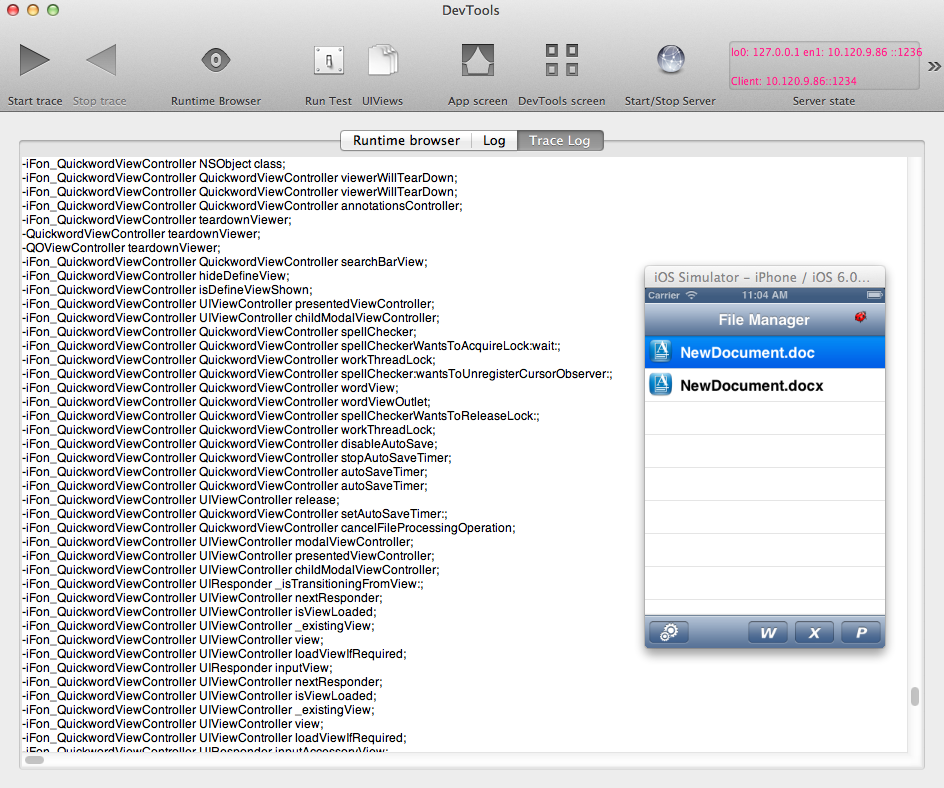
**DevTools - инструмент разработчика**

**Краткое описание и перечень возможностей**

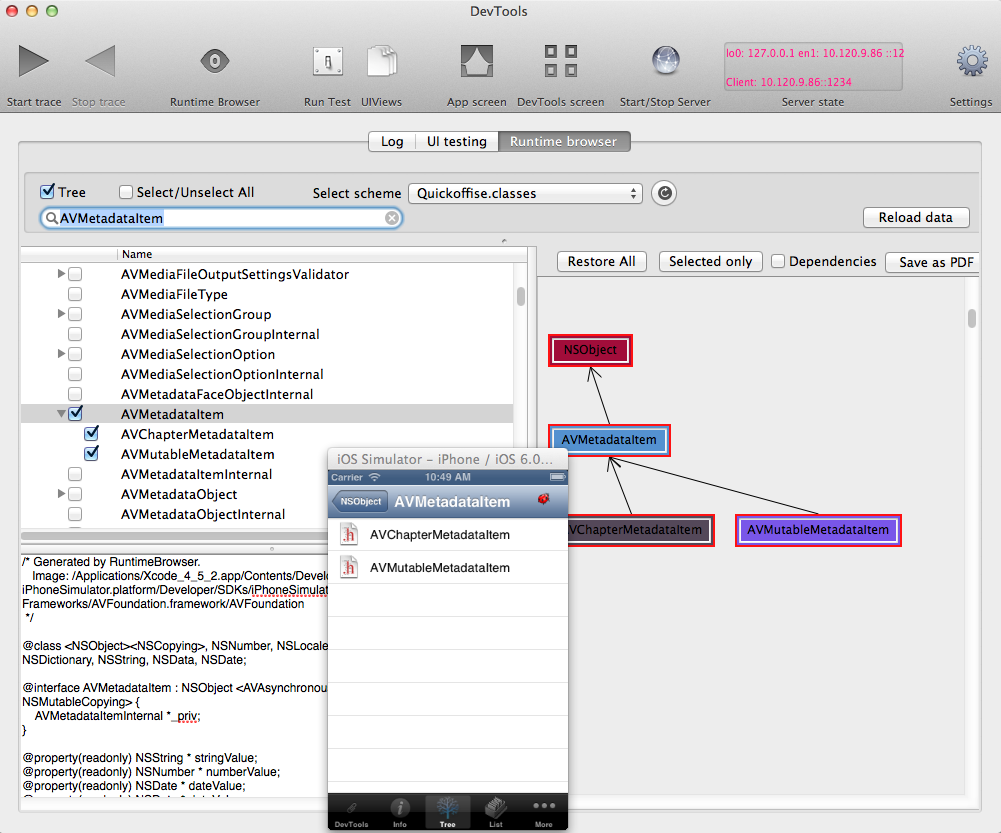
- **Трассировщик**

Позволяет отследить ход выполнения приложения путем логирования вызовов методов. Инструмент позволяет запустить на выполнение трассировщик на стороне клиента с возможностью логирования методов только определенных классов. Для создания такого списка классов используется Runtime browser (см. далее). Используется для исследования поведения классов и отладки.



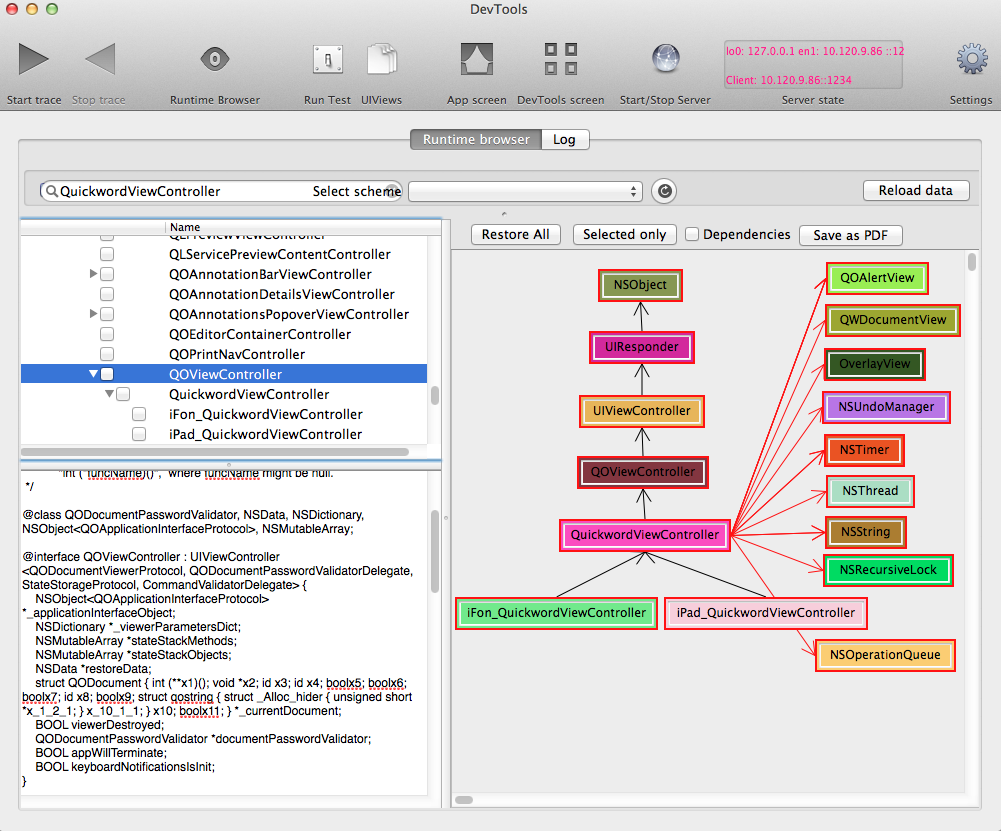
- **Runtime browser**.

Runtime библиотека ObjC позволяет во время выполнения приложения в частности формировать списки классов, методов и т.д. На этом основан Runtime browser. На стороне клиента его расширенная версия: дерево классов, список классов, заголовок класса (интерфейс, свойства, данные и т.д.), фреймворки, поиск класса. На стороне сервера выводится дерево классов, список классов, заголовок класса, поиск. Для каждого элемента дерева классов есть чек-бокс. Таким образом можно составить список классов. Используется для трассировщика. Можно сохранить и восстановить список помеченных классов из внешнего файла.



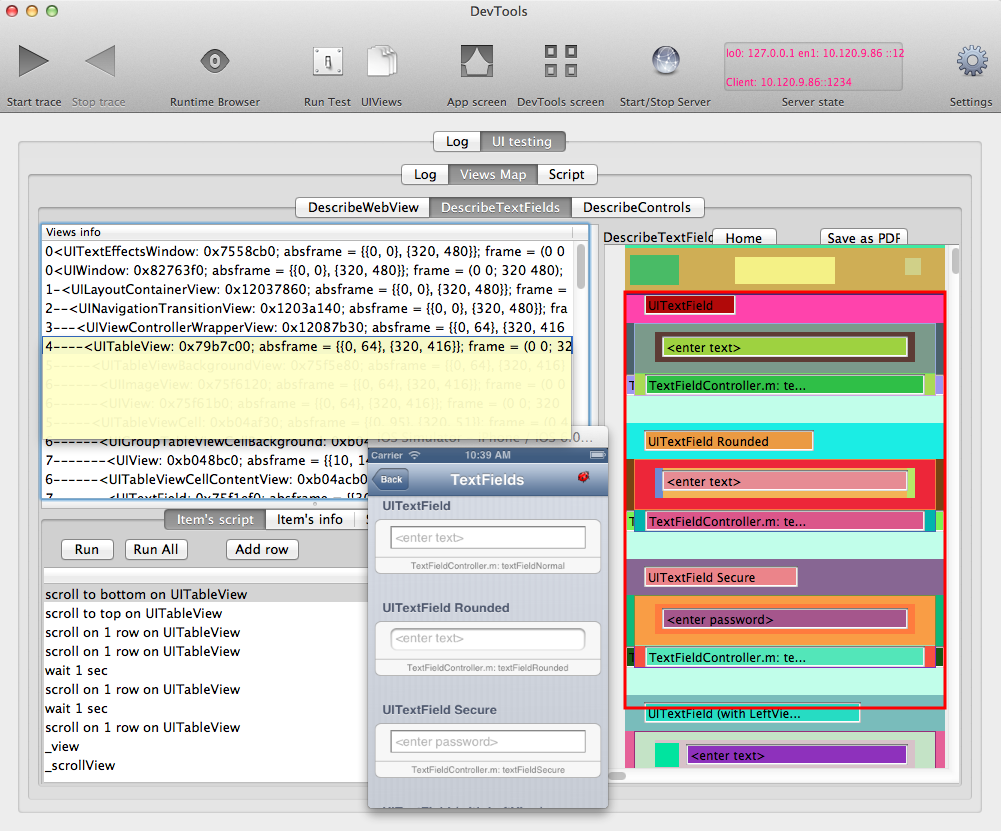
- **Диаграмма классов**

Графическая диаграмма наследования и зависимостей классов. Указав на элемент в дереве классов, инструмент строит для него диаграмму наследования. Дополнительно, для данного класса можно вывести диаграмму зависимостей. Имеется ввиду классы, от которых зависит данный класс. Можно использовать при разработке, а именно, при изменении интерфейса класса следует проверить возможные изменения в поведении зависимого класса. Например, как видно из скриншота, при изменениях в QWDocumentView следует проверить, как эти изменения могут отразиться на работе QuickwordViewController. Графическая диаграмма строится автоматически. Позволяет перемещать элементы диаграммы вручную. Архитектура инструмента позволяет менять путем внешних настроек цвета линий, классов, размеры блоков (в разработке). Можно сохранить диаграмму в PDF-файл.



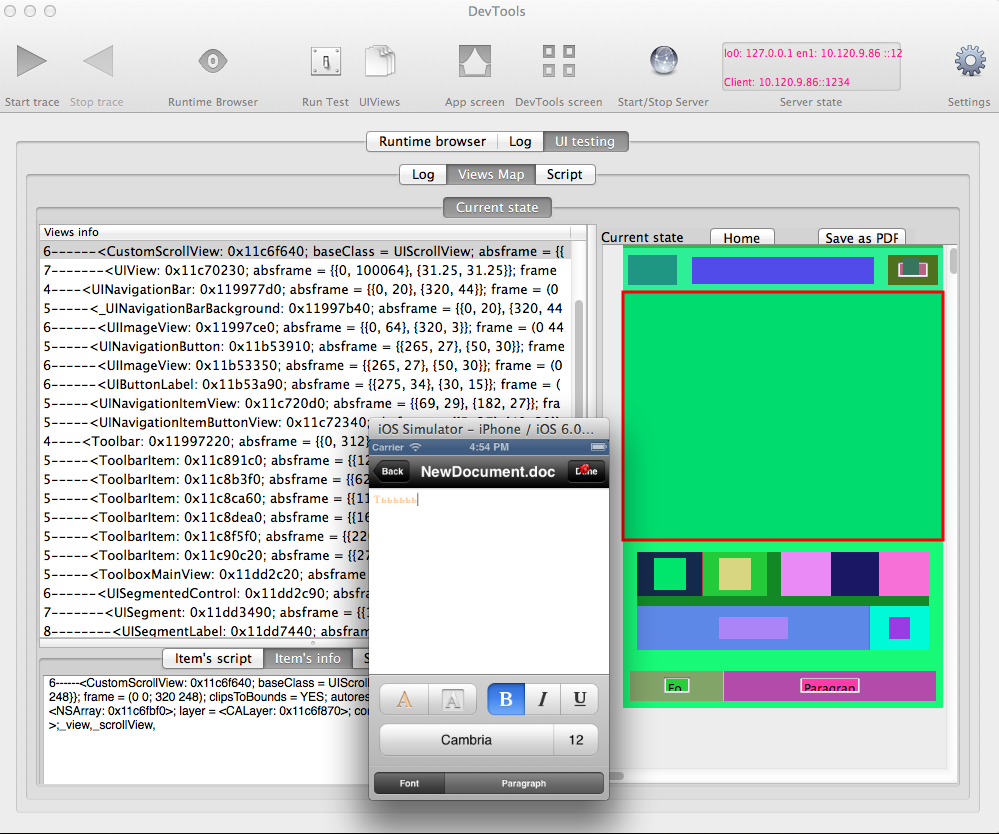
- **UI тестирование**

В основе проект с открытым исходным кодом UISpec. Позволяет писать UI-тесты, интегрированные в приложение, на языке Objectove-C или на специальном скриптовом языке (язык в стадии разработки, набор исполняемых команд документирован, исполнение команд реализовано. Имеется нативный скриптовый язык. Однако он не реализован до конца, поэтому не м.б. применен.). Интегрирован в клиентскую часть. Логи выводятся на сервер. Преимуществом данного подхода является практически неограниченные возможности в написании UI-тестов, возможность подключать к написанию UI-тестов разработчиков приложения.



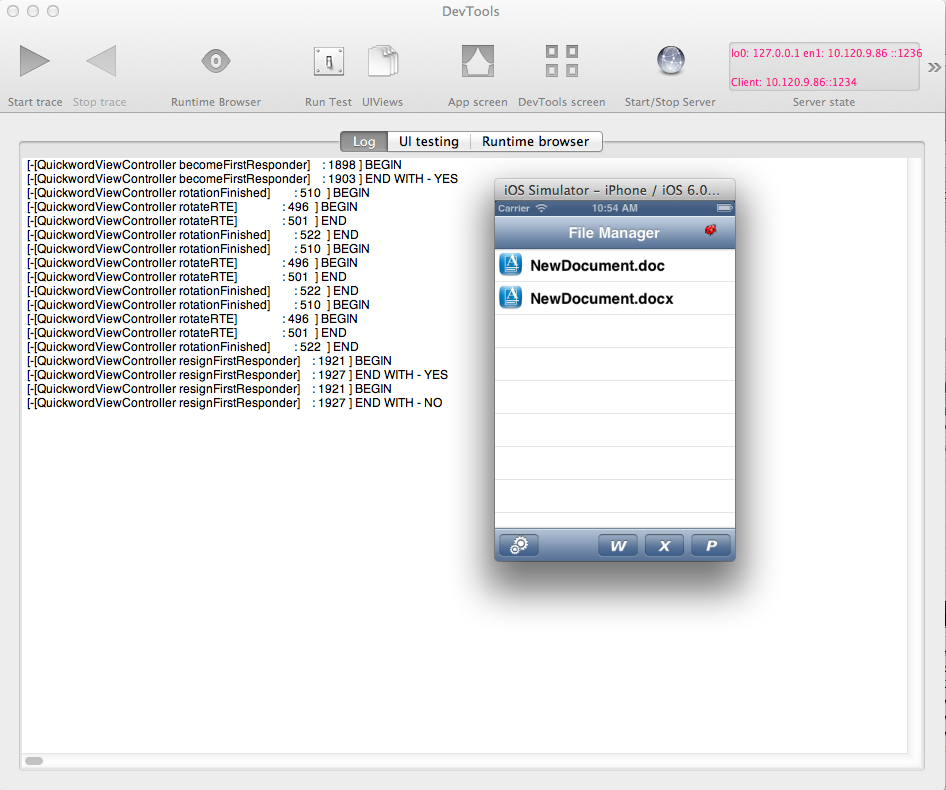
- **Иерархия UIView**

На стороне сервера выводится диаграмма View текущего состояния клиентского приложения. Сейчас диаграмма выводится перед выполнением каждого UI-теста. Так же есть возможность вывести диаграмму в любой момент по команде со стороны сервера. Графическая диаграмма позволяет перемещать элементы диаграммы вручную. Одновременно выводится дерево классов View, дополнительное и расширенное описание класса, что используется также для создания скриптов UI-тестировния. Дополнительно, выводятся строки команд (приложение анализирует свойства класса), которые можно запустить на выполнение. Можно сохранить графическую диаграмму в PDF-файл. Используется для разработки и отладки. Полезно для исследования иерархий View при разработке и исправления багов.



- **Отладка путем логирования**

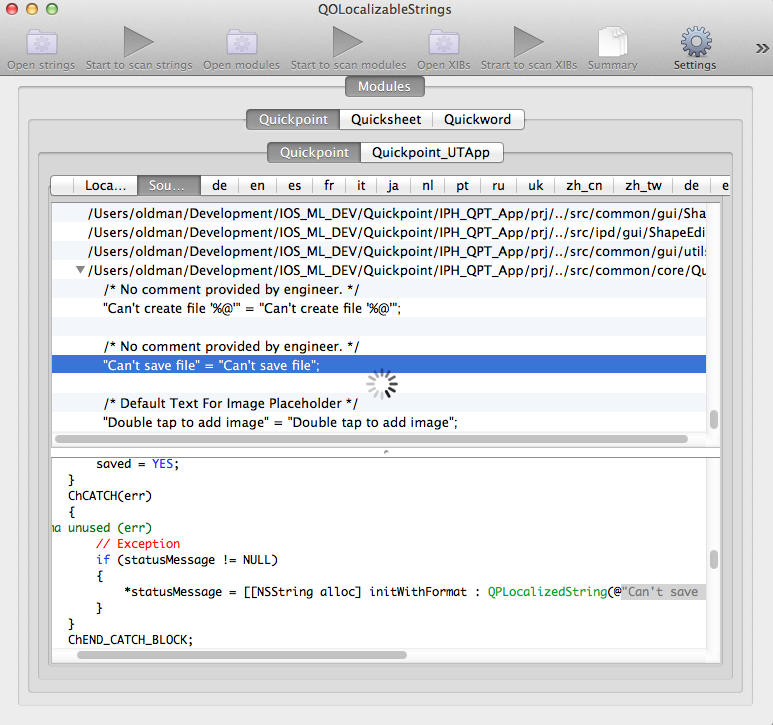
Логирование клиентского приложения. Cообщения передаются по сети, что исключает необходимость подключать i-девайс по USB-кабелю для отладки путем логирования. Полезно при отладке приложения, когда возникает необходимость отключить девайс от органайзера xCode. Например, при подключении и отключении от внешней клавиатуры.



- **Локализация приложения**.

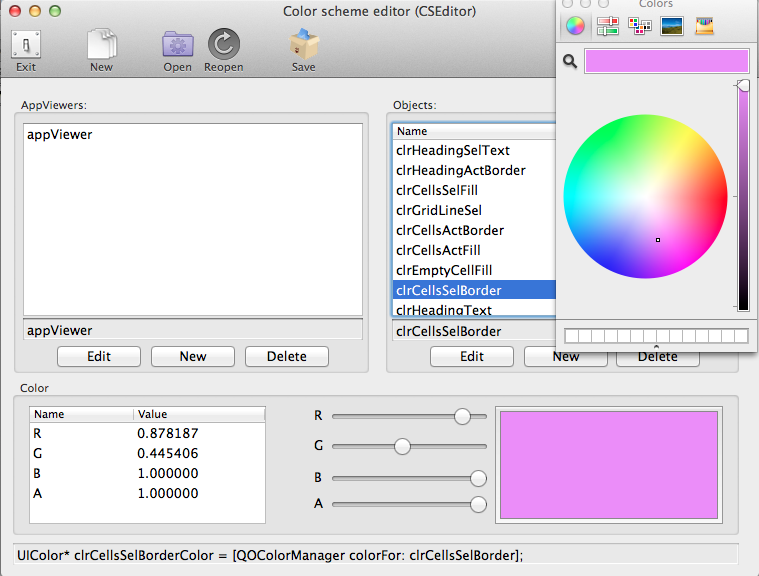
Настольное приложение. Создание строковых ресурсов для локализации одного проекта (в терминах приложения), или нескольких проектов суммарно. Есть автоматический перевод строк (Google translate). Поиск похожих строк по специальному критерию. Поиск строк в модулях и XIB-файлах. Прямой парсинг и обратный мерж в модули и XIB. Сканирование только полезных модулей. Приложение создает списки модулей, отсекает лишние мертвые программные модули. Списки модулей выводятся в виде дерева. Для выбранного элемента дерева выводится содержимое модуля с подсветкой синтаксиса и строк локализации. Приложение выводит полный действующий словарь строк (можно использовать при разработке, чтобы не создавать новые строки),

текущий словарь плюс строки, подлежащие переводу. Строки можно сохранять в текстовый файл, plist-файл с последующей загрузкой в базу из текстового файла или plist. Приложение хранит данные в локальной базе данных. Возможностей инструмента достаточно для всех видов работ по локализации. Использовался на протяжении нескольких релизов Quickoffice. Возможность формировать списки полезных модулей можно применить для чистки репозитория от мусора.



- **Цветовые темы**.

Настольное приложение. Позволяет интерактивно создать словарь цветов (plist - файл). При подключении словаря в качестве ресурса, можно легко менять цветовые схемы во время выполнения приложения. Инструмент могут использовать дизайнеры. Используются различные способы формирования цвета из компонент RGBA.

****

**Перспективы развития инструмента**

**- База знаний.**

Хранить классические примеры кода из учебников. Автоматическое копирование кода в буфер обмена. Варианты использования классов из iOS SDK. Готовые шаблоны языковых конструкций для использования непосредственно из DevTools или копирования в xCode. Согалашения по оформлению кода. Руководства по системам контроля версий и т.д.

**- Развитие диаграмм классов.**

Вывод информации для наследника. Что конкретно он наследует, что перегружено (overload).

Связать диаграммы классов с диаграммами View.

Дерево классов можно сортировать. Оставлять только пользовательские классы, скрыть классы SDK.

**- Дизайн.**

**-Settings**

**-Возможность подключаться нескольким клиентам одновременно.**

Для каждого клиентского приложения иметь свой набор закладок.

**- Состояния сервера**

Создать базу данных сервера. По инициативе клиента хранить его данные. Например, клиент передает ID приложения. Для него на сервере появляется запись, куда попадает лог. Хранить логи UI- тестирования с указанием временных меток. Таким образом можно документировать изменения в перформансе приложения на сервере в динамике.

**- Развитие UI тестирования**

развитие скриптового языка.

проблема в UI-тетсировании - идентификация элементов UI. Для ее решения можно попробовать развить UISpec в направлении создания категории UIView для добавления тега или символьного имени. Чтобы обращаться из теста к элементу по номеру тега или имени. Это значительно упростит написание тестов.

**- Развитие трассировки**

В перспективе возможно использовать для создания диаграмм состояния классов. Сейчас лог выводится в виде строк, что отражает последовательность выполнения. Предполагается структурировать вывод. Например, сводить все вызовы методов относительно классов. Можно выводить временные отсечки таймера. Можно использовать трассировщик для тестирования. Запустив трассировщик, можно сохранить лог (не весь, только контрольный точки). При повторном проходе сравнивать временные отсечки в контрольных точках.

Трассировка работает только на симуляторе. Исследовать, почему не работает на девайсе.

**Описание приложения**

DevTools состоит из серверной (DevTools) и клиентской части (DevToolsClient). В качестве серверной части выступает настольное MacOS приложение. В качестве клиентской - фреймворк, который может быть добавлен в зависимости проекта приложения. Фреймворк включает фреймворк, который обеспечивает обмен по локальной сети, UISpec-фреймворк для UI-тестирования. Клиент DevToolsClient имеет свой UI. Есть UI для переключения между окном приложения и клиентом DevTools.

UI клиента представляет собой табы рантайм-браузера, UITextView для вывода логов, настройка (settings) как серверной, так и клиентской части (сейчас используется для ввода сетевых параметров).

Как клиент, так и сервер содержат свой собственный сетевой клиент и сервер. Сервер получает команды и исполняет. Клиент выдает команды.

Подробное описание с инструкциями для разработчика в стадии разработки. В ресурсы проекта включен файл readme.rtf с инструкциями разработчику.

Кротких Сергей. 9.01.2013.