ключевой инструмент, позволяющий разработчикам динамически адаптировать интерфейс приложения к различным устройствам и условиям отображения без необходимости ручной настройки размеров и позиционирования элементов. Auto Layout автоматически вычисляет размеры и положение всех элементов в иерархии представлений, основываясь на наборе ограничений (constraints), определенных для каждого элемента. Главное преимущество этого подхода заключается в его способности адаптироваться к изменениям, будь то изменение размеров экрана устройства, изменение ориентации экрана, различные размеры экранов разных устройств, а также изменения в контенте, связанные с локализацией или изменением данных.

Внешние изменения, которые учитывает Auto Layout, включают в себя, например, изменение размеров окна на macOS или изменение ориентации экрана на iOS, а также адаптацию к различным размерам экранов разных устройств. Внутренние изменения могут включать в себя динамическое изменение контента, например, в зависимости от пользовательских настроек или выбранного языка.

2. Создание Auto Layout и его составляющих

В процессе разработки пользовательского интерфейса с использованием системы Auto Layout, разработчики имеют возможность определять ограничения (constraints) тремя различными методами:

1. Создание ограничений с использованием сочетания клавиш CTRL и операции перетаскивания, например, между меткой (label) и верхней границей контейнера.

2. Применение инструментов Ember In, Align, Pin и Resolve Tools, доступных в пользовательском интерфейсе для управления разметкой.

Среди вышеупомянутых подходов особо выделяется второй, поскольку инструментарий, включающий в себя функции Ember In, Align и Pin, является центральным элементом в процессе проектирования интерфейса.

Функция Ember In (рис. 5) позволяет группировать выбранные элементы интерфейса в контейнер Ember In View, при этом Interface Builder адаптирует параметры Ember In View в зависимости от расположения и характеристик элементов. Создание Ember In View возможно также путем его перетаскивания из библиотеки объектов, подобно другим элементам интерфейса.

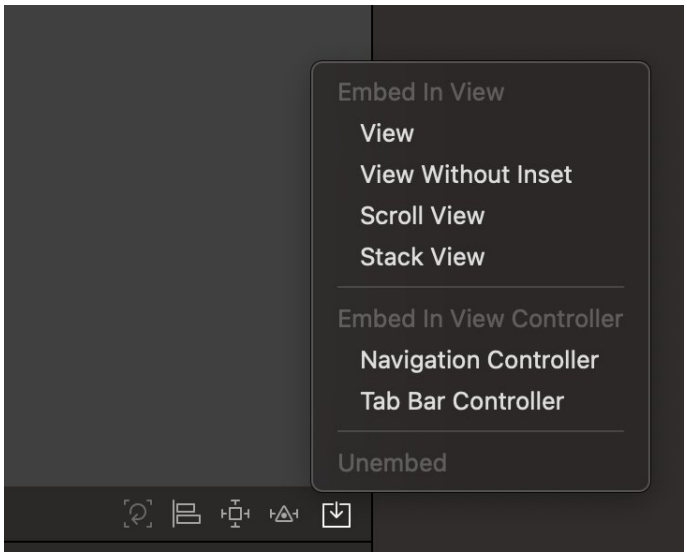

Рисунок 5. Функция Ember In

Инструмент Align (рис. 6) предоставляет возможность выравнивания элементов вдоль выбранной оси или границы, что аналогично выравниванию текста в текстовых редакторах – по центру или относительно краев страницы.

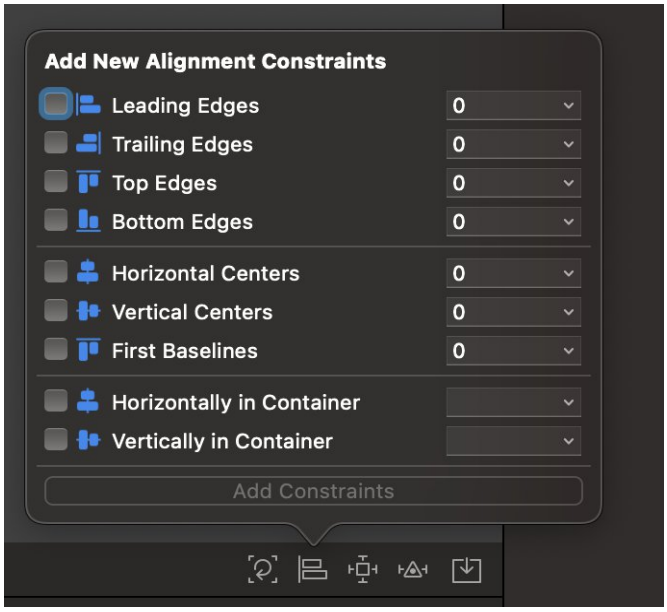

Рисунок 6. Функция Align

Pin (рис. 7), в свою очередь, дает возможность определения фиксированных границ для элементов интерфейса, позволяя устанавливать конкретные параметры ограничений в различных направлениях и обеспечивая их неизменность вне зависимости от размеров экрана.

Рисунок 7. Функция Pin

Resolve Tools (рис. 8) оказывается незаменимым помощником в процессе отладки ограничений, предлагая функционал для удаления, добавления или модификации ограничений, а также обновления положения объектов на экране.

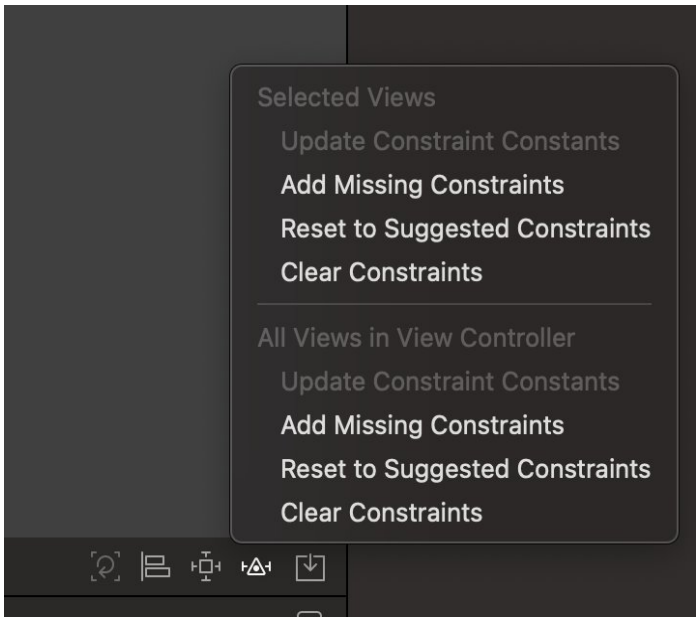

Рисунок 8. Resolve Tools

Модификация ограничений доступна через непосредственное взаимодействие с ними в Interface Builder, через Size Inspector или в списке Document Outline. Это позволяет разработчикам назначать идентификаторы для упрощения работы с ограничениями в процессе отладки.

Особое значение имеют параметры Content Hugging Priority и Compression Resistance Priority, влияющие на способность элементов изменять свои размеры в зависимости от контекста размещения. Эти параметры позволяют тонко настраивать поведение элементов в StackView и обеспечивают их адаптивность.

Следует отметить, что системы macOS и iOS используют различные подходы к расчету разметок: в то время как на macOS Auto Layout может изменять размеры окон и их содержимого, в iOS он ограничен изменением размеров содержимого в пределах заданных границ приложения [5, 6].

Заключение

Таким образом можно сказать, что Auto Layout значительно упрощает процесс разработки адаптивных интерфейсов, позволяя разработчикам легко адаптироваться к разнообразным размерам экранов и устройств. Эффективное использование ограничений и системы приоритетов в Auto Layout обеспечивает гибкость и масштабируемость интерфейсов, делая процесс разработки более интуитивно понятным и менее времязатратным. Однако с увеличением количества ограничений возникает сложность их управления, что требует от разработчиков глубоких знаний и понимания работы с инструментарием Xcode для эффективного решения возможных конфликтов.

СПИСОК ИСТОЧНИКОВ

1. Есть ли жизнь без Auto Layout? [Электронный ресурс] // Хабр. – URL: https://habr.com/ru/companies/oleg-bunin/articles/528328/ (дата обращения: 29.02.2024).

2. Auto Layout [Электронный ресурс] // DevTut. – URL: https://devtut.github.io/ios/auto-layout.html#space-views-evenly (дата обращения: 29.02.2024).

3. Auto Layout Guide: Understanding Auto Layout [Электронный ресурс] // Apple Developer Documentation Archive. – URL: https://developer.apple.com/library/archive/documentation/UserExperience/Conceptual/AutolayoutPG/index.html (дата обращения: 29.02.2024).

4. Auto Layout настройка кодом [Электронный ресурс] // Хабр. – URL: https://habr.com/ru/articles/690940/ (дата обращения: 29.02.2024).

5. Основы Auto Layout – Концепция, строение, применение [Электронный ресурс] // Хабр. – URL: https://habr.com/ru/articles/312782/ (дата обращения: 29.02.2024).

6. The best damn AutoLayout guide I’ve ever seen [Электронный ресурс] // GitHub. – URL: https://gist.github.com/chosa91/d2ab1be40c8207bedff9bfc8853ae6d7 (дата обращения: 29.02.2024).

7. About Internationalization and Localization [Электронный ресурс] // Apple Developer Documentation Archive. – URL: https://developer.apple.com/library/archive/documentation/MacOSX/Conceptual/BPInternational/Introduction/Introduction.html (дата обращения: 29.02.2024).

ИНФОРМАЦИЯ ОБ АВТОРАХ

Сазонов Артем Петрович, старший iOS разработчик, Kaseya Poland LLC, Краков, Польша.
e-mail: gigo12121212@gmail.com