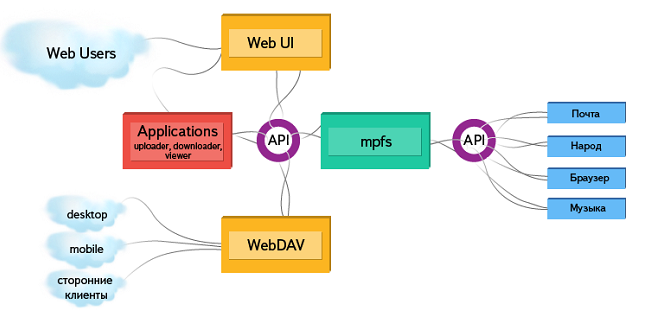
*1.3. Ознакомиться с использованием информационных сетей для решения задач структурного подразделения. Наличие локальных вычислительных сетей и задачи, решаемые с их помощью. Связь с глобальными сетями (Internet).*

Яндекс использует много разный технологий в своих сервисах, но сейчас мы рассмотрим такой сервис как Яндекс Диск.

Для начала рассмотрим архитектуру сервиса в целом. В основе сервиса лежит mpfs – Магическая проксирующая файловая система. Это бэкэнд, который содержит в себе всю бизнес-логику по работе с файлами, папками, каталогами: все операции по копированию или созданию новых файлов идут через эту систему. Эта же система отвечает и за хранение метаданных.

Синхронизация с клиентами происходит по протоколу WebDAV. На его основе мы создали клиенты под Windows, OS X и мобильные платформы. Но при необходимости с помощью WebDAV можно настроить синхронизацию с любым приложением, поддерживающим данный протокол.



Через веб-интерфейс пользователям доступны те же функции, что и через клиенты: загрузка, скачивание и просмотр файлов. Кроме того, через наш внутренний API производится взаимодействие с другими сервисами Яндекса. На данный момент это Народ, Почта, Музыка и Браузер. Например, через мобильный клиент Яндекс.Музыки можно получить доступ к загруженным в Диск аудиофайлам.

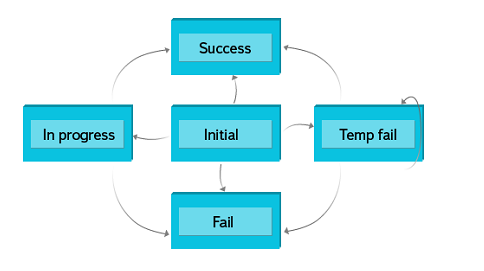
**Стадии задач**

Каждая задача, выполняемая загрузчиком делится на несколько стадий. Первым делом после получения файла от пользователя или сервиса производится подсчет контрольных сумм. В дальнейшем они будут использоваться при синхронизации файла для отслеживания изменений. Так как наш Кладун поддерживает загрузку изменений патчами, каждый файл разбивается на несколько блоков, для каждого из которых генерируется отдельная контрольная сумма. Если файл после заливки был слегка отредактирован, то при синхронизации перезалит он будет не целиком: заново будут загружены только те блоки, в которых произошли изменения и контрольные суммы перестали совпадать. Нечто подобное используется в системах управления версиями.

Использование хэш-сумм также позволяет избежать повторной загрузки уже залитых другими пользователями файлов. К примеру, если вы хотите залить на Диск какой-либо популярный видеоролик или установочный файл, а в хранилище Диска уже есть файл с идентичной контрольной суммой, использоваться будет он. Таким образом, даже очень крупные файлы могут быть помещены в хранилище буквально за пару секунд. Это выгодно обеим сторонам: пользователь получает очень быструю загрузку, а мы экономим ресурсы. Всего повторяющиеся файлы составляют около 12 процентов от всех заливаемых.

После заливки файла на машину загрузчика и подсчета хэш-суммы необходимо переместить файл в хранилище и отправить статус загрузки в mpfs. С этого момента файлы становятся доступны пользователю: они отображаются в веб-интерфейсе и могут синхронизироваться при помощи клиентов. Но на этом работа загрузчика не заканчивается, наступает черед двух стадий пост-обработки: создание превью и проверка на вирусы.

Каждая стадия проходит через определенный набор состояний. Изначально все они находятся в исходном состоянии (initial). Некоторым стадиям на выполнение требуется совсем немного времени, они практически сразу переходят в статус успешно завершенных (success). Но на пути к успеху могут случаться сбои. Они бывают двух видов: временные и окончательные. Например, когда при передаче данных в mpfs случается сбой сети или перестает отвечать один из дата-центров, стадия переходит в состояние временного сбоя (temp fail) и через какое-то время запрос повторяется. Если через определенное количество повторений успешно завершить стадию так и не удается, она переходит в состояние окончательного сбоя (fail).



В случае сбоя на стадии, которая никак не связана с сетью и выполняется локально, она сразу переходит в статус fail, так как повторный ее запуск, скорее всего, никаких результатов не даст. Именно так обстоит дело с генерацией превью. Это достаточно простой и понятный процесс, и если что-то пошло не так, то велика вероятность того, что мы просто пока не научились делать превью для этого типа файлов, а значит, повторение этой стадии – дело бессмысленное. Необходимость повтора операции, а также количество повторов, после которого она окончательно переходит в состояние fail прописывается для каждой стадии отдельно. Приведет ли сбой при выполнении той или иной стадии к провалу всей задачи, зависит от параметра обязательности успешного выполнения. Например, как уже говорилось выше, генерация превью далеко не всегда завершается успешно. Но для пользователя наличие превью в веб-интерфейсе не критично, главное, что доступ к своим файлам получен, а значит, основная задача выполнена. Однако если сбой произойдет на более важной стадии, такой как передача файла в хранилище, провалена будет вся задача. В этом случае десктопный клиент повторит попытку сам, а при заливке через веб пользователю будет передано сообщение о неудаче, после чего он может попробовать загрузить файл заново.

Кроме того, у каждой стадии есть максимальное время выполнения. Если стадии для успешного завершения требуется много времени, она может переходить в состояние «в процессе» (in progress). Обычно это происходит при получении файла от пользователя, так как загружаемые файлы бывают достаточно большими (максимальный объем файла на Диске – 10 ГБ), а скорость подключения – низкой.