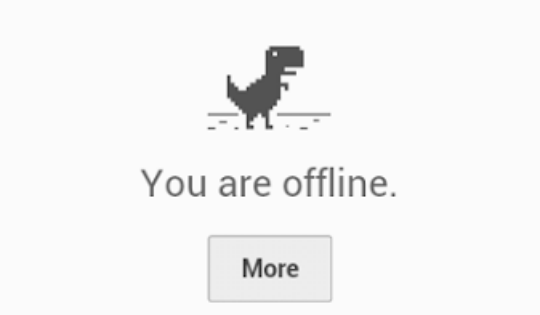
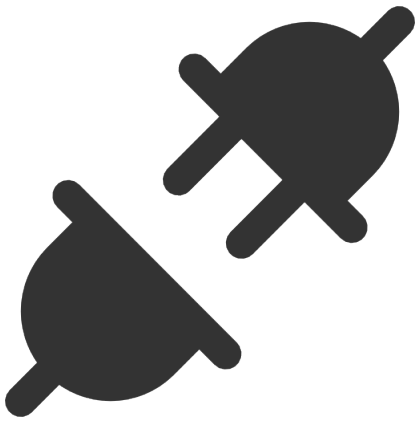


Conception et Architecture des Systèmes d'Information

Monographie

CASIFini - Le Web Offline



Anthony Courtin
Alexandre Hottin

A l'attention de : Frederic Baucher

Septembre 2015 — Janvier 2016

Table des matières

Chapitre 1

Question d'amorce

1.1 Réplication asynchrone de données

La réplication est un processus de partage d'informations pour assurer la cohérence de données entre plusieurs sources de données redondantes, pour améliorer la fiabilité, la tolérance aux pannes, ou la disponibilité. On parle de réplication de données si les mêmes données sont dupliquées sur plusieurs périphériques.

La réplication n'est pas à confondre avec une sauvegarde : les données sauvegardées ne changent pas dans le temps, reflétant un état fixe des données, tandis que les données répliquées évoluent sans cesse à mesure que les données sources changent.

1.2 Concepts HTML5

1.2.1 Application Cache

Application cache met l'application en cache et permet à l'application d'être disponible sans connexion à internet. L'utilisateur peut donc utiliser l'application en hors ligne. Il permet d'accélérer la vitesse de l'utilisation de l'application car elle est en cache et de réduire le volume de données à télécharger auprès du serveur.

1.2.2 Offline Storage

Offline Storage permet de sauvegarder les données et de faire fonctionner les applications ou les jeux sans être connectés à internet.

1.3 API HTML5

1.3.1 Web Storage

Les applications web peuvent être stockées en local dans le navigateur internet de l'utilisateur. Plus performant que les cookies.

1.3.2 Web SQL Database

Enregistrement des données dans une base de données pouvant être interrogée en SQL.

1.3.3 Indexed Database

La base de données contient des valeurs et des objets hiérarchisés. Un enregistrement se compose d'une clé et d'une valeur. Un index est implémenté pour optimiser l'accès aux données.

1.3.4 File Access

Représentation du fichier en objet dans l'application web. L'accès aux données se fait par les objets correspondants.

1.4 Bases de données

Les principaux types de bases de données offrant des fonctionnalités de réplication dans un contexte Web sont les suivantes :

- MySQL ; ;
- PostgreSQL

Chapitre 2

Partie A

2.1 A0. Introduction

Le sujet du projet est : Application Web Offline.
Parmi les problématiques que l'on aura à traiter on trouvera : la tolérance aux pertes partielles ou totales de connexion, la synchronisation de référentiels de données, la gestion des conflits, le stockage des données, authentification, gestion des données.

2.2 A1. Mots-clés

- HTML5 ;
- Application Cache ;
- Offline Storage ;
- Mobile ;
- Indexed Database & Web SQL Database ;
- Réplication asynchrone de données ;
- Application Web Offline ;
- Manifest File ;
- Synchronisation ;
- Complexité.

2.3 A2. Webographie

HTML5-CSS3, `http://www.html5-css3.fr/html5/tutoriel-application-web-offline-html5-cache`

Blog Nouvelles Technologies, `http://www.blog-nouvelles-technologies.fr/4116/comment-faire-p`

W3 Schools, `http://www.w3schools.com/html/html5_app_cache.asp`

Mozilla, `https://developer.mozilla.org/fr/Apps/Build/Hors-ligne`

IBM, `http://www.ibm.com/developerworks/xml/library/x-html5mobile3/index.html`

2.4 A3. Bibliographie

Rodolphe Rimelé, *HTML 5 - Une référence pour le développeur web*, Eyrolles, 7 mars 2013, 727 pages.
- Chapitre 15 - Stockage des données locales (Web Storage) ;
- Chapitre 16 - Bases de données (Indexed Database & Web SQL Database) ;
- Chapitre 17 - Applications web hors ligne.

Jean-Pierre Vincent et Jonathan Verrecchia, *HTML5 - De la page web à l'application web*, Dunod, 6 juillet 2011, 256 pages.
- Chapitre 10 - Application Web Offline.

Peter Lubbers, *Pro HTML5 Programming : Powerful APIs for Richer Internet Application Development*, APress, 1 septembre 2010, 304 pages.
- Chapitre 9 - Using the HMTL5 Web Storage API
- Chapitre 10 - Creating HTML5 Offline Web Application

2.5 A4. Organisations

2.5.1 Tata consultancy services

Tata consultancy services (TCS) est une filiale du groupe Indien Tata et est classée parmis les plus importantes sociétés de services en informatique au monde. Son modèle de livraison de réseau mondial, Global Network Delivery Model, permet de répondre aux besoins de ses clients le plus rapidement possible. L'activité de TCS va du conseil jusqu'à la mise en oeuvre et au suivi des systèmes. De plus, il est possible pour les clients d'externaliser le processus métier ou la mise en place de services d'ingénierie. En ce qui concerne la participation de TCS dans le domaine du Web Offline, elle a publié un Livre blanc disponible à cette adresse : `http://www.tcs.com/SiteCollectionDocuments/White%20Papers/TEG_Whitepaper_Developing_Offline_Web_Application_Using_HTML5_0212-1.pdf`

2.5.2 W3C

Le World Wide Web Consortium est une organisation à but non lucratif s'occupant de la standardisation dans le domaine de l'informatique. Elle fut fondée en 1994 dans le but de promouvoir les technologies du Web. La gestion du W3C est assurée par le MIT, l'ERCIM, l'université Keio et l'université Beihang. Le W3C est à l'origine de nombreux standads dans le domaine du Web. On peut notamment citer XHTML, SVG, SPARQL et HTML. En ce qui concerne la participation du W3C dans le domaine du Web offline, cette dernière a publier une note de groupe de travail disponible à l'adresse suivante : `http://www.w3.org/TR/offline-webapps/`

2.5.3 Mozilla Foundation

La Mozilla Foundation est une organisation à but non lucratif créé le 15 juillet 2003 et situé à Mountain View en Californie. Son but est de développer et de publier des logiciels libre de droit. La

fondation a créé un site web appelé Mozilla Developer Network (MDN) qui regroupe de la documentation sur les technologies Web telles que HTML, CSS et Javascript. En ce qui concerne la participation de la Mozilla Foundation dans le domaine du Web offline, cette dernière a publier une documentation disponible à l'adresse suivante : `https://developer.mozilla.org/fr/Apps/Build/Hors-ligne`

2.6 A5. Facteurs

- Functionality $->$ Interoperability (1)
- Portability $->$ Adaptability (2)
- Reliability $->$ Recoverability (3)

2.7 A6. Indicateurs

- Operation Time
- Number of Functions
- Data exchangeability (data format-based) (1)
- System software environmental adaptability (adaptability to OS, network software and cooperated application software) (2)
- Restoration effictiveness (3)

2.8 A7. Références théoriques

2.8.1 Load Everything pattern

- L'application charge toutes les données dont l'utilisateur a besoin d'utiliser directement
- L'utilisateur utilise les données stockées localement
- Le processus de chargement peut être très long suivant la connexion de l'utilisateur et la quantité de données à télécharger

2.8.2 Caching pattern

- L'application requête les données au serveur lorsque l'utilisateur en fait la demande et les stocke au cas où l'utilisateur en aurait encore besoin
- On peut citer les applications de messagerie qui stockent un certain nombre de mail, les applications de streaming musical ou bien celles utilisant une
- carte
- Il faut que l'application garde le cache à jour afin que l'utilisateur ne soit pas en retard par rapport à ce qu'il y a sur le serveur

2.8.3 Selectiveness pattern

- Ce pattern fonctionne comme le second à l'exception que l'utilisateur a le choix de ce qu'il souhaite mettre en cache

Chapitre 3

Partie B

3.1 B1. Approches techniques

L'intérêt du web offline est de permettre à un utilisateur d'utiliser ses applications habituellement connectées à internet (boite mail, streaming musical ...) ou de consulter ses pages web préférées (casisbelli ...), le tout, en étant hors-ligne. Cette technologie s'est principalement développée grâce à l'augmentation du nombre de smartphones et de tablettes.

Afin de pouvoir utiliser un site web ou une application hors ligne, nous devons connaître l'état de connexion de l'utilisateur, c'est-à-dire, savoir s'il est connecté à internet ou non *(pattern Message Endpoint)*. Cette étape est généralement réalisée à l'aide de fonctions de type javascripts en utilisant notamment : navigator.onLine. Une fois que nous avons pris connaissance de l'état de connexion de l'utilisateur, nous devons spécifier quelles sont et où sont situés les informations nécessaires afin d'afficher correctement le site internet ou le contenu de l'application. Cette étape passe généralement par la création d'un fichier appelé Cache Manifest et enregistré généralement sous le nom offline.manifest. Voici un exemple de fichier :

CACHE MANIFEST
angular.min.js
bootstrap.min.js
styles-perso.css
jquery-1.4.min.js
offline.js
index.html
exercices.html
contact.html

Il faut ensuite lier ce Cache Manifest à nos pages html de la manière suivante : $< html manifest = "offline.manifest" >$

Il est important de noter que les exemples donnés ci-dessus et par la suite de cette partie reposent sur l'utilisation de la technologie Application Cache. Il existe une autre technologie nommée Service Worker.

Ensuite, il est important de stocker les données localement afin d'avoir accès aux données et de pouvoir utiliser correctement le site web ou l'application *(pattern Message Store)*.

Pendant l'utilisation, il est nécessaire de veiller à enregistrer les données régulièrement dans la base de données locale au risque de perdre des données dans le cas contraire. La solution la plus simple à mettre en place est la sauvegarde du DOM des pages web à l'aides des objets suivants :

- window.sessionStorage pour sauvegarder pendant la période de Session
- window.localStorage pour sauvegarder pour une période plus importante

Dans tous les cas, ces objets offrent le même type de fonction permettant d'accéder aux données, de sauvegarder ces dernières ou de les supprimer facilement.

Une fois que la connexion à internet est rétablie, les données stockées en locale doivent être synchronisées avec le serveur distant *(pattern Messaging) (pattern Message Endpoint) (pattern Message Filter)*. Il faut donc prévoir de créer une queue des données pour être cohérent avec l'évolution des données en mode hors ligne *(pattern Resequencer)*. La transaction des données doit être vérifiée et réussie pour pouvoir passer à la transaction suivante *(pattern Guaranteed Delivery)*. Si la connexion à internet coupe pendant une transaction, c'est comme ci elle n'avait pas été réalisée et doit donc être refaite lors d'une nouvelle connexion à internet. De plus, si des modifications ont été réalisées sur le serveur distant entre deux connexions, ces dernières vont être téléchargées vers le dossier de stockage local *(pattern Shared Database)*.

La figure 3.1 explique de manière schématique les différentes grandes étapes de fonctionnement d'une application en web offline.

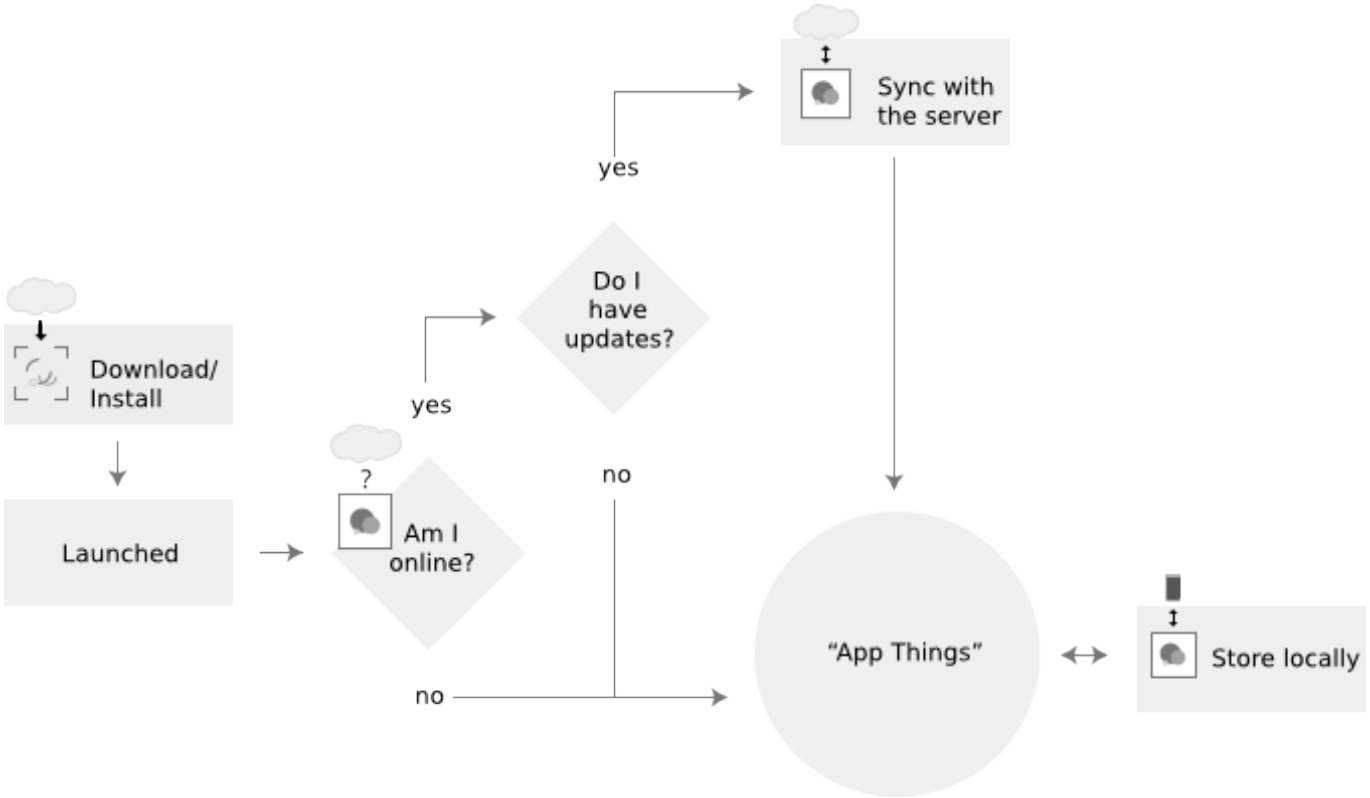


Figure 3.1 – Schéma explicatif

3.2 B2. Solutions technologiques

Nous n'avons trouvé que deux solutions :
- Application Cache ;
- Service Worker.

Ainsi nous utiliserons ces deux solutions technologiques afin de réaliser les prototypes.

3.2.1 Application Cache

L'utilisation d'Application Cache permet à l'utilisateur de naviguer hors-ligne, il peut donc utiliser un site même s'il n'est pas connecté à internet. Les ressources sont mises en cache en locale et se chargent donc plus rapidement. Cela permet également de réduire la charge du serveur. Application Cache est directement intégré à HTML5, il sera donc plus facile à implémenter.

3.2.2 Service Worker

Service Worker est basé sur un script Javascript qui s'exécute en tâche de fond depuis une page internet. Il permet de contrôler la gestion des requêtes depuis le réseau. Il ne nécessite pas l'intervention d'un utilisateur pour s'exécuter.

3.3 B3. Description technique

Détecter l'état de connexion à internet :
- offline.js
- failsafe.js

Stockage des informations nécessaire à l'affichage :
- manifest

Stockage des données :
- localForage
- localStorage

3.4 B4. Benchmark

- Functionality $->$ Interoperability
  - Data exchangeability (data format-based)
- Portability $->$ Adaptability
  - System software environmental adaptability (adaptability to OS, network software and cooperated application software)
- Reliability $->$ Recoverability
  - Restoration effictiveness

– Operation Time
– Number of Functions

3.5 B5. Work Breakdown Structure

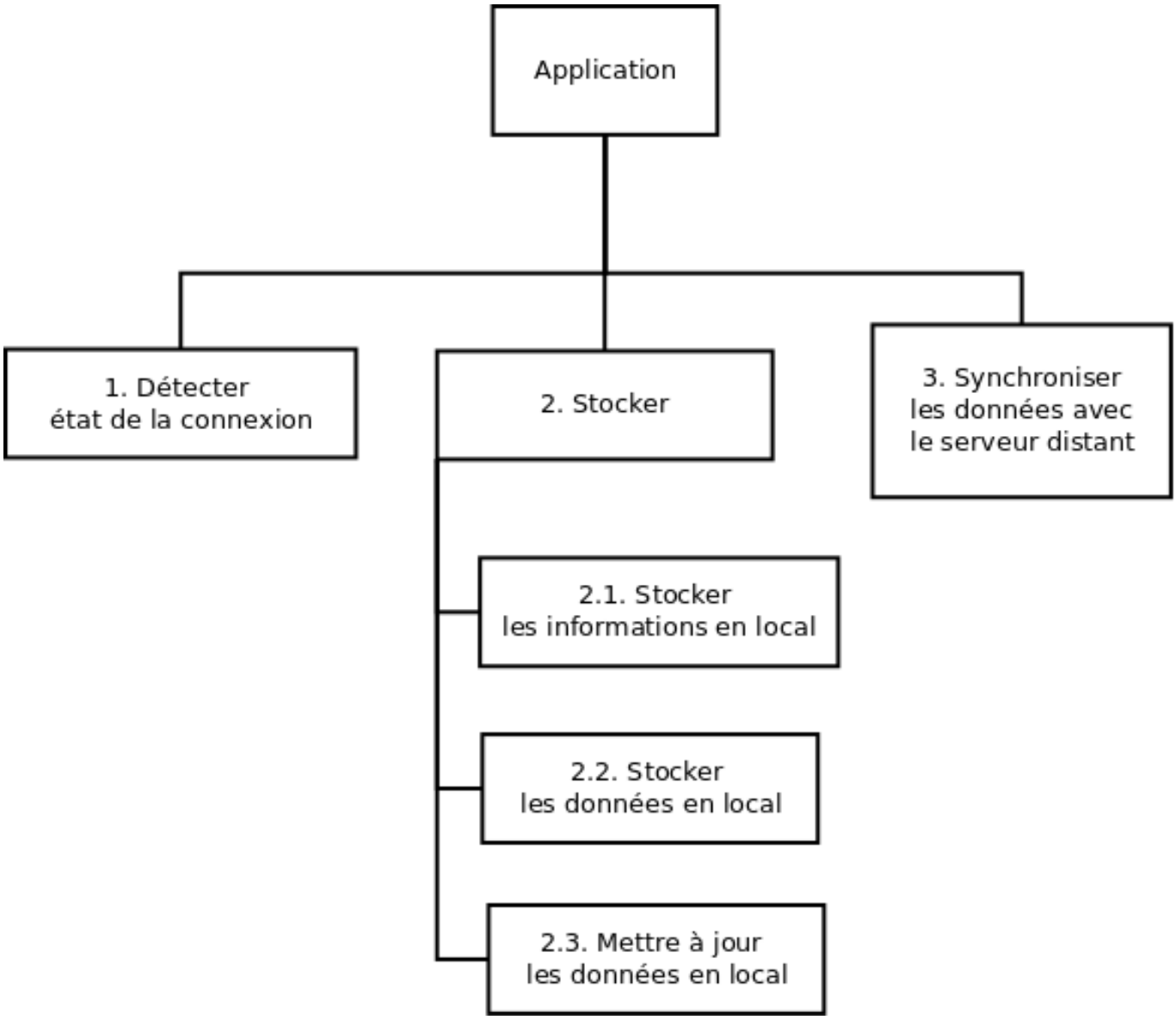


Figure 3.2 – Work Breakdown Structure

Table des figures