Ferry

Product Vision
Пользовательские требования

Составлено: Обучение в Otus, курс Разработчик C# 2021-04, команда YAT

Михаил Зюбин
Михаил Штребель
Валерий Шийко
Дмитрий Демин
Федор Пинега

18.08.2021

Содержание

Сведения о проекте и создаваемой системе

Данный проект реализуется в рамках курса “C# Developer Professional” командой YAT.

Создается сервис, который обеспечивает возможность интеграции между различными системами по аналогии с тем, как это делают Zapier/Integromat.

Рабочее название сервиса - Ferry.

Термины и определения

Термин	Значение
Сценарий	Программируемая в Ferry цепочка взаимосвязанных Шагов, которая обеспечивает интеграцию между различными системами. Синонимы: "Интеграция", "Воркфлоу"
Ferry	Создаваемый нами сервис. Синонимы: "Система"
Шаг сценария (Шаг, Step)	Объект, который принимает данные, может делать различные действия, и выдает данные как результат работы. Шаг сценария является Активным объектом (IActiveObject). Все Шаги сценария образуют цепочки действий, составляющие собственно сценарий. Все

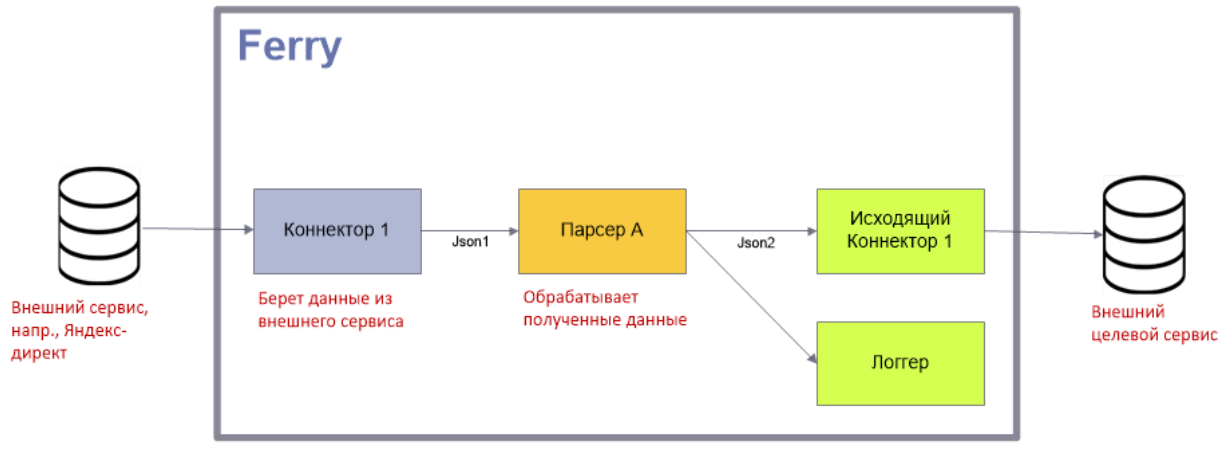
	шаги сценария связаны между собой через механизм передачи сообщений (IInterObjectMessage).
Активный объект	Объект, который принимает данные, может делать различные действия, и выдает данные как результат работы. Реализуется через интерфейс IActiveObject. Сценарий состоит из активных объектов, которые связаны между собой через через механизм передачи сообщений (IInterObjectMessage). Любые виды активных объектов - коннекторы, таймеры, логгеры, и др. - создаются на базе IActiveObject.
Личный кабинет, ЛК	Приватное пространство внутри Ferry, где Пользователь может работать со своими сценариями.

Основная идея

Сделать сервис, который позволяет осуществить интеграцию нескольких систем, по аналогии с тем, как это делают Zapier/Integromat, но этот наш сервис бесплатный и доступный в open-source, так что любой желающий может запустить его у себя на сервере и настроить как ему удобно.

Акцент делается именно на организацию взаимодействии между API различных сервисов, то есть Ferry опрашивает API различных систем (напр., Яндекс-директ, Яндекс-погода, курсы валют), берет оттуда json, xml или файл какого-либо формата, далее выполняет обработку этих данных, и отправляет их на другие сервисы (напр., БД нашей компании, где мы аггрегируем данные).

Обработка данных выполняется ступенчато, внутри цепочек действий, которые, например, могут выглядеть так:



Единицей работы Ferry является Сценарий. Сценарий -это последовательность блоков (которые называются Шагами сценария), обрабатывающих данные. Сценарий может быть запущен и остановлен. Создавая собственные блоки, использующие API внешних сервисов,

разработчик может обеспечивать интеграции с любыми доступными сервисами, у которых вообще есть API.

Целевая аудитория Ferry

Целевой аудиторией Ferry являются разработчики (программисты), перед которыми стоит задача бюджетно обеспечить бесперебойную интеграцию многих сервисов. В данный момент Ferry ориентирован на программистов, обладает простым графическим интерфейсом, и не позиционируется как Low-code решение.

Особенности Ferry (Features)

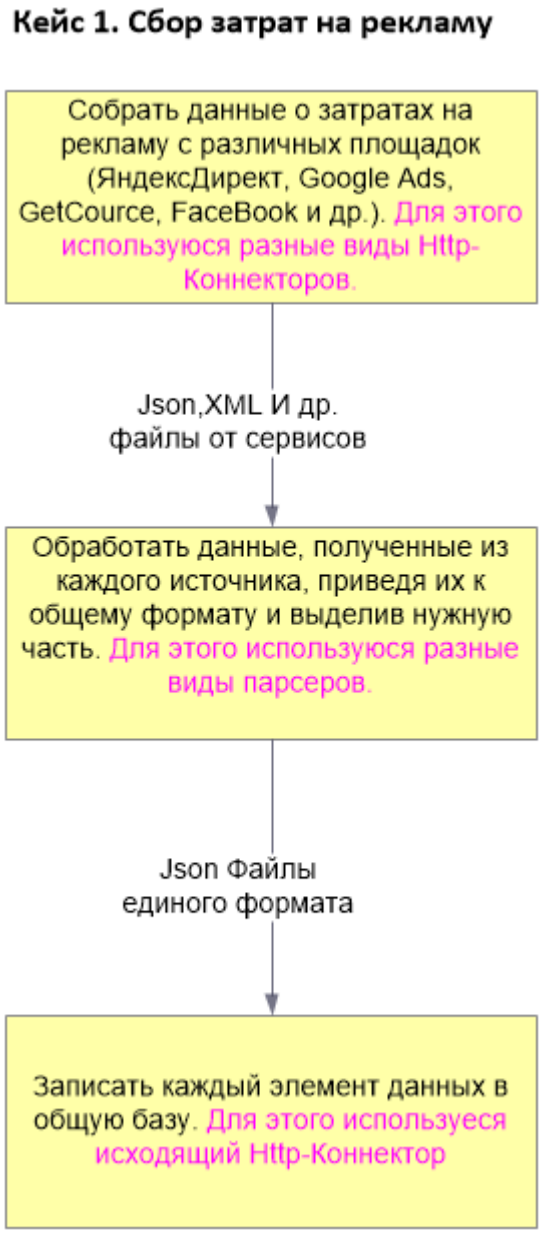
1. Ferry является платформой, которая используя архитектуру IActiveObject (см. Архитектурные особенности) позволяет быстро создавать и запускать в работу широкий диапазон интеграционных сценариев.

2. Уже сейчас на Ferry можно реализовать практически все, что может Integromat/zapier (кроме вычисления сложных выражений). Для этого только нужно написать соответствующие активные объекты.

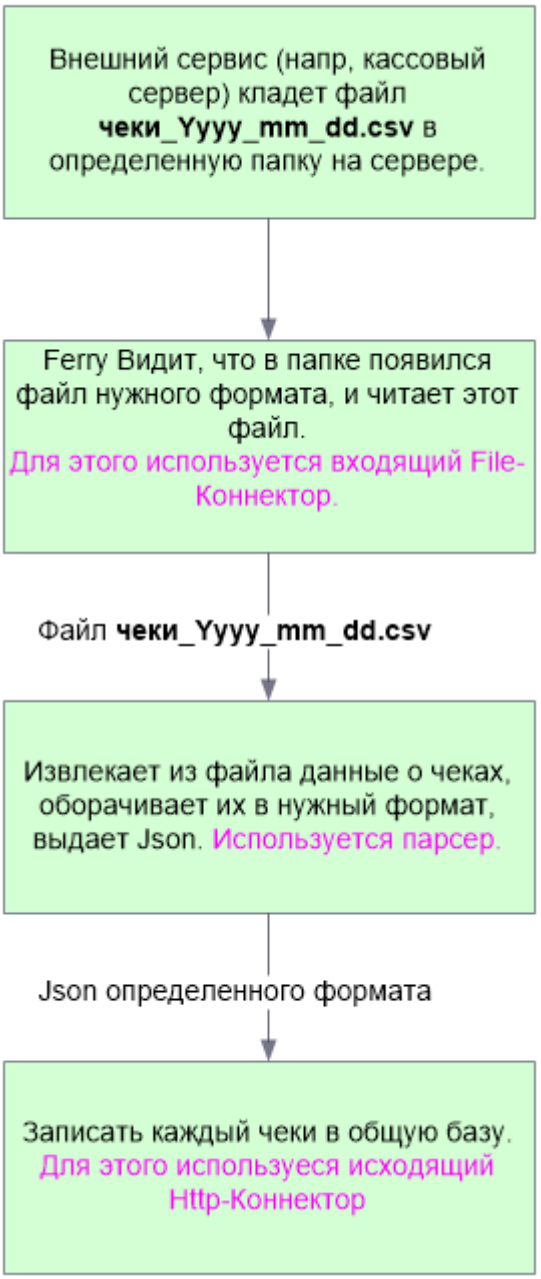
Ролевая модель Ferry

В данный момент используется унитарная ролевая модель, т.е. там всего 1 роль - Администратор.

Типовые сценарии использования (примеры)



Кейс 2. Аггрегация данных о чеках



Кейс 3. Нотификация (уведомление)

User Story

№	История	Сторипойнты/статус	Комментарий
A	Сервис целиком (логин, ЛК, перфоманс и др.)		
A1	Как Администратор **я хочу** иметь возможность самостоятельно создать аккаунт в Ferry, в который можно залогиниться, получить доступ к Личному кабинету (ЛК) внутри Ferry, работать там со сценариями, а также разлогинится **чтобы** работа с моими сценариями происходила внутри моего приватного пространства		
B	Сценарии		
B1	Как Администратор, находясь внутри своего ЛК **я хочу** создавать Сценарии интеграции между различными системами, а также управлять этими Сценариями (создать, удалить, редактировать) **чтобы** потом пользоваться ими.		
B2	Как Администратор, создав Сценарий, **я хочу** иметь возможность запускать и останавливать его, **чтобы** он выполнялся не все время, а только тогда, когда мне надо.		
B3	Как Администратор, создав Сценарий, **я хочу** иметь возможность создать его дубликат, **чтобы** не надо было вручную создавать заново аналогичный сценарий, если мне таковой потребуется.		
B4	Как Администратор, создав Сценарий, **я хочу** иметь возможность запускать его 1 раз (одно входящее действие или сообщение, т.е. режим RunOnce), **чтобы** иметь возможность быстро проверить его работу.		
B5	Как Администратор, во время выполнения сценария, **я хочу** иметь возможность смотреть его логи в реальном времени **чтобы** своевременно отлавливать ошибки и вносить изменения в		

	сценарий		
C	Работа и логика сценариев		
C1	Как юзер, редактируя Сценарий, **я хочу**, чтобы он имел несколько начальных точек и несколько конечных **чтобы** иметь возможность читать принимать данные из нескольких источников и передавать данные в несколько целевых сервисов.		
C2	Как юзер, во время работы Сценария, **я хочу** чтобы исполнение сценария могло переходить к тому или иному шагу в зависимости от некого условия (значения вычисляемой переменной) **чтобы** иметь возможность реализовать ветвления в сценариях.		
C3	Как юзер, редактируя Сценарий, **я хочу** чтобы Ferry проверял сценарии на корректность перед сохранением и перед запуском, **чтобы** потом не было ошибок при выполнении сценариев.		
D	Производительность, безопасность, требования к БД		
D1	Как Администратор **я хочу** иметь возможность быстрого и понятного развертывания Ferry на своем хостинге как веб-приложения/веб-сервиса **чтобы** решать стоящие передо мной вопросы интеграции систем.		
D2	Как Администратор **я хочу** иметь возможность выбирать, использовать мне реляционную БД, или нереляционную **чтобы** не покупать новый хостинг специально для Ferry		
D3	Как Администратор **я хочу** иметь возможность быстро написать драйвер под свою БД, если таковая не поддерживается Ferry **чтобы** не покупать новый хостинг специально для Ferry		Имеется в виду, что Ferry должно работать через БД через интерфейс, по паттерну

			Стратегия, чтобы можно было для каждой базы написать свой класс