DEVELOPPEMENT ET DEPLOIEMENT D'UNE LIGNE DE PRODUITS DE SYSTEME DE DIFFUSION D'INFORMATIONS EN INSTITUTS SPECIALISES

Rapport de stage

Tuteur entreprise :
Philippe Collet
Philippe.COLLET@unice.fr

Tuteur Miage :
Michel Buffa
micbuffa@gmail.com

Golfieri Guillaume
Licence 3 Miage
Université de Nice Sophia Antipolis
2012-2013

miage nice

Université Nice Sophia Antipolis

cnrs
dépasser les frontières

Table des matières

Remerciements

Je tiens à remercier dans un premier temps mon tuteur de stage en entreprise M. Philippe Collet, enseignant chercheur à l'i3s, de m'avoir permis de faire mon stage au laboratoire i3s et de l'aide qu'il a pu m'apporter. Je remercie également M. Simon Urli doctorant travaillant sur le projet Yourcast, de son aide précieuse et de son encadrement tout au long de mon stage.

Je remercie également Mme Mireille Blay-Fornarino enseignant chercheur à l'i3s, pour son aide et sa gentillesse, M. Sebastien Mosser enseignant chercheur à l'i3s, pour toutes les informations qu'il m'a donné tout au long du stage et M. Christian Brel pour ses informations sur le doctorat et le système de composition.

Je voulais aussi remercier toute l'équipe de Modalis pour leur gentillesse et la bonne humeur affichée pendant la totalité de mon stage.

Enfin, je remercie M. Michel Buffa qui a été mon responsable MIAGE pour ce stage en entreprise.

Résumé

Pendant trois mois de stage au laboratoire i3s, j'ai assuré la fonction de développeur informatique. Mon principal objectif était de créer un client Yourcast pour l'institut spécialisé Clément Ader et de corriger celui de l'Irsam.

Il existait déjà des clients fonctionnel à mon arrivé donc j'ai pu m'aider de ces clients pour comprendre le fonctionnement d'un client et pouvoir en créer un par la suite. J'ai dû apprendre à utiliser la librairie Prototype et la librairie Less, les librairies principales du client.

Dans un deuxième j'ai dû adapter les modifications réalisées sur ce nouveau client dans la génération pour pouvoir faire profiter tous les clients des avancées. Et notamment la modification des comportements en classe qui m'ont obligé à modifier tous les clients existants pour adapter les anciens comportements aux nouveaux.

Dans un troisième temps, je me suis concentré sur la stabilisation du client et la résolution des bugs en vue du festival les Choralies qui se déroulait du 1er août au 9 août à Vaison-la-Romaine où Yourcast était le principal diffuseur d'informations.

À la fin du stage, j'ai réussi à réaliser tous les objectifs du stage. Malheureusement je n'ai pas pu commencer la dernière partie du stage qui était la réalisation d'une interface pour la composition des comportements.

Abstract

For three months internship in the i3s lab, I was a computing developer. My main goal was to create a new client of Yourcast for specialized institute named Clément Ader and to fix the client of Irsam.

There were already functional clients when I arrived arrived so I could help from those to understand how work a client and be able to create or fix one later. I had to learn to use the Prototype library and Less library, the main libraries of the client.

In a second time I had to adapt the changes made in this new client in the generator. In this way these changes could be integrate to others client. In particular the changes about the behavior of the zone. Because of this change, I had to change all behaviors from all others client.

In a third time, I focused on stabilizing the client and bug fixes for the Choralies festival which took place from August 1 to August 9 in Vaison-la-Romaine where Yourcast was the main distributor information.

At the end of the course, I managed to achieve all the objectives. Unfortunately I was not able to start the last part which was the creation of an interface for behaviors' composition.

Présentation de l'entreprise

Dans cette partie je vais présenter rapidement le Centre National de Recherche Scientifique. Le CNRS est un organisme de recherche publique spécialisé dans le scientifique (biologie, mathématiques, etc.) et la technologie (ingénierie et systèmes, science de l'information, etc.). Le CNRS compte plus de 11 000 chercheurs.

Plus précisément je me trouvais au laboratoire i3s spécialisé dans les domaines de l'informatique, des signaux et des systèmes se trouvant à Sophia Antipolis (http://www.i3s.unice.fr/I3S/index.php). Ce laboratoire est constitué de quatre pôles : GLC (Génie du Logiciel et de la Connaissance), COMRED (COMmunications, Réseaux, systèmes Embarqués et Distribués, MDSC (Modèles Discrets pour les Systèmes Complexes) et SIS (Signal, Images et Systèmes). Je me trouvais dans GLC (http://www.i3s.unice.fr/I3S/labos/labo2.html).

Il y a quatre équipes dans ce pôle et je me trouvais dans Modalis. Modalis a pour but de développer des outils pour exploiter un très grand nombre de produits. La recherche de ces outils intègre aussi le design de tels produits. Pour ma part je me trouvais dans le projet Yourcast qui est un système de diffuseur d'informations en continues. Le produit livrable est entièrement généré.

Introduction

Dans le cadre de ma licence 3 Miage à l'Université de Nice Sophia Antipolis, j'ai eu l'opportunité de réaliser un stage au sein du laboratoire i3s appartenant au CNRS. Passionné depuis longtemps par la recherche, ce stage était pour moi l'occasion de découvrir réellement ce milieu. En effet, il est plutôt difficile de récolter des informations sur cette activité tant les chercheurs sont occupés et que les journaux spécialisés sont en anglais et ne sont pas facilement accessibles.

L'objectif principal de ce stage était de m'occuper des clients liés aux clients Yourcast déployés dans les instituts personnalisés. Pour l'institut Clément Ader, il possédait une version ancienne de Yourcast que je devais recréer avec la nouvelle version. Pour l'Irsam, le client était fini mais des bugs résidaient. Enfin si j'avais le temps je devais mettre de nouvelles fonctionnalités comme la gestion de la vidéo ou la gestion de l'ordre dans les informations affichées.

Dans un premier temps, je vous présenterai le projet Yourcast et mon stage, puis comment il s'est déroulé pendant ces trois mois et enfin ma contribution au projet. Dans une dernière partie je ferai un bilan pour l'entreprise et pour moi.

I. Contexte

A. Présentation de Yourcast

1. La ligne de produit

Dans cette première partie, je vais présenter le système Yourcast. Tout d'abord, il faut savoir que Yourcast est un système de diffusion d'informations continues provenant de différentes sources. Ces sources peuvent être internes comme des annonces, des menus (cantine, restaurant) ou bien externes comme les tweets, les photos Picasa. Il faut savoir que Yourcast est aussi un sujet de recherche sur les lignes de produits. Dans le schéma ci-dessous, vous avez une présentation détaillée du système Yourcast. Vous pouvez voir qu'est-ce que contient la ligne de produits exactement et comment est réalisé l'affichage final de Yourcast.

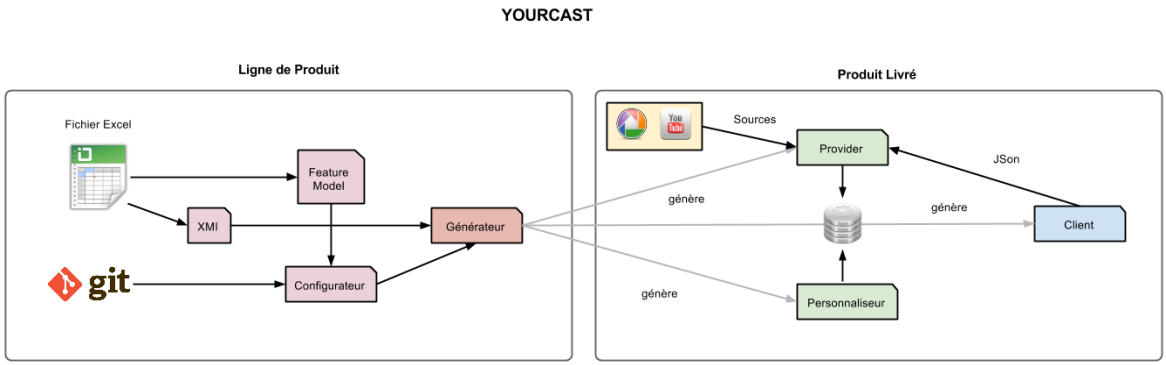


Figure 1 - Fonctionnement de Yourcast

On part tout d'abord d'un fichier Excel qui détaille tout ce qu'on doit avoir dans le générateur qu'on appelle Feature Model. C'est une sorte de bases de données des fonctionnalités réalisables dans le client. On retrouve ainsi dans ce tableau les renderers, les comportements, les layouts, etc. Pour faire un travail d'équipe efficace, on utilise un substitut à subversion (pas vu en cours) : GIT. La principale différence avec subversion est que lorsqu'on commit, celui-ci est local (donc pas sur le serveur principal). Pour que les modifications soient mises sur le serveur, il faut faire un push de tous les commits locaux.

Dans ce GIT, on retrouve les codes associés à ces fonctionnalités. Pour traduire le fichier Excel, on utilise un configurateur qui va adapter le générateur aux fonctionnalités proposées. Le générateur est le dernier élément de la ligne de produits car lorsqu'il a généré ses codes, il n'y a plus qu'à les mettre sur un serveur pour qu'ils soient opérationnels.

2. Le client final

On va maintenant détailler le client qui est le produit livré par le générateur. Le client est écrit en Javascript et est basé sur la bibliothèque Prototype qui est un système équivalant à JQuery. L'énorme problème par défaut de Javascript est qu'il n'est pas orienté Objet comme on le pense c'est-à-dire avec des classes, de l'héritage, etc. En effet, pour faire des classes en Javascript, on utilise des fonctions. Pour pallier ce problème, on utilise Prototype qui reprend les standards de Java, C++ et autres langages orientés objet.

Le client est découpé en zones d'informations qui constituent un layout (un style). Ces zones récupèrent les informations d'un fichier Json (Json est un format de fichier ressemblant fortement à un tableau de données) qui sont extraites et interprétées à l'aide des renderers. Ces informations sont affichées selon un comportement (si les informations clignotent, alternance de couleur, etc.). Enfin, grâce à un framework réalisé pendant la première partie de mon stage, on peut maintenant réaliser plusieurs animations de manière fluide (sans utiliser script.aculo.us) : scrolling, apparition, mouvement, etc.

Néanmoins, avant de pouvoir générer quoi que ce soit, il faut créer les clients à la main. Le personnaliseur et les providers sont générés ensuite. Les providers sont des services qui génèrent des informations à partir d'une base de données. Le client va ainsi venir récupérer ses informations et les traiter. Le personnaliseur va lui venir remplir la base de données nécessaire aux providers. Le client, qui correspond à l'affichage de Yourcast, est entièrement écrit en Javascript et Html. Ainsi on peut l'afficher dans un navigateur commun et donc sur des écrans avec un ordinateur. On retrouve Yourcast notamment à Polytech, chez les Choralies, à Clément Ader, à l'i3s et à l'Irsam. Vous pouvez retrouver ci-dessous une image du client de l'i3s.

Figure 2 - Écran de l'i3s

Nous allons voir maintenant le contexte du stage.

B. Contexte du stage

1. Le Client Yourcast

Mon stage se situe dans une phase importante du projet. En effet, la demande de prolongation du projet est en attente, la fin est prévue en janvier 2014. On arrive donc dans la dernière ligne droite. Les clients doivent être opérationnels pour pouvoir les déployer cette année.

Au milieu du stage l'équipe a du faire une présentation à l'université d'un client personnalisé. On a donc du généré très rapidement un client. Vous pouvez le voir ci-dessous.

Figure 3 - Client généré pour la présentation à l'Université de Nice

La deuxième grosse échéance qui a rythmé la deuxième partie de mon stage est le festival les Choralies. Je détaillerai plus tard dans ce rapport tout ce qu'il s'est passé pendant cette période.

Avant de passer au cahier des charges je voudrais faire un résumé de comment marche un client au début du stage. J'ai réalisé un schéma pour être plus clair :

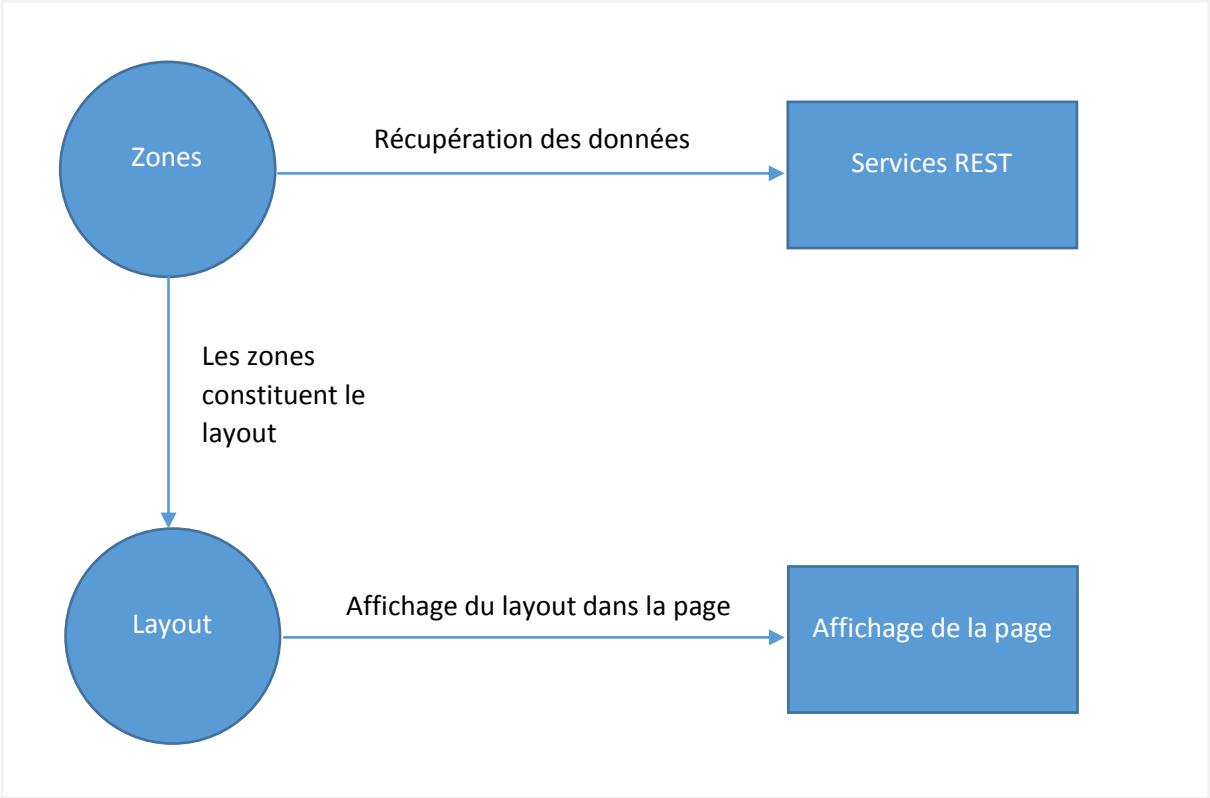


Figure 4 - Schéma résumé du client

Un client Yourcast possède un layout avec différentes zones. Par exemple pour celui généré pour l'Université de Nice, on voit 4 zones : l'une avec la météo en haut, une autre à droite avec un fil d'actualité, à gauche avec des annonces et enfin en bas avec des tweets. Les zones sont « connectées » à des services REST pour récupérer les données. Enfin ce layout est affiché selon un style dans un navigateur.

Je vais maintenant détailler un peu plus le fonctionnement de la zone.

2. Les zones

Le client possède un layout c'est-à-dire une mise en page. Un layout possède des zones. Une zone n'est pas que graphique grâce au Css, c'est aussi une classe Javascript qui permet de gérer la zone. Cette classe permet notamment de mettre à jour les informations de la zone, le chargement des images, etc. Pour pouvoir les afficher on utilise des comportements. Ceux-ci permettent de changer le contenu de la zone. On peut ainsi afficher le contenu reçu ou bien les animer.

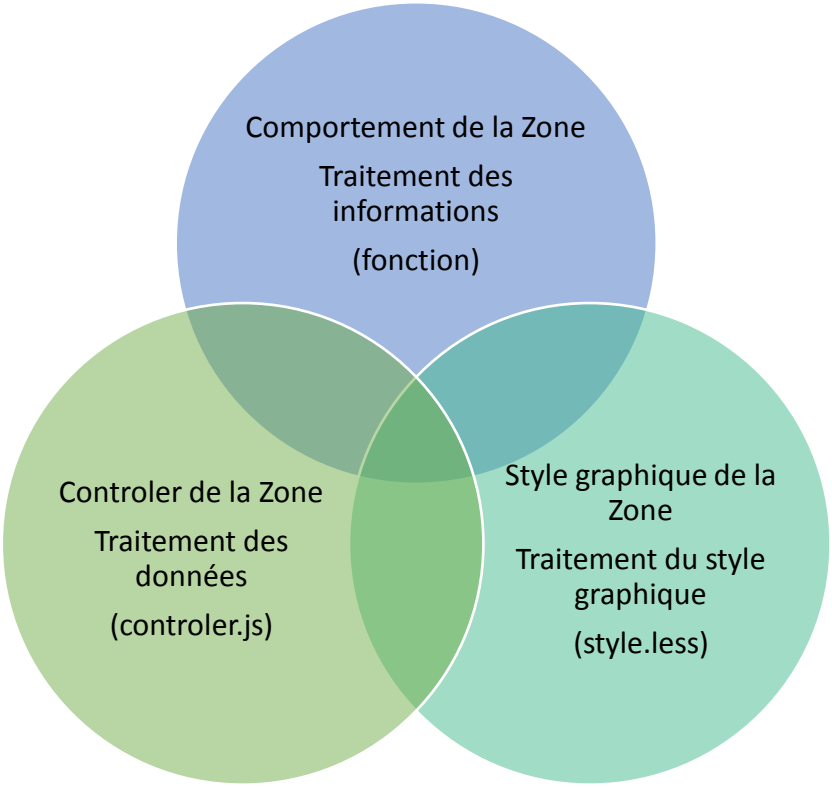


Figure 5 - Schéma d'une zone dans Yourcast

Une zone fonctionne comme ceci de base. Il reste un point non abordé lié aux zones : les renderers. Un renderer est une fonction qui transforme les données brutes en texte html traité. En effet le client envoit une requête Ajax à un service Rest qui lui renvoie un fichier de type Json. L'avantage de ce format est qu'il est exploitable dans énormément de langage car très simple et très léger.

Par exemple on reçoit les données Json suivantes :

Données Json :

```
{
  - informations: [
      - {
          - Weather2_1: [
              - {
                  - weather: {
                      - forecast: [
                          - {
                              temp_unit: "c",
                              night_min_temp: "13",
                              day_max_temp: "25",
                            - night: [
                                - {
                                    weather_code: "0",
                                  - wind: [
                                      - {
                                          speed: "7",
                                          dir: "SSE",
                                          dir_degree: "160",
                                          wind_unit: "kph"
                                        }
                                    ],
                                    weather_text: "Clear skies"
                                }
                            ],
                            - day: [
                                - {
                                    weather_code: "0",
                                  - wind: [
                                      - {
                                          speed: "11",
                                          dir: "SE",
                                          dir_degree: "125",
                                          wind_unit: "kph"
                                        }
                                    ],
                                    weather_text: "Sunny skies"
                                }
                            ],
                            date: "2013-08-26"
                        },
                      - {
```


Figure 6 - Données Json

Ces données sont inutilisables telles quelles. Il faut les rendre affichable. C'est la fonction des renderers de les transformer en données utilisables. Pour cet exemple on utiliserait un renderer de type météo. Il en existe autant qu'on peut afficher différemment les informations.

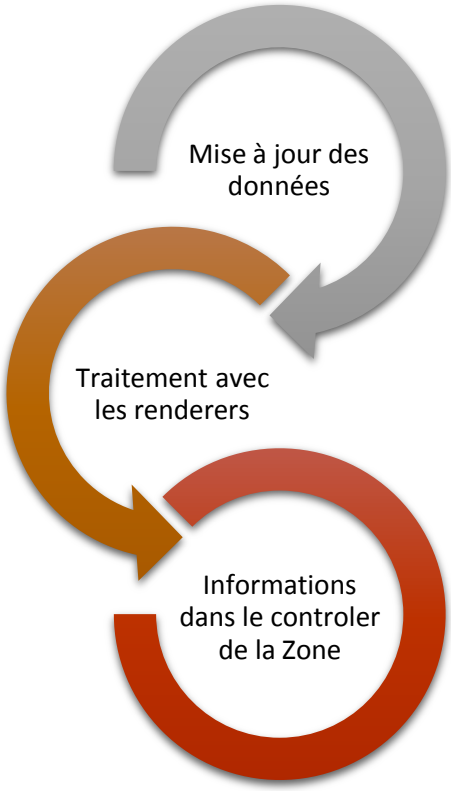


Figure 7 - Schéma de fonctionnement des données dans une Zone

C. Cahier des charges

Ce qu'il faut savoir c'est qu'au début du stage, j'ai reçu un cahier des charges prévisionnel qui pouvait évoluer en fonction de mon avancement. L'objectif principal de mon stage était de réaliser le client de Clément Ader avec quelques fonctionnalités supplémentaires :

- Vérifier les problèmes et les solutions possibles pour l'IRSAM
- Remettre en route la synthèse vocale
- Alerte : Gérer les alertes incendie, etc.
- Si j'avais le temps, regarder le problème des vidéos
- Gérer le mélange des informations

Après plusieurs réunions des objectifs ont été ajoutés :

- Ajouter les modifications du client au générateur
- Refaire une synthèse vocale
- Changer la gestion des animations
- Refaire les comportements des autres clients
- Refonte de l'Irsam

Au final le cahier des charges a augmenté mais j'ai souvent fait des modifications hors cahier des charges qui ont impacté tous les projets comme la modification des comportements.

II. Déroulement

A. Début du stage

À mon arrivée au stage j'avais un cahier des charges provisoire qui pouvait s'embellir par rapport à mon avancé. Le principal objectif était de créer un client pour Clément Ader car leur client était vieux. Pour connaître exactement les souhaits de l'établissement, j'ai pris contact avec le responsable du projet. À part des renderers spécifiques et un comportement similaire, il en est ressorti trois avancées majeures dans le client :

- **La gestion des pauses** : pouvoir mettre en pause un comportement.
- **La gestion des inactivités** : pouvoir afficher une zone par rapport à un élément (temps, paramètre à true, etc.)
- **La gestion des alarmes** : pouvoir afficher une zone lorsqu'il y a des données récupérées.
- **Ajout de la vidéo** : au début du stage le client ne pouvait pas gérer l'affichage de vidéos car les navigateurs sont assez indépendants sur la gestion des vidéos.

Pour la gestion des pauses, je me suis confronté à un problème au niveau du comportement. En effet les comportements étaient définis par des fonctions qui étaient récursives (pour passer à l'élément suivant ils avaient un paramètre indice qui s'incrémenté à chaque appel). Il y a deux problèmes à ce modèle. Le premier problème est le plus important. Si on fait une fonction « pause » pour mettre en pause un comportement dans un fichier de comportement, il faudra le mettre dans tous les autres fichiers de comportement. Ce qui va vite poser problème. Le deuxième problème est lié au premier, c'est le maintien du code. En effet rajouter une fonction qui arrête une fonction pour la reprendre après, on voit que ça peut devenir très vite compliqué.

Pour résoudre ce problème, j'ai créé un nouveau modèle de comportement avec une nouvelle classe qui permet de faire des manipulations tels que passer à l'élément suivant, précédent, mettre en pause, arrêter le comportement, etc. Une classe est plus simplement maintenable et peut être hérité pour avoir de nouveaux comportements sans avoir besoin de réécrire toutes les fonctions.

L'écriture de ce comportement a permis de consolider une première fois le client grâce aux tests unitaires mise en place avec la librairie **QUnit** (http://qunitjs.com/). Avec ces tests unitaires on réduit les problèmes du client. Ci-dessous vous pouvez une partie des tests effectués sur la classe « Comportement ». On tests en priorités la crédibilité des informations reçues c'est-à-dire la valeur des paramètres. Par exemple dans une fonction on veut une zone donc on contrôle que le paramètre contient bien une instance de la classe Zone.

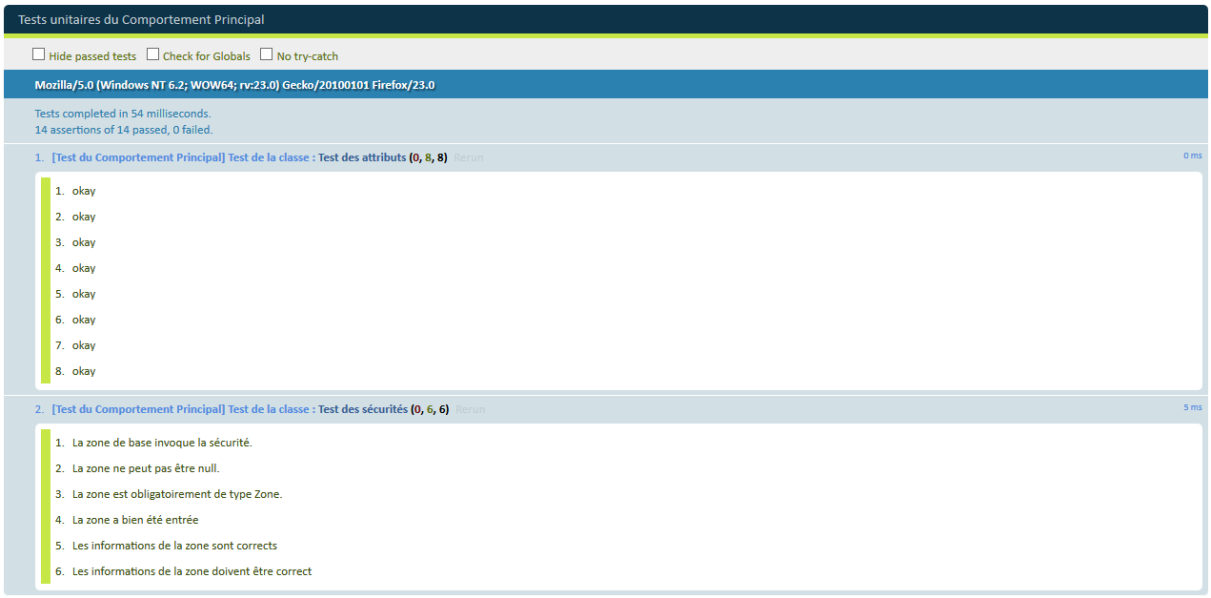


Figure 8 - Test unitaire de la classe comportement

La gestion des inactivités et la gestion des alarmes sont gérées de la même manière grâce au nouveau modèle de comportement. L'écriture de ces nouveaux comportements à modifier une première fois l'architecture du client. En effet, la mise à jour des informations de la zone se faisait toutes les 10 minutes. Le problème était que lorsque la zone se mettait à jour le comportement était arrêté même s'il était au milieu de sa boucle. Maintenant le rafraîchissement des informations d'une zone se fait lorsque le comportement atteint la fin de sa boucle. Un résumé de changement est disponible ci-dessous.

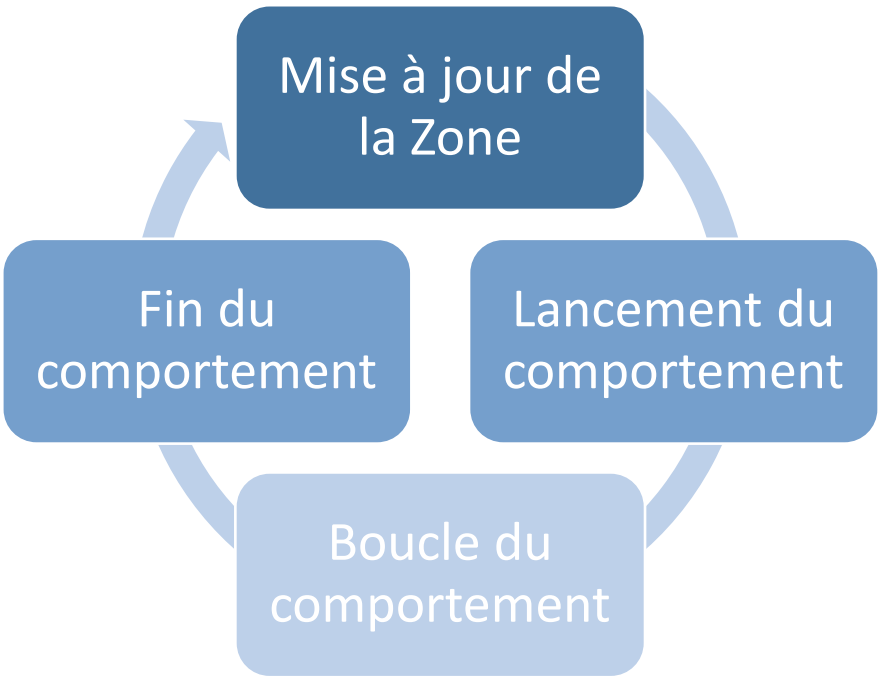


Figure 9 - Schéma nouvelle architecture de mise à jour des informations de la zone

La gestion de la vidéo a été une vraie étude à cause des navigateurs. En effet ceux-ci ne gèrent pas les mêmes formats de vidéos nativement grâce à l'Html5 et les lecteurs de vidéos ne sont pas assez complets et évolué en open-source, que ce soit les lecteurs flash que les lecteurs Html5. Dans les formats supportés par Mozilla Firefox et Google Chrome (ce sont les deux navigateurs retenus pour la compatibilité) on retrouve l'ogv et le webm. Dans le tableau ci-dessous vous pouvez retrouver un récapitulatif.

Navigateurs	H.264/MP4	OGG Theora	WebM
	non	Firefox 3.5+	Firefox 4+
	non	Opera 10.5+	Opera 10.6+
	Internet Explorer 9+	non	si les codecs sont installés
	Google Chrome 4-16 mais supprimé après	Google Chrome 4+	Google Chrome 6+
	Safari 5+	non	non

Figure 10 - Récapitulatif des formats de vidéos gérés par les navigateurs

Au final vu que les navigateurs choisis sont le Chrome et Firefox, le format de vidéo retenu et le webm. On voit que déjà trois navigateurs gèrent nativement ce format (plus internet explorer si les codecs sont installés). Il y a un autre problème que nous avons rencontrés et le format que renvoi une source. En effet, on peut remarquer que chaque données envoyées dans le web possède un type Mime (ou Content/Type). Lors d'une récupération de vidéo, je n'arrivais pas à afficher la vidéo faute à ce petit paramètre. En effet, par exemple pour un format de type webm, ce paramètre vaut « video/webm ». Or ici le type était « text/plain ». On a pu voir ainsi encore des différences entre les navigateurs car Firefox arrivait à afficher la vidéo alors que chrome non.

Au final j'ai obtenu cet écran pour Clément Ader :

Figure 11 - Écran pour Clément Ader

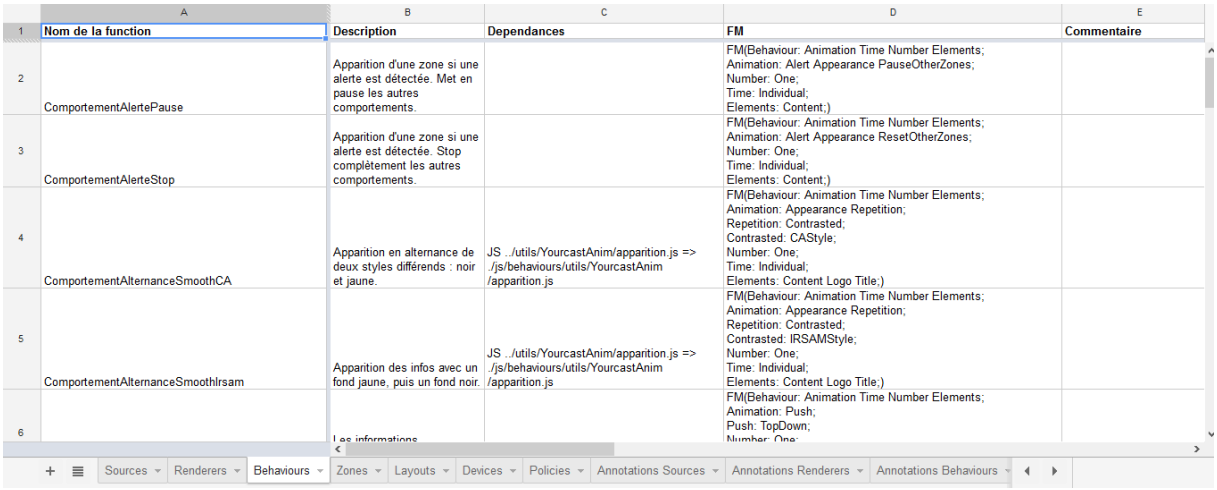
On va voir maintenant comment j'ai dû adapter les modifications que j'avais faites sur mon client à la génération pour que tous les clients bénéficient de ces modifications.

B. Adaptation des modifications pour la génération des clients

1. La génération des clients

Le problème, qui a pris le plus de temps à résoudre, est le passage du développement du client à la génération. En effet le projet Yourcast est avant tout un produit livrable, qui peut être générer et personnalisable. La génération a des contraintes élevées mais permet de créer un client très rapidement avec énormément de variabilité.

Il faut savoir que la génération se fait à partir de deux éléments : le répertoire git et un tableur. Je vais d'abord présenter le tableur.



	A	B	C	D	E
1	Nom de la function	Description	Dependances	FM	Commentaire
2	ComportementAlertePause	Apparition d'une zone si une alerte est détectée. Met en pause les autres comportements.		FM(Behaviour: Animation Time Number Elements; Animation: Alert Appearance PauseOtherZones; Number: One; Time: Individual; Elements: Content;)	
3	ComportementAlerteStop	Apparition d'une zone si une alerte est détectée. Stop complètement les autres comportements.		FM(Behaviour: Animation Time Number Elements; Animation: Alert Appearance ResetOtherZones; Number: One; Time: Individual; Elements: Content;)	
4	ComportementAlternanceSmoothCA	Apparition en alternance de deux styles différends : noir et jaune.	JS ../utils/YourcastAnim/apparition.js => ./js/behaviours/utils/YourcastAnim /apparition.js	FM(Behaviour: Animation Time Number Elements; Animation: Appearance Repetition; Repetition: Contrasted; Contrasted: CAStyle; Number: One; Time: Individual; Elements: Content Logo Title;)	
5	ComportementAlternanceSmoothIrsam	Apparition des infos avec un fond jaune, puis un fond noir.	JS ../utils/YourcastAnim/apparition.js => ./js/behaviours/utils/YourcastAnim /apparition.js	FM(Behaviour: Animation Time Number Elements; Animation: Appearance Repetition; Repetition: Contrasted; Contrasted: IRSAMStyle; Number: One; Time: Individual; Elements: Content Logo Title;)	
6		Les informations		FM(Behaviour: Animation Time Number Elements; Animation: Push; Push: TopDown; Number: One;	

Sources | Renderers | Behaviours | Zones | Layouts | Devices | Policies | Annotations Sources | Annotations Renderers | Annotations Behaviours

Figure 12 - Tableur pour la génération

Ce tableur permet de changer presque toute la variabilité de Yourcast : les layouts, les comportements, les renderers, les sources, les zones, etc. ainsi que les Feature Model (un feature model est élément composé d'un layout, de x zones, de x renderers et de x sources). Il a fallu changer toute la page sur les comportements et ajouter un certain nombre après le découpage de l'écran de l'EPU.

Donc le générateur récupère toutes les données de ce tableur traite tous les feature model possible et génère un configurateur. Avec ce configurateur l'utilisateur choisit son layout, ses zones, etc. ce qui donne son client Yourcast. Et il lance la génération du client. À ce moment-là le générateur va récupérer les fichiers et les traiter. Ces fichiers se trouvent dans un répertoire GIT.

Pour que tout ce processus fonctionne encore après les modifications, il m'a fallu respecter toute l'architecture dans le GIT et des architectures au niveau des zones qui sont générés en Java (car l'utilisateur peut choisir un nom pour la zone).

Le plus long a été d'adapter ces nouveaux comportements aux autres clients (c'est-à-dire tous les refaire car il ne fonctionnait plus avec le nouveau contrôleur de la zone).

2. Étude du chargement du client

Le client Yourcast utilise une librairie pour le style sur la page appelée **Less** (http://lesscss.org/). Cette librairie pallie tous les problèmes du css :

- **les variables**

Un grand problème du css est l'absence totale de variables. Lorsque vous voulez changer une couleur qui est par exemple dans de nombreux block vous devez tous les changer à la main ce qui peut devenir une perte de temps. Pour résoudre cela Less introduit cet outil.

Exemple :

```
@nice-blue: #5B83AD;
@light-blue: (@nice-blue + #111);

#header { color: @light-blue; }
```

- **les mixins**

Les mixins sont des blocks d'instructions que vous pouvez mettre n'importe quel block d'instruction comme les defines dans le langage C.

Exemple :

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}

#menu a {
  color: #111;
  .bordered;
}
```

Et d'autres évolutions comme les namespaces, les opérations, etc. Ce langage est donc très utile. Cependant le client utilisait la version non compilé du langage c'est-à-dire le fichier Javascript Less.js. Le problème est qu'à chaque changement dans la page html, les fichiers Less sont recompilés et donc ralentit le client. Ils sont compilés une première fois au chargement ce qui ralentissait le chargement. Pour pallier ce problème j'ai fait un script java qui permet de convertir les fichiers less en Css avec la librairie LessCompiler.

Ainsi le temps de chargement a été reduit de moitié dans certains cas. La deuxième variable du chargement du client est le chargement des données des services REST. On ne peut rien faire sur ce point-là. À la fin du stage j'ai fait en sorte que le contrôleur de la zone supporte plusieurs adresses pour les données ce qui permet d'avoir moins de risque de bug à ce niveau-là.

C. L'évènement Choralies

Qu'est-ce que les Choralies ? Les Choralies est un festival qui rassemble des chorales et des chœurs tous les trois ans à Vaison-la-Romaine. Pour cette édition 2013, des écrans avait été placé un peu partout dans la ville pour informer les gens sur le déroulement du festival et divers autres informations. Il y avait quatre clients répartis dans toutes la ville plus celui pour mobile : au théâtre, aux ateliers, celui en attente.

Le développement s'est énormément accéléré et de nombreuses fonctionnalités sont apparues. Un problème apparaît très vite, c'est la stabilisation du client du point de vue du code principale et des bugs qui apparaissent sans cesse et de plus en plus souvent à cause de toutes ces modifications.

L'une des avancées est la gestion des erreurs qui indique le fichier, la ligne et une description de l'erreur. Ceci permet de résoudre un bug beaucoup plus rapidement qu'avant. Voici la console Firebug après la mise en place de cette gestion d'erreurs.

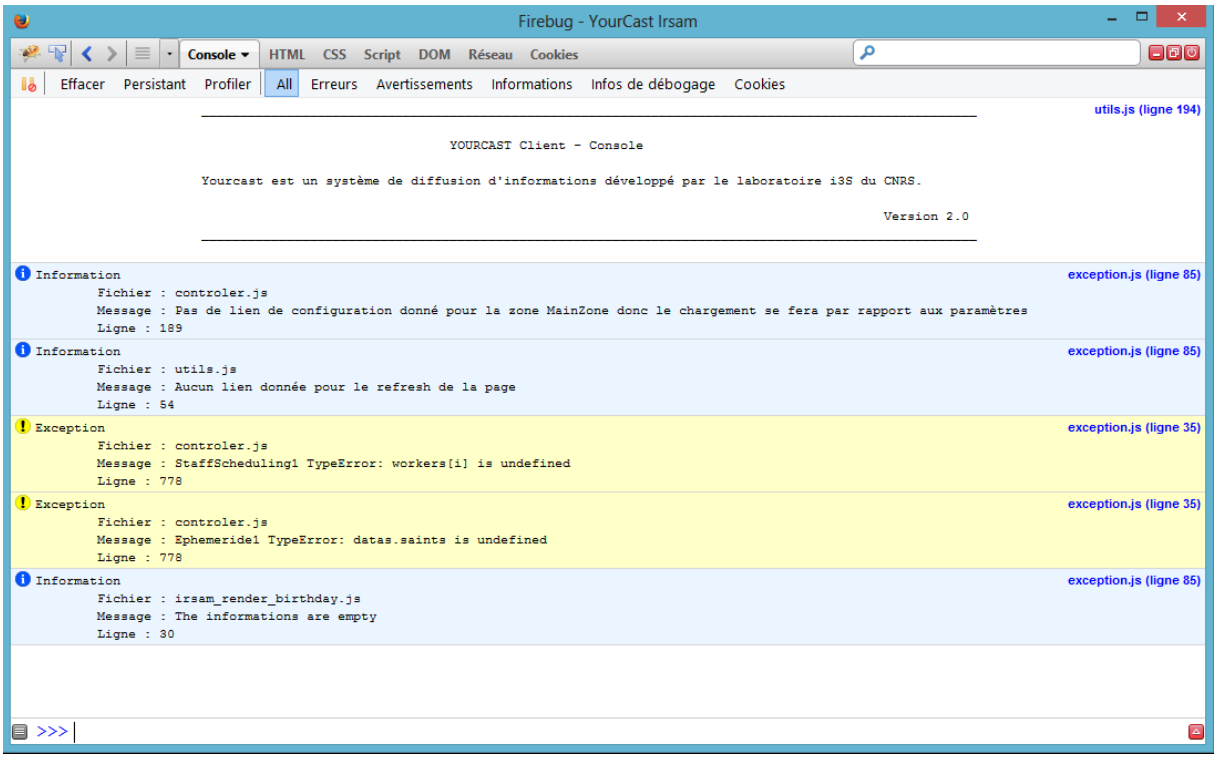


Figure 13 - Console Firebug pour la gestion des erreurs

On peut voir des **Informations** qui permettent de savoir lorsque le client prend une décision. Les **Exceptions** sont vraiment des erreurs dans le client mais qui ne bloquent pas le client. Au contraire des **Erreurs** qui bloquent le client. Toute cette gestion d'erreur s'inscrit dans une stabilisation du client en ajoutant des try catch le plus possible pour éviter que le client se bloque à la moindre erreur.

Beaucoup de problèmes liés aux modifications de la première partie se sont répercutés sur les clients des Choralies. En effet, il a fallu modifier en profondeur tous les clients dont ceux des Choralies un mois avant l'échéance.

Au final le client choralies ressemblait à ce client :

Figure 14 - Client des Choralies

Vous pouvez retrouver ce client à l'adresse suivante http://choralies2013.supralog.com/tv-replay/.

D. Outils et méthodes utilisés

Dans cette partie je vais détailler les outils et les méthodes utilisées tout au long de ce stage. Tout d'abord je vais parler du Javascript car on ne l'a jamais vu en cours. Ce langage a longtemps été décrié et il était recommandé de le désactiver. Désormais il a bien évolué et est devenu très puissant. Il appartient à la famille des langages de types faibles comme le php (les variables sont indifférencié contrairement au Java ou au C dans lesquels les variables sont de type Int, Char, etc.). On peut mettre ce qu'on veut dans les paramètres et dans les variables ce qui pose souvent des problèmes.

J'ai ainsi pu voir un framework très puissant AngularJs et une librairie semblable à Jquery : Prototype. On va commencer par AngularJs qui est un framework open source qui a été développé par Google et sa communauté (à travers GitHub). Il a pour but de faciliter la création d'interface monopage. En effet lorsqu'on page on ne change pas de fichier html, on change juste le contenu d'un div à travers une requête Ajax (qui récupère le contenu de la prochaine page) et d'une manipulation du DOM (Document Object Model). J'ai notamment utilisé ce framework pour créer une interface de test pour la synthèse vocale.

La librairie Prototype est complètement différente d'Angular. Elle est ici pour faciliter l'utilisation du JQuery. Elle est très semblable à Jquery sauf sur un point : la gestion des classes. En effet, le Javascript ne gère pas nativement les classes comme on l'entend : balise *Class nomClasse*. Pour créer une classe il faut utiliser des fonctions. Prototype apporte cette balise et une gestion de l'héritage.

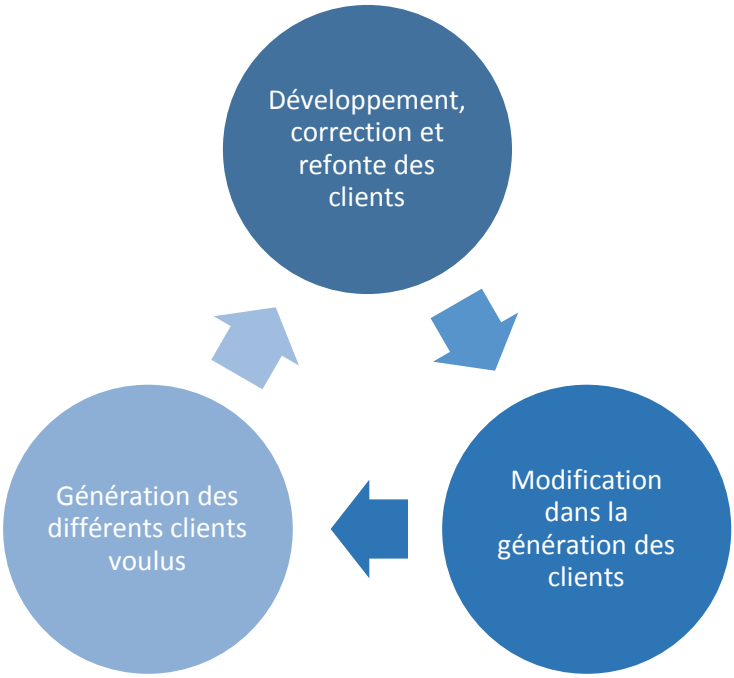


Figure 15 - Développement agile de la deuxième moitié du stage

Pendant les études on développe des projets très linéaires. On a un cahier des charges et on doit le respecter. Ici on est vite arrivé à un développement agile, c'est-à-dire que lorsque j'effectué une modification, une création il y avait toujours un retour qui permettait de réorienter mon travail et qui bien souvent partait dans une direction inverse à celle de départ.

L'évènement des Choralies a complètement accéléré le développement du client et en même temps changé la façon de travailler. On est passé à un développement beaucoup plus agile. C'est-à-dire qu'on est passé d'un développement linéaire à un développement dynamique entre les deux éléments précédents. On peut résumer ce nouveau modèle comme le schéma ci-dessous. Je développais de nouvelles fonctionnalités, corrigés des bugs de générations, faisait une refonte des clients puis on faisait les modifications dans le répertoire principal de la génération et enfin on régénérait le client pour voir si tout était comme on voulait.

Cette méthode est rapide et efficace mais demande une disposition de l'utilisateur qui fait les retours beaucoup trop importante. Cette méthode est bien souvent inenvisageable. De plus j'ai eu la chance de pouvoir réaliser plusieurs réunions (dont une en visio conférence avec des chercheurs de Lille) qui ont permis de ne jamais m'éloigner des priorités de développement.

III. Contribution

A. Modification du client

Ma plus grande contribution fut sur la modification du client. J'en ai parlé précédemment donc je vais faire un résumé des modifications dans cette partie. Je vais commencer par le début c'est-à-dire le chargement du client.

1. Chargement du client

Il faut savoir qu'il y a deux variables pour le chargement du client : le chargement des données et le chargement du style. Le client utilise la librairie Less pour gérer ses styles. L'avantage de cette librairie réside dans le fait que tout ce qui manque au Css a été rajouté : les variables, les fonctions, les mixins, etc. La librairie Less causée de grand ralentissement et a donc été convertit en Css lors de la génération. Le développement se fait donc toujours avec la librairie Less.

La deuxième variable n'est pas contournable ou corrigeable car c'est le chargement des données du service REST en Ajax qui prend un peu de temps par moment. Ceci peut devenir gênant pour un utilisateur c'est pour cela que j'ai rajouté une barre de chargement sur le client de l'Irsam.

2. Modification des comportements

La modification des comportements a été ma plus grande contribution car elle a nécessité un remaniement complet des autres comportements des clients. La nouvelle classe de comportement comporte des tests unitaires et possède de nombreuses fonctions utiles telles que la pause, le passage à l'élément suivant ou précédent. Vous pouvez retrouver toutes les fonctions disponibles sur l'image ci-dessous :

```
var Comportement = Class.create({

    /**...*/
    initialize: function() {...},
    /**...*/
    next: function() {...},
    /**...*/
    before: function() {...},
    /**...*/
    goto: function(indice) {...},
    /**...*/
    pause: function() {...},
    /**...*/
    run: function() {...},
    /**...*/
    clear: function() {...},
    /**...*/
    reset: function() {...},
    /**...*/
    stop: function() {...},
    /**...*/
    isRunning: function() {...},
    /**...*/
    loop: function() {...},
    /**...*/
    securiteZone: function() {...},
    /**...*/
    securiteInfosZone: function() {...},
    /**...*/
    setZone: function(nouvelle_zone) {...}

});
```

Figure 16 - Fonctions disponibles dans la nouvelle classe des comportements

Cette classe fonctionne comme ceci :

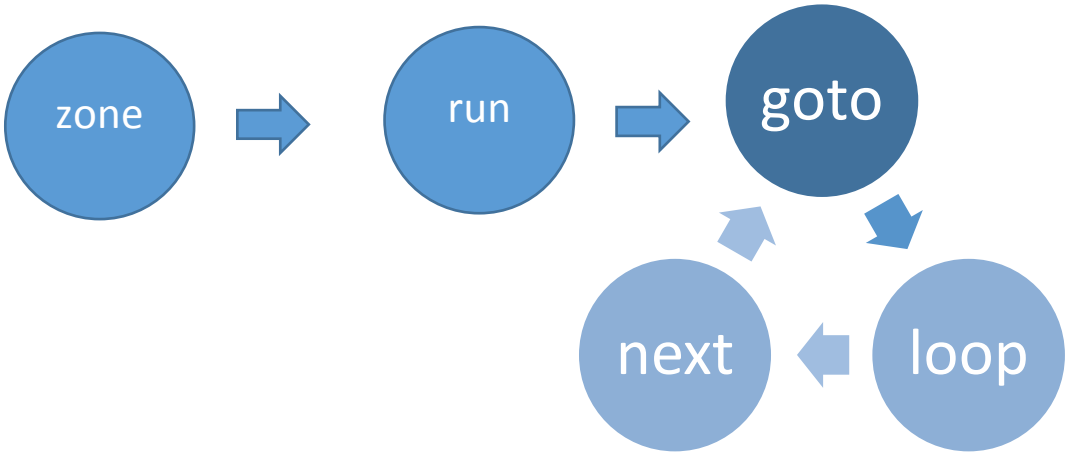


Figure 17 – Schéma du fonctionnement du comportement

3. Stabilisation du client

La stabilisation du client est passée par deux grandes étapes : l'ajout de nombreux try catch pour éviter le blocage du navigateur sur un problème et l'ajout de classes pour les erreurs. Ces classes d'erreurs permettent notamment (si un jour cela devient nécessaire) d'envoyer ces erreurs sur un serveur externe. Ces erreurs affichent le minimum d'informations pour le développeur : une description, la ligne et le fichier où se situe l'erreur.

4. Plus de possibilité de personnalisation

J'ai essayé tout au long du stage d'ajouter des possibilités de personnalisation pour le client. Cela a commencé par un ajout d'un ordre pour les renderers dans une zone. Ensuite par une externalisation des principales données d'une zone (ce qui permet de la personnaliser par la suite). La personnalisation est passée aussi par le contrôleur général qui permet de rafraîchir le client ou bien même mettre en pause celui-ci.

5. Mise à jour des informations

J'en ai parlé précédemment mais cela a été un grand changement dans le client. En effet au lieu de rafraîchir les données de la zone tous les dix minutes on rafraîchit lorsqu'on est arrivé au bout des informations ce qui permet d'éviter de louper des informations ou de couper le comportement en plein milieu.

6. Ajout d'un contrôleur général

Ce qui manqué au client jusqu'alors était un moyen de gérer le client dans son ensemble. En effet chaque zone faisait ce qu'elle voulait et il n'y avait pas de relation entre elles. L'ajout d'un contrôleur

général était indispensable si on veut mettre des relations entre les zones et de pouvoir les gérer en même temps. Ce contrôleur est principalement utilisé pour l'instant pour les alertes en mettant en pause toutes les zones. Il permet aussi de rafraîchir tout le client lorsqu'une variable change de valeur dans un Json récupéré en Ajax.

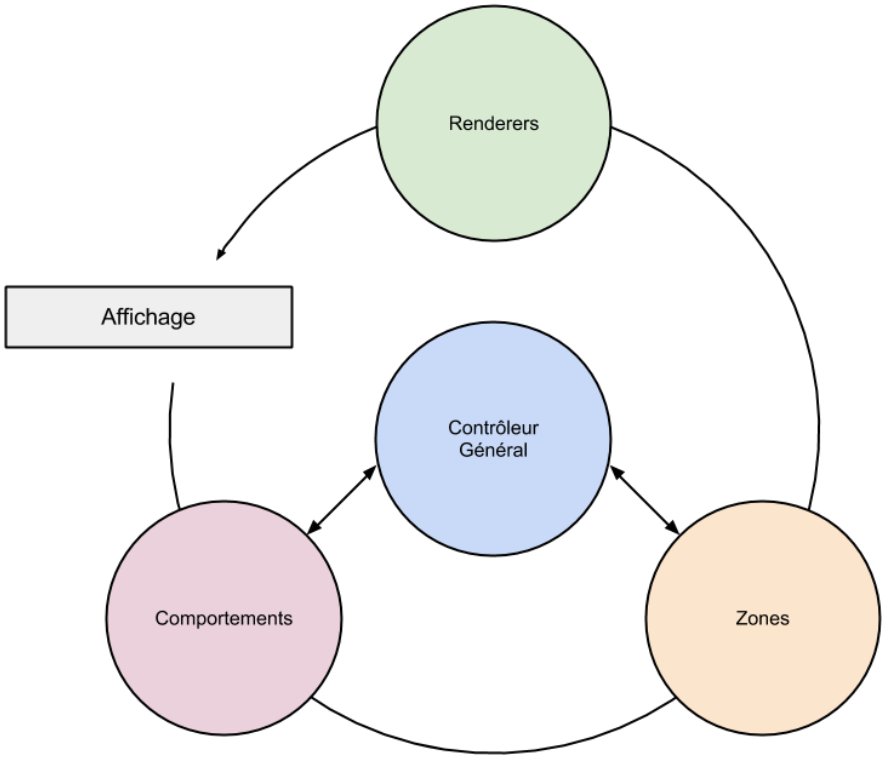


Figure 18 - Schéma du fonctionnement du Contrôleur Général

B. Suppression des dépendances

Dans un projet en évolution et surtout sur le web les dépendances vers d'autres librairies peuvent devenir gênantes surtout si le programme n'utilise qu'une infime partie de la librairie. En effet si vous utilisez une librairie qu'à 1 ou 2 %, vous perdez des ressources et vous rajoutez du temps de chargement.

Il y avait deux dépendances qui étaient gênante :

- La dépendance au niveau au framework d'animation script.aculo.us
- La dépendance au niveau du client de la synthèse vocale (dépendance à Windows et à une synthèse vocale vieillissante). La synthèse vocale était un programme et non une interface web.

1. Le framework d'animation

La première dépendance n'a pas été facile a enlevé pour deux raisons. La première est qu'il n y a que trois façons d'animer un objet sur le web : flash, le javascript ou le css. Le plus utilisé (mais pratiquement plus) était le flash car il est très fluide. L'inconvénient est le temps de chargement qui est très long. Le problème du Javascript est que les navigateurs ne le gèrent pas très bien encore. Les animations sont faites sur les ressources du navigateur ce qui montrer des effets de ralentissement lors de l'animation.

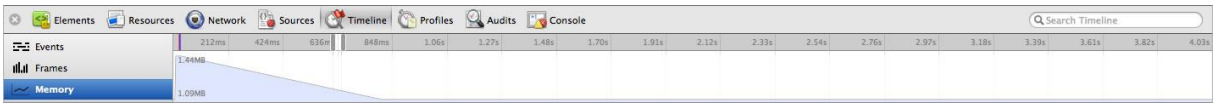


Figure 19 - Mémoire utilisée par une animation de 300 div en Css

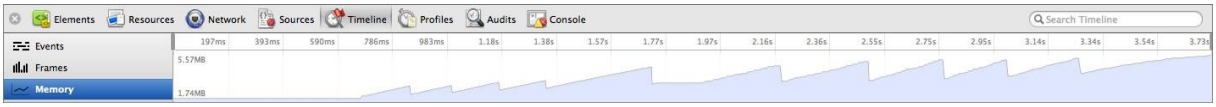


Figure 20 - Mémoire utilisée par une animation de 300 div avec JQuery

On voit très bien la différence de consommation entre les deux (pour plus d'informations voici l'article du site d'opera http://dev.opera.com/articles/view/css3-vs-jquery-animations/ où j'ai récupéré ces images). Les animations en Css sont donc plus rapides, plus fluides car le navigateur utilise le CPU de l'ordinateur directement. Pourquoi tous les Frameworks d'animations et les sites n'utilises-t-il pas le css pour animer ? Tous simplement parce qu'il faut avoir Css3 pour pouvoir utiliser les animations. Et si on regarde la compatibilité il manque le navigateur le plus utilisé InternetExplorer.

Voici un graphique montrant les navigateurs les plus utilisés en Europe :

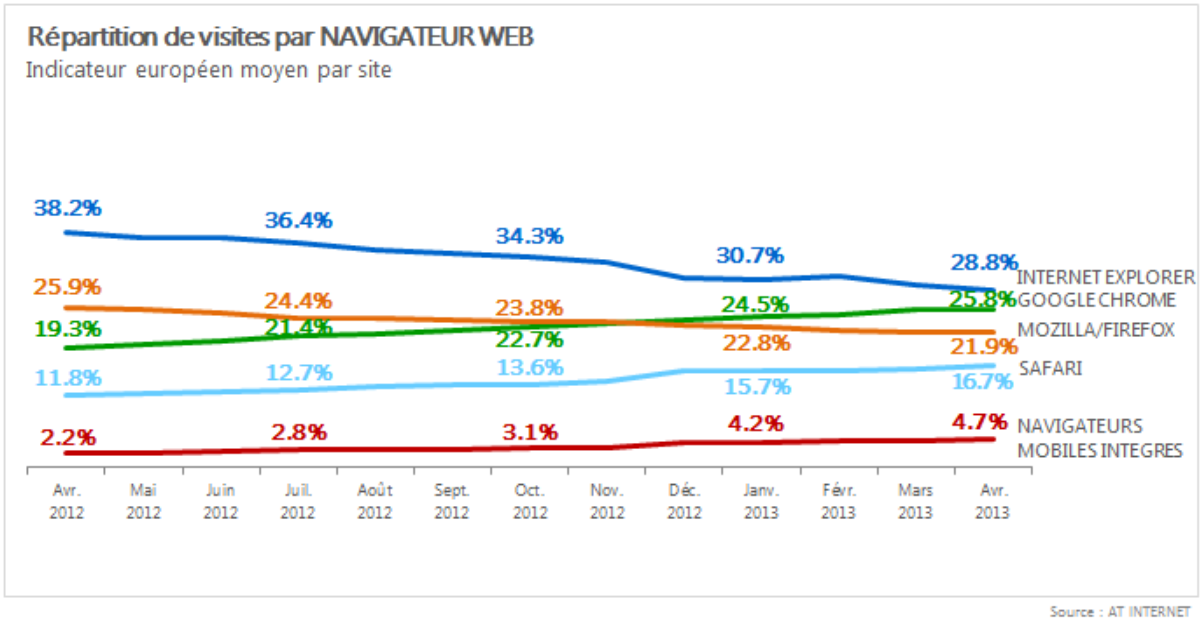


Figure 21 - Graphique des navigateurs les plus utilisés en Europe

Et voici les compatibilités sur les navigateurs :

Figure 22 - Compatibilité des animations Css dans les navigateurs

Néanmoins vu que les navigateurs choisis sont Chrome et Firefox, les animations ont été transposées en Css.

2. La synthèse vocale

La deuxième dépendance a enlevé était le framework de la synthèse vocale qui était trop vieux, impossible à maintenir et dépendant de Windows. Trop de problème pour un seul framework. La piste la plus probable était la synthèse vocale de Google (Google TTS). Malheureusement les requêtes au serveur sont refusées sauf si elle vient de Google Translate. Impossible de l'utiliser donc. Après des recherches infructueuse, on a réussi à trouver une librairie qui permettait de faire ce que j'avais besoin pour cette synthèse vocale : **VoiceRss**. Malheureusement un nouveau problème apparaissait : Firefox ne lit pas les fichiers audio en mp3.

Il a fallu trouver une librairie capable de lire les fichiers audio sur n'importe quel navigateur. La librairie que j'ai choisie est **SoundManager2** (http://www.schillmania.com/projects/soundmanager2/). Elle est légère et peut lire beaucoup de format. Avec ces deux librairies, j'ai créé les renderers, adapter la classe de la Zone et créer un comportement pour la Synthèse Vocale ce qui permet de l'utiliser sur les clients normaux.

C. Ajout et refonte de nouveaux clients

Je vais récapituler les clients ajouté et refait. Le premier client à avoir été ajouté est le client de Clément Ader. Ensuite le client de l'i3s a été refait puis découpé. Vous pouvez retrouver un exemple dans la partie « Contexte du stage » pour la présentation à l'université de Yourcast. Enfin le dernier client que je n'ai pas encore traité et celui de l'Irsam refait pendant les deux dernières semaines du stage. Vous pouvez voir l'écran de chargement ci-dessous.

Figure 23 - Page de chargement du client de l'Irsam

On voit deux évolutions par rapport aux autres clients : une barre de chargement et une console. La barre de chargement permet de montrer l'avancement à l'utilisateur pour qu'il ne soit pas perturbé si le client n'avance pas. La console n'est affichée que lors du développement du client et affiche les mêmes informations que dans Firebug.

Après une réunion avec le responsable du projet à l'Irsam, il est ressorti que le client de Clément Ader possédait des fonctionnalités vraiment intéressantes et qu'il faudrait intégrer certaines sur celui de l'Irsam. On y distingue notamment la ligne rouge en dessous de l'headerZone et le logo en haut à gauche. Vous pouvez voir ce client ci-dessous :

Figure 24 - Client de l'Irsam

Le dernier client a été réalisé par Clément Duffau. Je me permets de le mettre sur ce rapport pour des problèmes qui ont été plutôt dérangeant et qui ont pris plusieurs jours à résoudre. Ce client est pour Polytech Sophia.

Les bugs qui sont apparus :

- Lors du chargement d'un script Javascript grâce à une autre fonction Javascript on appelait la fonction « document.write ». Néanmoins en lançant le client tout était blanc et sans rien dans le code source. Une chose qu'il faut savoir c'est que cette fonction ne fonctionne pas lorsque la page est chargée. Elle réécrit tout ce qui dans la page. Pas très utile pour le chargement de script ! Pour pallier ce problème on avait deux choix possible : synchrone (et page non chargé) avec « document.write » ou asynchrone en ajoutant directement le script dans l'en-tête de la page.
- Un autre problème est survenu lorsqu'on a voulu charger la librairie less.js en synchrone. L'erreur rapportée par Firebug est que la classe Date n'existait pas. Rien à voir avec le chargement de cette librairie. En chargeant en asynchrone la page tout était réglé.
- Le dernier problème est apparu lors du changement de contenu entre chaque passage des éléments. J'avais mis en place un changement de contenu sécurisé (suppression de l'interprétation du Javascript et autre langage web). Cependant, il en avait besoin et on a perdu du temps sur ce problème pour savoir comment on pouvait le contourner. Finalement on a enlevé cette sécurité.

Vous pouvez voir le client développé pour Polytech ci-dessous.

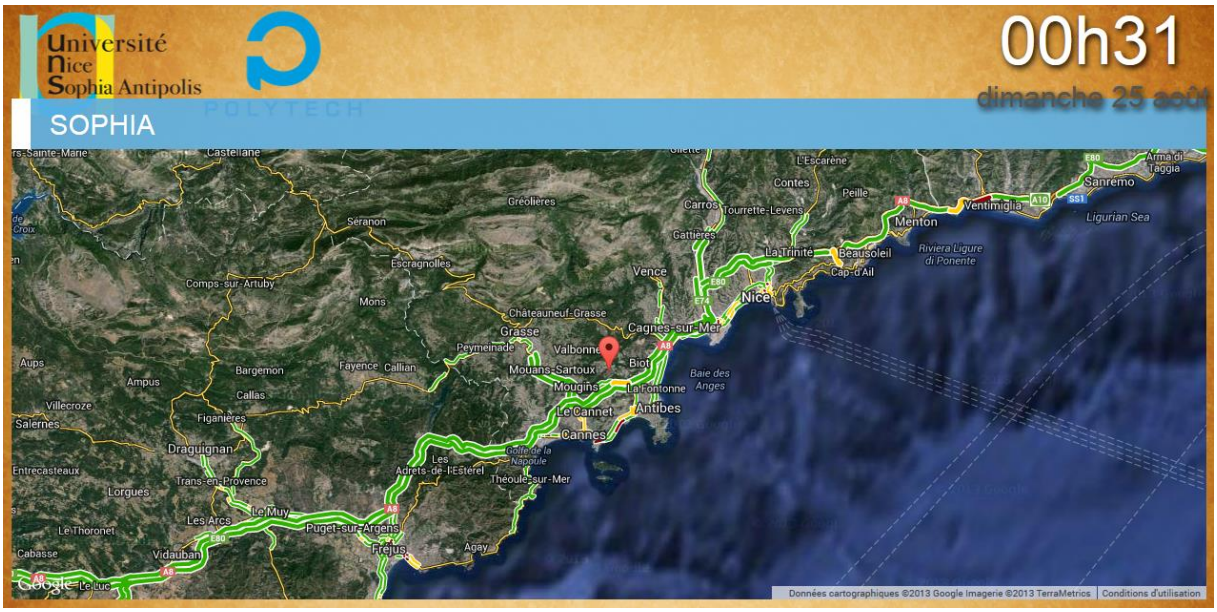


Figure 25 - Client de Polytech de Sophia Antipolis créé par Clément Duffau

En conclusion les modifications apportées au client ont été longues et ont affectées beaucoup de clients qui étaient en développement ce qui a souvent posé problème sur la version que chaque

développeur avait sur les fichiers principaux : le contrôleur de la Zone, le fichier index.html et le fichier utils.js.

IV. Bilan du stage

A. Bilan du stage pour l'entreprise

Pour l'entreprise c'est-à-dire pour le projet Yourcast à la fin de ce stage, ils ont un client beaucoup plus stable qu'avant avec des comportements complétement personnalisable et avec de nombreuses fonctions disponibles. De plus, la remise à niveau de la synthèse vocale et la suppression de la dépendance à la librairie script.aculo.us leur permettra d'être indépendant au niveau de leur choix et des possibilités envisagées.

L'ajout de fonctionnalités comme la personnalisation de l'ordre des renderers, le mélange des informations, la possibilité de mettre plusieurs lien de données pour éviter la perte de connexion lors de la mise à jour des données, l'externalisation des données tel que l'ordre des renderers, leur temps d'affichage, etc. permet une plus grande variabilité dans la ligne de produit et dans la personnalisation.

B. Bilan du stage pour l'étudiant

Pour ma part, j'ai appris un langage que nous n'avions jamais vu en cours : le Javascript. Le Javascript est un langage client. C'est-à-dire qu'il s'exécute sur le navigateur de l'utilisateur. Ainsi ce langage permet de créer des sites web dynamiques. En effet lorsque la page est chargée des langages comme l'html ou le php ne peuvent plus intervenir, c'est pourquoi le Javascript existe.

Des interfaces utilisent même presque que du Javascript. Google est très porté sur ce langage. On peut voir la création d'AngularJs (système entièrement en Javascript et Ajax. Exemples d'applications faites avec AngularJs : http://builtwith.angularjs.org/). Gmail et Gdrive sont aussi écris en Javascript. J'ai ainsi créé une interface pour tester la librairie VoiceRss.

Comme je l'ai dit précédemment, lorsqu'on étudie à l'école le développement agile n'a pas sa place pour toutes les contraintes évidentes telles que le temps qu'il faut, le cahier des charges ne peut pas changer en cours de route contrairement à un développement agile. Un sujet de projet à l'école est bien souvent commun à plusieurs groupes et donc pour noter le travail d'un groupe le cahier des charges ne peuvent pas changer.

Conclusion

Ce stage a été une vrai réussite pour moi tant sur le point programmation que humain. Mon adaptation a été très rapide et très chaleureuse. J'ai pu participer à des réunions d'équipes dont une avec visio-conférence avec des chercheurs de Lille. J'ai aussi pu voir la présentation de fin de doctorat de M. Christian Brel.

Au niveau technique, j'ai énormément appris en Javascript et dans les méthodes de travails tels que GIT ou le développement agile. J'ai eu la chance de découvrir de nouvelles librairies telles que Prototype, Less, script.aculo.us et encore VoiceRss.

Tous mes objectifs de de stage présent dans le cahier des charges ont été remplis et j'ai essayé de rendre le client le plus fonctionnel (absence de bug) possible avec le plus de nouvelles fonctionnalités.

Malheureusement je n'ai pas pu arriver à faire ce que je voulais concernant les comportements. En effet il y a plusieurs comportements qui « fusionner » donne un nouveau tout à fait fonctionnel avec des propriétés différentes. Ce procédé se rapproche énormément de la composition et je n'avais pas le temps de traiter ce point avec le peu de temps qu'il me rester.

Enfin désirant continuer en doctorat dans deux ans, ce stage m'a énormément profité car j'ai pu récolter énormément d'informations sur cette poursuite d'étude en discutant avec les tuteurs des doctorants et les doctorants eux-mêmes.

Table des illustrations

Annexe

Liens vers les différents clients Yourcast :

- Clément Ader : http://grid-vm3.unice.fr:8080/yourcast-CA-client/
- Irsam : http://grid-vm3.unice.fr:8080/NewIrsam/
- Polytech : http://grid-vm3.unice.fr:8080/yourcast-polytech-client/
- Choralies : http://choralies2013.supralog.com/tv-replay/
- I3s : http://grid-vm3.unice.fr:8080/yourcast-EPU-client/