Décisions finales

Table des matières

[Architecture 1](#_Toc528308674)

[Interfaces 1](#_Toc528308675)

[Langages / Protocoles 1](#_Toc528308676)

[Autres 1](#_Toc528308677)

# A Penser

Quelles interfaces pour faire communiquer les pairs ? (Interface == langage de communication)

Structure des messages

Protocoles (IP, TCP, UDP, HTTP, FTP, SMTP)

Découverte de pair(s)

Fonctionnalités : CRUD (create, read, update, delete)

Langage : JAVA

Format : JSON, XML

Fonctions :

* Connect
* Disconnect
* Register
* Unregister
* ListPeer
* GetFile(s)
* GetFileList
* SetFile(s)
* DeleteFile

# Résultat

HTTP(S), c’est mieux, c‘est moins de boulot, c’est plus facile, et ça fait le café.

CRUD :

Create file pair POST

Read file fileList pairList GET

Update file UPDATE

Delete file pair DELETE

File : id, contenu / nom / taille, pairs

Pair : id, ip, port

REST

## Service

* Interface Abstraite
* Interface Concrete comment ? JSON, HTTP

REST

FICHIER (données)

FICHIER (métadonnées)

* fileId string
* size int64
* name string

PAIR

* url string

## Ressources

<pair>/files get  M

upload(M)

<pair>/files/<fileId> get  D

delete(FileId)

upload(D)

<pair>/peers register

list

<pair>/peers/<peerId> unregister

## Choix

REST, JSON, http

## Diagramme

Diagramme de séquence (exemple)

Machine à états (protocole)

Pré-post conditions

Requêtes :

Register

Message d’erreur

Liste des erreurs http potentielles (code erreur)

Faire un test par type de requête

Projet :

Client

Serveur

Interface graphique

Interface P2P

Sauvegarde pairs

Sauvegarde fichiers

Déploiement : installation

Fichier paramétrage

class File

class Peer

# Architecture

# Interfaces

# Langages / Protocoles

# Autres