
Université Lille 1
IUT-A
Département informatique

Web-Tv ou playlist multi-sourcing

Rapport de stage

Date présentation : 19 juin 2017

par

Maïté Delecaut

Encadrant entreprise : Monsieur Benoit Hermant

Encadrant universitaire : Monsieur Michael Hauspie, professeur à l'IUT A de Lille1

Norsys



Avant-Propos

Si les noeuds m'étaient contés ou plus simplement connaissez-vous les noeuds borroméens ? Il s'agit de trois noeuds assemblés de telle sorte que si l'un est défait, tous se séparent. Seul l'union de ces trois formes permet la pérennité de cette entité qui se veut inséparable. Cette figure est la métaphore de l'idéal de Norsys, responsabilités, vocations, métiers, chacun va de soi mais aucun ne va sans les autres.

Remerciements

Je souhaite dédier cette section à tous ceux qui ont contribué à la réussite de ce stage permettant ainsi l'aboutissement d'un projet personnel et professionnel.

Ces premières lignes seront pour la société Norsys que je remercie pour l'opportunité qu'elle m'a offerte. Je remercie de ce fait, Sylvain BREUZARD, PDG de Norsys groupe ainsi que Denis CASSORET, directeur de l'agence Norsys Régions Nord pour m'avoir accueilli dans son agence.

Je remercie également Mélanie WILFART, directrice des ressources humaines pour m'avoir offert la possibilité de participer à cette belle aventure qu'ont été ces trois mois de stage.

Je souhaite également faire part de ma reconnaissance à Cyrille PACH pour la formation qu'il nous a dispensé durant les trois premières semaines. Trois semaines intenses d'apprentissage qui se sont vu adoucies par les jeux sérieux et la méthode d'apprentissage de Benoît HERMANT.

Benoît HERMANT que je remercie également pour avoir été mon référent au sein de Norsys et pour nous avoir guidé le long de notre projet.

Je tiens également à remercier Thomas DEBLOCK pour la formation Angular 2 qui nous a dispensé. Ainsi qu'à Yoan Lucas, Thibaut Lepage et Laurent Thibault pour l'aide apportée à l'élaboration de ce projet. Je remercie également toute l'équipe Norsys pour son aide et ses conseils tout le long de l'élaboration de ce projet ainsi que pour leur accueil chaleureux.

Pour conclure ces remerciements, je souhaite remercier l'ensemble du corps enseignant du DUT Informatique de Lille 1 pour la qualité de l'enseignement qui m'a été dispensé et qui a contribué à la réalisation de ce projet. Je tiens également à remercier Monsieur Michaël HAUSPIE pour avoir été mon professeur référent tout au long de ce stage.

Table des matières

Abstract	1
Introduction	2
Chapitre 1 Norsys	4
1.1 Norsys Groupe	4
1.1.1 Norsys Région Nord	5
1.1.2 Norsys Paris- Île-de-France	5
1.1.3 Norsys Lyon-Rhônes-Alpes	5
1.1.4 Norsys Côte d’Azur	5
1.1.5 Norsys Nantes	6
1.1.6 Norsys Afrique	6
1.2 Les Fondations Norsys	6
1.3 Norsys et ses valeurs	6
1.4 Ses locaux ou pourquoi Ennevelin	7
1.5 Des moments informels et de convivialité	8
Chapitre 2 La formation Norsys	10
2.1 Formation technique	10
2.1.1 GIT, un « Source Control Manager »	11
2.1.2 Maven, le facilitateur	11
2.1.3 Java	11
2.1.4 Spring Framework et Spring Boot	12
2.1.5 Angular 4	13
2.1.6 InteliJ, un environnement	13
2.1.7 Docker et ses containers	14
2.2 Méthode de projet	15

2.3	Une Web-TV pour projet	16
2.3.1	Présentation du client	16
2.3.2	Mon projet à la loupe	16
2.3.3	Mon équipe	18
2.3.4	Présentation du projet	18
2.3.5	Story Mapping	19
2.4	Problèmes et Solutions	20
2.4.1	Afficher une vidéo	20
2.4.2	Présentation d'une chaîne	21
2.4.3	Appel à l'API	22
2.4.4	Reception des objets	23
2.5	Problème et Solution : Playlist multi-sourcing	24
2.5.1	VideoGular	24
2.5.2	Youtube : Les URL	25
2.5.3	Manipuler les url Youtube	25
2.5.4	API iframe et player Youtube	26
2.5.5	Préserver le contexte	27
2.5.6	Input et Ouput	27
2.5.7	Transmettre des informations entre composant via les URL	29
2.5.8	Dailymotion : Un peu de Back	30
2.5.9	Dailymotion Player	31
2.5.10	Dailymotion-sdk	31
2.5.11	Dailymotion Evenement	32
2.5.12	Quel player ?	32
Conclusion		33
Glossaire		34
Bibliographie		36
Annexe A Annexes		38

Abstract

J'ai passé mes trois mois de stage dans les locaux de Norsys Nord, une entreprise de service du numérique. J'y ai effectué la réalisation d'un projet de Web-TV, notre client Monsieur CASSORET n'est nul autre que le directeur de l'agence Norsys Nord. Sa motivation ou plutôt le besoin auquel il souhaite répondre avec ce projet est assez simple : créer de la communication au sein de l'agence et entre ses collaborateurs. La Web-TV devra être projetée sur des écrans qui seront installés dans la salle à manger de Norsys et diffusera en continu des vidéos réalisé par Norsys, des photos de leurs événements ainsi que sous la vidéo un fil d'actualité. Ma mission a donc été de construire une playlist capable de diffuser à la fois des vidéos du type YouTube ou Dailymotion mais également des images, le tout sans qu'aucune intervention extérieure ne soit nécessaire.

I spent my three months of internship at the premises of a digital services company, Norsys Nord. There, I did a Web-Tv project for our client, Mister CASSORET, who is none other than Norsys Nord's director. His motivation or rather the need that this project deals with is simple : create communication within Norsys and among its collaborators. The Web-Tv will have to be projected on screens which will be installed in Norsys's cafeteria and will broadcast continuously videos made by Norsys and pictures of its events as well as a news feed under the video. My mission was to create a playlist able to broadcast simultaneously videos from both Youtube and Dailymotion, but also pictures. And all this without any external intervention needed.

Introduction

Étudiante en DUT informatique, j'ai eu la chance de profiter d'un enseignement professionnelisant, c'est-à-dire destiné à nous préparer au monde de l'entreprise. J'y ai étudié des domaines divers allant de l'informatique, à la gestion et à la communication. Cette formation pluridisciplinaire est un avantage évident, surtout lorsque l'on s'apprête à entamer son premier stage, toutefois je me suis petit à petit rendu compte d'une chose. Il est difficile d'être préparé à tout, qu'importe la formation que l'on suit.

Lorsque l'on étudie en DUT Informatique on ne cesse de toucher du doigt la réalité de ce que sont les métiers informatiques. Nous sommes formés à des technologies et des méthodes d'entreprise, destinés à nous ouvrir les portes du monde de l'entreprise. Toute fois je suis bien forcée d'avouer que de nombreuses choses semblent bien inconnues lorsque ce que l'on a pris tant de soin à préparer devient une réalité.

De nombreuses choses ne sont pas forcément appréhendables dans un cercle universitaire, où les enjeux sont bien différents. Cette différence est d'autant plus vraie lorsque vous arrivez dans une société telle que Norsys.

Dans le module de recherche de stage, on m'a préparé à passer des entretiens mais rien ne pouvait me préparer à ce que j'allais découvrir en passant les portes de cette société des plus originales.

À Norsys on ne perd pas son temps avec des débats creux, à Norsys on a cherché à voir l'humain derrière le nom apposé sur le CV. Au travers un atelier d'expression, j'ai ressenti leur soucis de me connaître, par leur jeu de programmation leur intérêt pour mes compétences et lors des trois mois de formation, l'importance apportée aux partages et à la pédagogie.

Ce sont toutes ces petites choses qui m'ont donné envie d'effectuer mon stage à Norsys, ça et le projet que j'ai choisi d'effectuer. Une page blanche, une demande et une liberté de modeler ce projet. L'indépendance et l'autogestion qui nous ont été en partie laissées m'ont permis de faire face aux réalités du métier de développeur. Toujours rechercher une solution et lorsqu'on l'a trouvée, en chercher une encore plus efficace, plus optimale.

Apprendre de nouvelles technologies afin de pouvoir donner vie à une envie a été pour moi une opportunité réellement motivante et j'espère pouvoir au fil de ces quelques pages vous faire partager mon enthousiasme.

Mais quel était mon projet me direz-vous, et bien au cours de ces trois mois de stage, ayant débuté le trois avril, j'ai eu à réaliser une Web-TV. Une application web dans laquelle il doit être possible de visionner des médias issus des diverses plateformes utilisées par Norsys. Je vous détaillerai plus en détail les modalités de ce projet et les techniques utilisées dans la suite de mon rapport.

La problématique à laquelle j'ai dû chercher à répondre pendant la majeure partie de ce projet à été, comment pouvoir lire, visionner des médias issus de différente sources (Youtube, Dailymotion, Cloud...) sur une seule et même plateforme ? Comment construire une playlist multi-source ? En effet, il est rapidement apparu que je ne pouvais me contenter d'afficher les médias. Il fallait que je sois capable de détecter la fin pour pouvoir démarrer le média suivant sans soucis de type du média.

Je vais vous faire part des réponses que nous avons trouvé au fil de ce rapport. C'est pour cela que dans un premier temps je vais vous présentez Norsys. Norsys le groupe, ses valeurs, sa convivialité. J'aborderais brièvement les trois semaines de formation, afin de vous présentez les technologies qui ont été le support de mon projet de WebTV.

Un projet ou plutôt une envie de Monsieur Denis CASSORET qui nous a fait confiance pour en faire une réalité. Nous aborderons donc la présentation du projet, les démarches de gestion que nous avons mises en place et ensuite je vous parlerai des problèmes que j'ai pu rencontrer donc ce qui fut mon plus gros défi pour ce projet.

J'exposerai pour chacun de ses problèmes, les solutions qui ont été trouvé et comment cela a influencé l'évolution de mon projet.

1

Norsys

1.1 Norsys Groupe

Norsys est une société de services et d'ingénierie en informatique aujourd'hui appelée entreprise de service du numérique. Elle a été créée en 1994 par un groupe composé d'ingénieurs et de dirigeants possédant une expérience riche dans le domaine du système informatique. Lorsque l'on interroge Monsieur Sylvain BREUZARD sur ce qui l'a poussé à vouloir s'engager dans l'aventure Norsys, ses mots sont clairs mais percutants. Ce que ses employeurs de l'époque proposés ne le satisfaisait pas. Il est tentant de s'engouffrer dans un système basé sur ce que l'on peut appeler « business trop facile . » Nombreux clients ne savent pas de quoi sera fait ce qu'ils demandent, ils cherchent juste à ce que cela fonctionne mais Norsys refuse de se contenter de quelque chose qui fonctionne. C'est pour cela que la société a décidé, depuis 2015, d'investir dans une stratégie consistant à tendre vers le haut de gamme.

Mais qu'entendre par « haut de gamme » ? Fondamentalement, il s'agit d'un état d'esprit. Cet état d'esprit est à l'origine de tout et de lui découle une manière de faire incluant techniques, procédés, moyens et surtout dynamiques humaines. En effet si je dois retenir une chose de la politique de Norsys, si je peux l'appeler ainsi, c'est que l'humain est au centre de leurs préoccupations.

Norsys s'est spécialisée, au fil des années, dans la prise en charge de grands projets intégrant les techniques innovantes, avec une approche globale du système d'Information. Lorsque l'on recherche des informations sur Norsys on trouve régulièrement le terme easymakers. Mais pourquoi easymakers ? Qu'est-ce que cela signifie ?

Ce sont des facilitateurs qui se préoccupent autant des savoir-faire en relations hu-

maines que technologiques ou méthodologiques. Si Norsys est pour ses clients un prestataire de qualité, ce n'est pas uniquement parce que ses prestations sont de qualité. C'est en osant toujours allier, dans un même enthousiasme, la fraîcheur d'esprit et la force de ses réalisations.

Aujourd'hui, l'entreprise est composée de plusieurs structures :

1.1.1 Norsys Région Nord

Il s'agit du siège social de l'entreprise. Situé à Ennevelin, dans la banlieue lilloise, il regroupe une centaine de salariés et coordonne l'activité de toute la société. Nous avons interrogé les dirigeants sur la raison de choisir Ennevelin comme lieu d'implantation car en effet, cela nous paraît bien loin de tout et difficilement accessible par transport en commun. La réponse a été que lorsqu'ils ont acheté, ils ont voulu prioriser l'espace et un prix convenable pour pouvoir se permettre de créer un lieu agréable pour les employés. Effectivement l'espace et le terrain extérieur représentent un cadre des plus agréables pour travailler ou profiter d'une pause bien méritée.

1.1.2 Norsys Paris- Île-de-France

Ouverte en janvier 2001, à l'origine sa création avait pour but de faciliter les contacts avec les clients d'Île-de-France. Cette structure fait désormais partie intégrante de la dynamique Norsys. Aujourd'hui composée de plus de 70 salariés, majoritairement dans le secteur du conseil, ses principaux clients sont dans le secteur de la santé.

1.1.3 Norsys Lyon-Rhônes-Alpes

Ouverte depuis novembre 2004, cette agence joue également un rôle de facilitateur de contact mais cette fois avec les clients de la région de Lyon. À l'origine, elle a été créée exclusivement sur demande d'un client qui voulait que Norsys soit à dix minutes de son entreprise. L'objectif de cette antenne était de progressivement diversifier sa clientèle et cela c'est avéré être un pari réussi.

1.1.4 Norsys Côte d'Azur

C'est à Sophia Antipolis que Norsys ouvre en 2013 les portes d'une nouvelle agence. Cette nouvelle agence a à son tour permis le rapprochement vis-à-vis de ses clients dans

le secteur d'activité de la santé et du social comme le RSI et l'ACOSS.

1.1.5 Norsys Nantes

Dernière arrivée parmi les agences Norsys, cette antenne a été créée en 2015 et ce encore dans le but de se rapprocher de ses clients.

1.1.6 Norsys Afrique

Située à Marrakech au Maroc, cette antenne compte une trentaine de salariés. Elle a pour but principal la sous-traitance d'une partie des développements de Norsys France. Elle abrite également Norsys Fondation créée en 2001 dont l'objectif est le partage de la connaissance en menant des projets à finalités économiques, sociales et sociétales.

1.2 Les Foundations Norsys

Comme je l'ai mentionné en parlant de son engagement à Marrakech, Norsys soutient plusieurs fondations dont Emmaus Connect où j'ai pu aller passer une demi-journée. Dans le processus d'intégration en entreprise, on m'a proposé d'aller passer une journée dans cette association. Emmaus connect a pour objectif de permettre l'accès et l'apprentissage de l'outil informatique à ceux qui n'en ont pas les moyens ou qui n'ont pas les connaissances. Encore une fois on la retrouve dans sa politique d'échange et d'apprentissage. Confrontée à des personnes ne sachant pas, pour certaines, utiliser un clavier d'ordinateur là où cela me paraît le plus naturel au monde, j'ai été émue de cet engagement. À l'heure où l'informatique joue un rôle important dans notre société, je trouve important de permettre à tous d'y avoir accès. De ce fait, je suis reconnaissante à Norsys de m'avoir permis de participer à cette permanence à Emmaus Connect.

1.3 Norsys et ses valeurs

La stratégie d'entreprise de Norsys s'organise autour du terme « performance globale », alliant des objectifs à la fois économiques, sociaux mais également humains. Ce sont ses trois axes qui se rapprochent de la symbolique des noeuds borroméens, symbole de Norsys, et démontre que seul l'union des trois font de Norsys ce qu'elle est.

Cette stratégie a été établie avec les salariés, impliqués à plusieurs niveaux : dans des démarches d'écoute et de veille, des séminaires de créativité et dans des groupes de travail dont l'objectif est d'identifier la pertinence de nouvelles idées et leur faisabilité.

Trois challenges concrétisent cette stratégie :

1- Devenir la plus grande société de conseil et d'ingénierie à taille humaine.

2- Être la société de conseil et d'ingénierie la mieux notée par ses clients.

3- Être la société de conseil et d'ingénierie où l'on est le mieux pour travailler.

Je souhaite d'ailleurs ajouter quelque mot sur ce dernier point.

Monsieur Sylvain BREUZARD, lorsqu'il évoque ses expériences professionnelles antérieures à Norsys déclare qu'il y a des endroits où l'on se sent bien. Des endroits qui nous font sentir que c'est là que l'on veut être. C'est ce qu'il recherche pour ses employés.

Il nous a raconté l'anecdote d'une nouvelle arrivante dans l'entreprise. A son arrivée il lui a proposé de choisir l'endroit où elle souhaitait s'installer pour travailler. N'importe où tant que c'est l'endroit où elle se sentait le mieux et de s'y installer quitte à réorganiser la disposition du bureau. Une soucis du confort de ses employés qui fait de Norsys une société où il fait bon d'aller travailler.

1.4 Ses locaux ou pourquoi Ennevelin

Les locaux de l'agence Norsys Nord ont été construit à Ennevelin alors que toutes les entreprises cherchaient à s'installer en centre-ville à proximité des transports en commun. À ce moment, les dirigeants réalisent un pari, proposer à leurs employés un environnement plus bucolique.

Le calme et l'espace du lieu nous font bien vite oublier le peu de transport en commun desservant la zone.

Ainsi, installé à Ennevelin, les collaborateurs profitent d'un environnement de travail agréable et surtout calme contrairement aux grandes villes.

C'est pour cela que Norsys s'est installé sur un terrain et profite d'un bel espace extérieur où il est agréable d'échanger avec l'équipe Norsys.

Un extérieur également utilisé pour diverses activités telles que l'installation d'un potager ou encore ce que l'on appelle Norsys Campus, qui est un événement ayant pour but de rassembler les collaborateurs dans une ambiance plus festive.

1.5 Des moments informels et de convivialité

Le premier moment informel qu'il me paraît important de nommer est l'entretien de recrutement. Là où l'on se figure le classique entretien privé avec une ou un responsable des ressources humaines, Norsys voit les choses différemment.

Quand je suis arrivée à Norsys, je n'étais pas la seule conviée et le programme n'était pas aux entretiens formels.

Après une présentation de l'entreprise par Norsys, j'ai passé une heure et demie sur une épreuve technique. Au cours de cette épreuve j'ai pu discuter avec deux collaborateurs de l'équipe Norsys.

Leur objectif n'était pas de juger nos compétences mais de se faire une idée de notre capacité à analyser, comprendre et réagir à un problème. Ce fut un moment, non pas stressant mais enrichissant.

Sur un petit concept de concours avec un visuel de notre gain de « points », il nous fallait programmer un ascenseur. Ouvrir les portes, les fermer, laisser entrer les utilisateurs. Le tout en échangeant avec des informaticiens confirmés.

L'heure et demi suivante je l'ai passé en compagnie de Madame Mélanie Wilfard et Monsieur Benoit Hermans. Avec les autres aspirants stagiaires nous avons participé à plusieurs jeux sérieux ayant pour but d'échanger et, de manière détendue, apprendre à nous connaître, cibler ce qui est important pour nous et recueillir notre ressentit sur la journée passée dans l'entreprise.

Ces jeux sérieux sont, comme je l'ai découvert plus tard, une sorte de marque de fabrique de Norsys. Nous les avons utilisés au cours de la formation et notés qu'ils permettent de créer assez facilement une dynamique d'équipe. Comment ?

En rassemblant autour d'un même objectif tout en ôtant le côté sérieux d'un projet à mener à terme. Monsieur Benoit Herman organise d'ailleurs chaque jeudi midi une session de jeu sérieux permettant aux collaborateurs de partager un moment agréable.

On voit ainsi que la vie à Norsys ne se limite pas au travail. On n'a pas d'obligation d'arriver à huit heures pour faire une pause à midi et partir à dix-huit heures.

Cela m'a d'ailleurs un peu désarçonnée au début lorsque l'on m'a dit qu'il m'appartenait de gérer mes horaires étant habituée aux créneaux clairement définis de la vie universitaire.

Évidemment Norsys profite de ses moments constituant la vie d'une entreprise. Les pauses café, les repas entre collègues etc. Mais en dehors de ça, toute occasion semble bonne à prendre pour se réunir et apprendre à mieux connaître les personnes avec qui

on passe nos journées. Il y a les formations organisées par Norsys, elles peuvent être techniques ou axées sur le management. Organisées et réalisés par des collaborateurs de Norsys ou bien par des clients, tous deux peuvent y participer.

Il y a également les soirées telles que celle organisée afin de gagner des fonds pour une association et au cours de laquelle chaque collaborateur a donné de sa personne en faisant le plus de kilomètres possible sur un vélo d'intérieur.

Et enfin, une fois par an, une période nommée le Campus durant laquelle les formations et les soirées s'enchaînent dont deux sportives. À cette occasion, les stagiaires de Norsys forment une équipe afin de participer à un petit tournoi au cours duquel ils affrontent collaborateurs et clients. Un moment convivial grandement attendu au sein de l'agence.

2

La formation Norsys

Afin que nous ne soyons pas directement propulsés dans un projet particulier, utilisant des technologies que nous ne maîtrisons pas, Norsys propose d'intégrer l'ensemble des stagiaires progressivement par le biais d'une formation. Une formation sur les bonnes pratiques de développement et de déroulement de projet.

Cette formation se concentre autour d'un sujet, cette année il s'agissait d'un projet commandé par les pirates de l'agence. Un projet de gestion d'humeur intitulé "MoodTracking".

Le but est de créer un lien entre les collaborateurs dans l'agence et de pouvoir, en fonction des retours, proposer des améliorations toujours dans le souci du bien-être des collaborateurs.

Durant trois semaines nous avons eu une démarche de création de projet. Nous avons rencontré le client, analysé sa demande, réalisé des maquettes et fait des propositions telles que l'intégration d'un espace anecdote de l'agence.

Mais avant de rentrer directement dans le vif du sujet, nous avons participé à une semaine de formation animée par Monsieur Cyrille PACH sur les techniques que nous seront amenés à utiliser. Ces séances étant entrecoupées d'ateliers fonctionnels basés sur des « serious Games » animés par Benoît Hermant.

2.1 Formation technique

La partie technique de notre formation a été assurée par deux collaborateurs de l'équipe Norsys. Ils ont été présent tout au long de ces trois semaines, nous présentant les outils utilisés chez Norsys

Ils ont par la suite été nos référents techniques au cours de la réalisation de notre projet.

2.1.1 GIT, un « Source Control Manager »



Mais GIT qu'est-ce que c'est ?

J'avais déjà été amenée à l'utiliser au cours de mes deux années de formation à l'IUT. Toutefois il a été intéressant de voir l'utilisation qu'en fait une entreprise comme Norsys.

Ce qui a changé dans l'utilisation que je faisais de cet outil c'est avant tout l'utilisation de conventions de nommage.

Finit les petits messages de trois mots que moi seule pouvait comprendre.

Ils nous ont également parlé du processus d'intégration continue, du développement jusqu'au déploiement.

2.1.2 Maven, le facilitateur



Un outil de construction de projets (build) open source développé par la fondation Apache.

Nous avons pu découvrir Maven, un outil qui permet de faciliter et d'automatiser la gestion et la construction d'un projet Java. Le but de Maven est de pouvoir rapidement connaître l'état global du projet.

2.1.3 Java

C'est un langage de programmation orienté objet, c'est à dire un langage de programmation qui vise à faciliter la réutilisation de morceaux de programme déjà écrit dans des circonstances identiques.

Java permet de développer des applications autonomes mais aussi, et surtout, des applications client-serveur.



Ses caractéristiques, ainsi que la richesse de son écosystème et de sa communauté, lui ont permis d'être très largement utilisé pour le développement d'applications de types très disparates. Java est notamment largement utilisé pour le développement d'applications d'entreprises et mobiles.

2.1.4 Spring Framework et Spring Boot

On nous a également présenté ce Framework ainsi que Spring Boot permettant un gain de temps considérable lors du développement. C'est ce que l'on appelle à Norsys « l'auto-magie ». Un système KISS, « Keep It Simple, Stupid ».



Spring est un Framework aux nombreuses fonctionnalités autant sur les aspects web ou de sécurité que d'accès aux données dans le cadre du développement.



Spring Boot est un micro-framework qui a notamment pour but de faciliter la configuration d'un projet Spring et de réduire le temps alloué au démarrage d'un projet. Mais comment ? Nous avons appris qu'un site web (<https://start.spring.io>) permet de générer rapidement la structure d'un projet en y incluant même les dépendances Maven nécessaires à l'application.

2.1.5 Angular 4



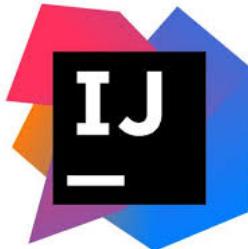
Angular est un framework Javascript, dont l'utilisation est totalement placée côté client. Il s'agit d'un langage interprété. On code en typescript et ce dernier est interprété en Java-Script.

Angular utilise des services et des composants. Afin de créer un service, on utilise la commande « ng g serve /chemin », ici le "g" signifie "generate".

Dans les services on va trouver tous les traitements que l'on effectue sur les données que l'on récupère du back. C'est ici que l'on fait les requêtes au back.

Les composants se divisent en trois parties. L'HTML, le CSS et le ts. C'est le composant.ts qui va faire appel aux services pour récupérer des objets issus de la base de données. Ils sont créés avec la commande suivante "ng g c /chemin", le "g" signifie également "generate" et le "c" signifie "create".

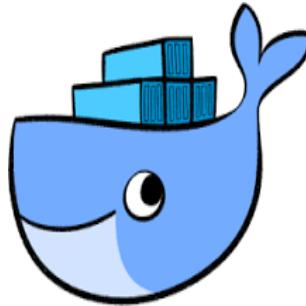
2.1.6 IntelliJ, un environnement



Nous avons utilisé IntelliJ comme environnement de développement. Il permet d'avoir un environnement unique pour le Backend et le Frontend

IntelliJ est un environnement de développement Java commercialisé développé par JetBrains.

2.1.7 Docker et ses containers



Docker permet de créer des environnements (appelées containers) de manière à isoler des applications.

Prenons un cas d'utilisation simple et concret afin d'expliquer l'intérêt de Docker.

Imaginons une entreprise qui souhaite développer une application. L'équipe en charge du développement est composée de deux personnes. Et tous deux n'utilisent pas le même environnement.

En effet, cela pose problème que deux développeurs ne travaillent pas sur les mêmes environnements... Mais avec Docker, on peut faire en sorte que les membres de cette équipe travaillent sur les mêmes versions Linux sans craindre des problèmes de compatibilité entre leurs codes respectifs.

Comment ? Dans ce cas, le plus simple est de mettre en place un Dockerfile, document "chef d'orchestre", qui permettra aux développeurs de monter une image similaire. De ce fait, les développeurs pourront travailler sur un environnement identique.

Grâce à Docker, on peut multiplier les environnements sur sa machine, sans limiter les performances de son ordinateur. Les ressources sont partagées avec la machine hôte ! Chaque environnement peut être configuré simplement grâce à un Dockerfile, présent à la racine.

Cette première semaine de formation a été très éprouvante. Il y avait beaucoup à apprendre mais le principe d'alterner les présentations avec des petites sessions de mise en pratique était très pédagogiques.

De plus nos formateurs ont utilisé le concept de Mob-Programming qui donne un aspect très pédagogique à un exercice de programmation. Il s'agit de session d'une dizaine de minutes où, tour à tour, nous passons devant le pc principal. C'est à dire le pc dont l'écran est projeté. Cela permet d'installer une dynamique de groupe, ainsi qu'une réflexion d'équipe tout en nous familiarisant avec ces nouveaux outils.

2.2 Méthode de projet

Nous avons dans un premier temps échangé avec notre client, ce fut une expérience très intéressante. Le projet nous a été présenté par un membre de l'équipe pirate portant ce projet, Monsieur Gautier GASPARD.

Il avait étudié les humeurs et nous a donc expliqué que ces dernières étaient qualitatives et quantitatives. De ce fait, une personne devait pouvoir choisir une émotion ainsi qu'un degré de motivation. Toutefois nous avons rapidement soulevé un point.

L'application ne pouvait uniquement servir à entrer une humeur chaque matin car cela manquerait d'attractivité pour l'utilisateur.

Nous avons donc fait plusieurs propositions et réalisé des maquettes de ce que pourrait être le rendu final.

Ainsi, l'utilisateur peut désormais voir l'humeur de son agence, consulter en graphique de l'évolution de cette dernière mais également avoir accès à une section anecdote. Un petit espace de partage d'anecdote pour l'agence.

Il était intéressant de réaliser ce projet de cette manière. En effet, nous avons ainsi pu partir d'une page blanche et faire comme bon nous semblait. Pouvoir échanger avec le client, lui proposer nos idées et débattre sur l'apparence et les fonctionnalités de l'application a été un véritable plaisir. Mais au-delà de ça, cela m'a permis d'appréhender l'importance de la phase d'analyse ainsi que de notre rôle. Bien que programmeurs de l'application, nous avons également été une force de proposition, en nous investissant dans ce projet nous avons tenu à être un générateur d'idées.

Toujours dans la méthodologie de projet, nous avons dans un premier temps défini les User-Story, dont l'écriture se base sur le template suivant.

En tant que <Rôle> que je <Action> alors <Résultat>.

Cela nous a permis d'avoir une vision commune du produit, en utilisant un langage comprehensible par tous.

Nous avons par la suite effectué un priorisation des User-Story. En nous basant sur une technique que j'avais déjà pu pratiqué à l'IUT au cours de différents projets, soit : *To Do - Doing - Done*. Une méthode AGILE qui permet de voir en continue l'évolution, l'avancée du projet à chaque instant.

Il est ainsi plus aisément de repérer les points difficiles et d'établir une date de démonstration. En effet, nous avions comme mission de rendre une version backend fonctionnelle pour une démonstration au client à la fin de la seconde semaine.

Nous avons d'ailleurs respecté les délais et le client s'est montré satisfait de cette première version.

La dernière semaine, bien plus brève s'est consacrée à la réalisation du front, soit du visuel de l'application. Pour cela nous avons utilisé Angular, ce qui a été une grande découverte.

Ces trois intenses semaines se sont conclues sur une démonstration de notre application à plusieurs membres de l'agence dans la confidence.

Car en effet, j'ai omis de le dire, mais il s'agit d'un projet secret qui s'est vu révélé plus tard au cours d'une réunion d'agence à Marrakech.

Il était très intéressant de pouvoir suivre l'évolution de la création d'un projet du début à la fin. Partir d'une idée, la rendre petit à petit plus concrète et pouvoir ensuite la proposer à un client. Le tout en faisant preuve de créativité et de prise d'initiative.

C'est donc sur une note pleine d'optimisme que nous avons pu rejoindre la plateforme Norsys et nous installer à ce qui serait notre espace de travail pour les deux mois suivants.

2.3 Une Web-TV pour projet

2.3.1 Présentation du client

Chaque projet naît d'une envie, d'une demande d'un client et ce projet de Web-TV ne fait pas exception. Pour ce projet, notre client n'est nul autre que Norsys.

En effet, le projet sur lequel je travaille depuis un bon nombre de semaines maintenant est un projet interne à l'entreprise. Ce projet est né d'une envie de Monsieur CASSORET qui souhaite par ce geste créer un lien entre les employés.

De plus, ce serait également un moyen d'animer le réfectoire de l'entreprise, en créant une ambiance conviviale propice à l'échange.

2.3.2 Mon projet à la loupe

Tout projet que l'on met en œuvre veut répondre à un besoin, ici il s'agit de créer un projet de Web-TV mais pour répondre à quel besoin ? Une Web-TV, un besoin ?

Norsys organise beaucoup d'événements et tourne de nombreuses petites vidéos. Par exemple, chaque année, on demande aux stagiaires de tourner une petite vidéo sur un sujet précis. Une fois cela fait ladite vidéo est mise en ligne, mais là commence le problème. En effet les médias de Norsys se trouvent souvent hébergés sur différentes plateformes de

diffusion. Vous trouverez aisément une chaîne Norsys sur Dailymotion et une autre sur Youtube.

Le besoin auquel on nous a donc demandé de répondre est de rassembler ces sources sur une seule et même application.

Toutefois à cette demande s'ajoute également le souhait d'avoir accès à des médias de type non pas vidéo mais image.

En effet, de nombreuses photos sont prises au cours des évènements mais les collaborateurs n'ont pas toujours l'opportunité de les voir. De ce fait, on se propose de les intégrer dans notre application.

Également dans le but de rendre cette application à la fois attractive et utile, les collaborateurs auront également accès aux « news » de l'entreprise. Des petits messages sur les nouveautés en lien avec l'entreprise qui défileront sous la vidéo. Un peu comme on peut le voir sur certaines chaînes d'information. Un fil d'actualité en somme. Notre mission est donc de créer une plateforme où l'on pourra visionner à la fois des vidéos Youtube et Dailymotion, mais également des images tout en ayant accès aux nouvelles de l'entreprise.

Afin de réaliser ce projet, nous avons, dans un premier temps mis en place un front dépourvu de lien avec une base de données.

En effet, la valeur principale de notre projet est de pouvoir visionner des vidéos et pour cela une base de donnée n'est pas primordiale. C'est pour quoi j'ai simulais la réception de JSON, normalement émis par le back, en les inscrivant en dur afin de pouvoir simuler le comportement de l'application de la manière la plus correcte possible.

Nous avons alors constaté que certaines parties seraient plus aisées que d'autres à réaliser.

C'est en nous basant sur les conventions en usage sur les différentes plateformes multimédias que nous avons créé notre application. Comme toute bonne application web, elle se compose de différentes pages. Nous avons la page d'accueil répondant à la charte Norsys dans son design et sur laquelle on peut voir différentes informations. Une vidéo vitrine, la dernière vidéo ajoutée, la liste de chaînes etc... Ainsi lorsqu'un collaborateur se rendra sur cette page il comprendra plus aisément le sujet et le but de cette application. Grâce à un menu on peut accéder à différentes fonctionnalités. Notamment celle permettant d'accéder aux chaînes. De là il est également possible d'obtenir le détail de ces chaînes afin de visionner les vidéos de cette dernière de manière unitaire mais également d'accéder à une description de la chaîne . C'est de là que l'on peut déclencher le mode « web-TV ».

Un mode Web-TV qui s'est rapidement avéré être le point le plus complexe de ce projet.

2.3.3 Mon équipe

Pour réaliser ce projet j'ai travaillé avec Clément Lambert et Clément Cantrainne, deux autres stagiaires étudiants en licence professionnelle Da2I.

Ce qui fut intéressant et surtout très enrichissant avec ce projet c'est qu'étant laissé en autonomie, il nous a appartenu d'établir notre mode de fonctionnement.

Benoit Hermant, notre référent dans ce projet, a mis en place un tableau *To/Doing/-Done* qu'il nous a appartenu de tenir à jour.

Et à l'aide de debrief régulier, nous lui retracions notre avancée.

Au cours de ces trois mois nous avons pu nous investir dans chaque aspect du projet. Qu'il soit question de front ou de back. Toutefois lorsqu'il a fallu se diviser les tâches, la division a été assez simple et instinctive.

En effet au cours de notre projet de formation, j'avais commencé à toucher au front et à utiliser Angular. Contrairement à Clément Lambert qui n'avait pas eu cette occasion.

C'est donc ainsi que j'ai commencé à coder les services du front et tout ce qui touche aux vidéos alors que Clément Cantrainne se chargeait du design et des news et que Clément Lambert travaillait sur le back.

C'est ainsi que notre équipe de stagiaire a pu débuter sur le projet Web-Tv avec pour chef de projet Benoit Hermant.

Toute fois nous avons également reçu l'aide des collaborateurs de Norsys, notamment Thibaut Lepage, qui m'a beaucoup aidé sur une partie difficile constituant à faire communiquer un composant père et fils lors de l'élaboration de la playlist, ainsi que Yoan Lucas qui a partagé notre espace de travail pendant les deux mois.

2.3.4 Présentation du projet

Comme chaque projet a un point de départ, le nôtre s'est tenu dans la petite cafétéria de Norsys en compagnie de Benoît Hermant.

Nous avions déjà eu vent du sujet de notre projet car nous avions eu l'opportunité de le choisir, mais l'image que l'on se fait d'un projet est parfois différente de celle du client.

Nous avons donc écouté Benoît nous présenter la demande du client, à renfort de story mapping. L'objectif est alors devenu plus clair.

La demande principale est une Web-TV, le but est donc de pouvoir projeter sur une télé

un player jouant sans interruption, ou intervention de quiconque, des médias de diverses sources.

Mais quelles sont les sources et qu'entend-on par média ?

Les sources font référence aux plateformes que Norsys utilise pour diffuser ses médias, cela peut être Youtube, Dailymotion ou Cloud. C'est d'ailleurs sur ces trois supports de diffusion que nous avons axé nos efforts.

Un média qu'est-ce que c'est ?

Ici un média peut être une vidéo ou une image ou une new. Il nous fallait donc projeter notre média avec en dessous notre fil d'actualité. Toutefois au-delà de la Web-TV il faut également que l'application soit quelque chose d'agréable lors de l'utilisation. En nous basant donc sur des plateformes usuelles de diffusion nous avons structuré notre application en différentes pages. Page d'accueil, page sur les différentes chaînes, pages des médias correspondant à cette chaîne etc...

Nous nous sommes rapidement mis au travail avec comme référence une maquette faite par Benoît Hermant.

2.3.5 Story Mapping

Afin de définir nos tâches nous nous sommes basés sur le Story Mapping, définissant ainsi les priorités.

Je me suis rapidement lancé sur le Web-TV et ce qui touche aux vidéos. Au début nous avions des valeurs en dur stockées dans des variables sous forme de JSON n'ayant pas de back-end fonctionnel.

J'avais donc défini un média comme un objet ayant uniquement un identifiant. Cette définition a ensuite évolué en fonction des besoins rencontrés et surtout des demandes du client.

En effet, l'idée que l'on se faisait au départ du média s'est complexifiée tant par la demande du client que par l'utilisation que nous avions besoin d'en faire.

Désormais notre média contient plus d'informations telles qu'une description, un type...

On a ainsi pu voir qu'un projet ne cesse pas d'évoluer sous prétexte qu'il a commencé, que la demande a été formulée.

Un projet évolue tout au long de sa conception et on peut être amené à devoir ajouter, ou supprimer des choses, en fonction du retour du client ou parce qu'on se rend compte que ce que l'on avait imaginé n'est pas la meilleure option.

Tous ces changements il a fallu les prendre en compte et parfois recommencer. Je ne

mentirais pas en disant que cela n'avait pas quelque chose de frustrant par bien des égards mais c'est également extrêmement intéressant car on apprend à ne pas se contenter de faire quelque chose qui marche. On apprend à faire quelque chose de bien.

2.4 Problèmes et Solutions

Au début du projet, on nous a demandé de nous focaliser sur la réalisation du visuel de l'application, c'est-à-dire le front.

Le choix de la technique que nous avons utilisées à été assez simple. En effet, on nous avait présentés Angular au cours de notre formation et ayant remarqués les nombreux avantages que confère sa structure en composant nous l'avons choisi.

Mais afin de pouvoir commencer à afficher, utiliser des vidéos il nous fallait simuler une réception de données correspondant à ce que nous voulions afficher et ce sans appel à une base de données.

Pour cela nous avons utilisé des JSON. J'ai créé plusieurs JSON composés des éléments dont j'avais besoin, par exemple un média.

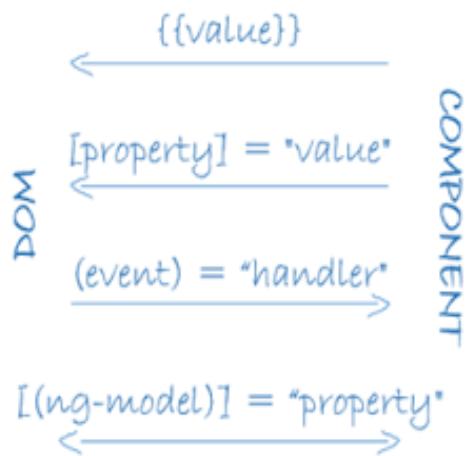
Cette manière de simuler la réception d'un objet par le front permet de rendre notre service indépendant du back qu'il est supposé appeler. Ainsi il est possible d'avancer sans être limité par l'avancée, ou l'absence d'un back.

Une fois mes objets et mes services construits j'ai pu passer à leur affichage, à leur traitement mais plusieurs problèmes ce sont posés.

2.4.1 Afficher une vidéo

Cette tâche semble en apparence simple. On utilise une balise iframe qui permet d'afficher un média quel que soit son type. Image, vidéo Youtube ou Dailymotion, en somme exactement ce dont nous avions besoin pour notre tâche. Simple à utiliser si l'on entre la source du média en dur dans l'HTML, mais un peu plus complexe si on souhaite utiliser une valeur récupérée par un service.

C'est le fichier .ts qui se charge de récupérer les informations transformées par le service. Une fois dans ce dernier elles peuvent être transmises à l'HTML. De manière plus générale ces deux fichiers peuvent partager des informations. Afin de faire communiquer l'HTML et le composant on peut utiliser plusieurs notations.



L’élément : valeur permet de donner au composant HTML une valeur contenue dans le composant .ts, toutefois, lorsque j’ai voulu insérer dans mon iframe l’URL de ma vidéo je suis tombé sur une erreur.

En effet, lorsqu’il s’agit d’un paramètre d’un élément HTML, tel que la source d’une iframe, l’écriture est alors légèrement différente de ce à quoi on s’attend.

```
<div *ngIf="firstMedia !== undefined" class="embed-responsive embed-responsive-16by9 space">
| <iframe [src]='firstMedia' frameborder="0" allowfullscreen></iframe>
</div>
```

Cela a été la première difficulté rencontrée avec l’affichage des vidéos mais pas la dernière.

Toutefois, pouvoir ainsi faire passer des informations entre les différents éléments d’un composant est extrêmement utile. De plus, la méthode est la même pour tous les types de médias.

2.4.2 Présentation d’une chaîne

À ce stade du projet, j’étais capable d’afficher toute une liste de vidéo contenu dans une chaîne précise.

Toutefois les afficher c’est bien mais on ne peut pas les afficher n’importe comment. Il fallait se conformer à la maquette du projet. Or, je n’étais alors pas capable d’afficher les vidéos à condition de les afficher toutes.

Mais ce n’est pas ce que nous voulions faire dans tous les cas. En effet, lorsque nous affichons la liste des chaînes nous affichons également ses vidéos. Toutefois, vous conviendrez qu’afficher une série de vidéos sans la structurer n’est pas très agréable à regarder,

nous avons donc décidé de limiter le nombre de vidéos affichées sur cette page à quatre par chaîne.

J'ai donc entrepris de construire une boucle afin de limiter l'affichage à quatre vidéos pour chaque chaîne. Cette boucle, je l'ai placée dans mon composant HTML. Il est en effet possible de faire des boucles en HTML à l'aide du ”*ngFor” qui va parcourir un objet. Toutefois, je me suis rendu compte qu'une simple boucle ne pouvait pas suffire à ce que je voulais faire. En effet, je n'arrivais pas à afficher quatre vidéos différentes.

J'ai donc construit une Map avec pour clé la chaîne de la vidéo et en valeur une liste de vidéo.

Afin de pouvoir la parcourir en HTML j'ai construis une double boucle. Après plusieurs essais j'ai finalement réussi à afficher uniquement quatre vidéos pour chaque chaîne.

Mais que ce passe-t-il si ma chaîne ne dispose que de deux ou trois vidéos ? Et bien dans ce cas-là j'affiche une image blanche de vidéo non trouvée. Pas très agréable pour l'utilisateur. Le mieux est donc de ne rien afficher dans ce cas-là. De ce fait, lorsque la vidéo est inexistante je transmets un média vide.

À ce stade je peux décider d'afficher une image ou non grâce au ”*ngIf” qui fonctionne comme un ”if” classique en Java. On vérifie une condition, c'est-à-dire ici, si la valeur que l'on va passer à l'iframe est vide ou non.

Cet épisode nous a permis de nous rendre compte que traiter la gestion de l'affichage était assez complexe sur ce genre d'objet. En effet il nous fallait par moments n'afficher qu'une vidéo, à d'autres moments quatre etc...

C'est pour cela que j'ai demandé à Clément Lambert de créer une fonctionnalité à laquelle on spécifie le nombre de vidéos que l'on souhaite récupérer.

Un avantage de l'autonomie dont j'ai profité au cours de ce stage est que j'ai pu tirer mes propres conclusions et chercher des améliorations à nos fonctionnalités.

2.4.3 Appel à l'API

À ce moment-là, notre front était fonctionnel mais ne faisait pas appel à l'API et donc à la base de données. J'ai donc entrepris de créer ce lien côté front.

Côté back, Clément Lambert a créé un Docker nous permettant d'obtenir un port pour accéder à l'API.

Côté front, on doit placer ce port dans le fichier ”environnement.ts” qui était jusqu'alors le port 8080, soit en local.

Côté service, on fait donc appel à l'API avec des requêtes de type ”http” afin de

récupérer les données dont nous avons besoin. Il est important de faire attention au chemin qui a été déclaré côté API et au paramètre demandé car sinon on n'obtient pas du tout ce que lon veut.

Les chemins sont spécifiés côté back et côté front on définit les bases stables telles que l'appel à tel ou tel service dans des variables globales alors que les bases variables se définissent dans les appels http de même que les paramètres.

The screenshot shows two code editor windows. The top window displays the `environment.ts` file:

```

1  export const environment = {
2    production: false,
3    api : {baseUrl : 'http://localhost:32789/api'}
4  };
5

```

The bottom window displays the `MediaPlayerService.ts` file:

```

const API_URL = environment.api.baseUrl;

@Injectable()
export class MediaPlayerService {
  private MEDIA_URL = `${API_URL}/media`;
  private CHANNEL_URL = `${API_URL}/channel`;
}

```

Une fois cette liaison crée j'ai voulu tester le résultat et je suis tombée sur une erreur de CORS, c'est-à-dire **Cross-origin resource sharing**. CORS définit une façon dont un navigateur et le serveur peuvent interagir pour déterminer si oui ou non il est sûr de permettre la demande.

Ici notre navigateur refusait de faire la demande. Il a donc fallu y remédier. J'ai donc entamer une phase de recherche jusqu'à trouver une solution.

Afin de résoudre ce problème, j'ai ajouté dans les contrôleurs de notre API l'annotation `@CrossOrigin`. Au départ je ne l'avais placé que sur une méthode du contrôleur, de ce fait, j'ai rencontré ce problème sur mes autres services. Afin de le résoudre de manière globale on place cette annotation sur la classes du contrôleur et non pas sur les méthodes la composant. Ainsi l'interaction entre le serveur et le navigateur a pu être autorisée.

2.4.4 Reception des objets

J'avais désormais accès aux données de l'API et donc au JSON généré par cette dernière. De ce fait, je n'avais plus besoin d'utiliser mes propres JSON, je pouvais désormais utiliser ce transmis par l'API.

Toutefois je me suis rendu compte que le format de mes objets différait de celui de l'API. Cela a donné lieu à une grosse modification du code et à une évolution des objets.

En effet, côté API on faisait appel aux API YouTube et Dailymotion. Grâce à ces API, on peut récupérer des objets. Pour YouTube, il s'agit de Spinet dans lesquels on trouve toutes les informations dont on a besoin. C'est à partir de ces Spinet que l'objet Media a été construit. Il dispose donc de plus de paramètres que l'objet Media que j'utilisais jusqu'alors.

En effet, les demandes de notre client avait légèrement changé. Par exemple, il souhaitait que chaque vidéo dispose d'une description. Il a donc fallu redéfinir le modèle de notre objet Media côté front.

Il est donc important de faire attention à la structure que l'on donne à un objet et de s'assurer que cette structure est la même côté front et API.

Rappelons-le, le but du back était d'uniformiser cette structure pour que l'utilisation des médias soit simplifiée côté front. C'est pour cela que je dois pouvoir réceptionner des objets me permettant de construire mes objets Media sans avoir à effectuer de transformation particulière.

Afin de toujours connaître la forme du JSON que j'allais recevoir, j'ai pris l'habitude d'utiliser l'outil Postman, un plugin Google qui permet de simuler des appels à l'API. Ainsi on peut également voir quels sont les retours de nos requêtes.

2.5 Problème et Solution : Playlist multi-sourcing

2.5.1 VideoGular

Tout d'abord j'ai voulu mettre en place l'outil Videogular, un framework d'angularjs pour les vidéos.

Avec ce framework, on doit pouvoir jouer plusieurs formats de vidéos tout en ayant connaissance de l'état de la vidéo. Ce framework semblait donc répondre à notre besoin.

Toutefois nous n'arrivions pas à le mettre en place malgré les imports et les exemples que nous utilisions. Il semble y avoir un problème de compatibilité entre le framework et l'utilisation que nous voulions en faire.

Ce fut assez décevant de se rendre compte que nous ne pouvions pas l'utiliser mais cela nous a poussé à entreprendre plus de recherches. Lorsque nous avons commencé notre projet, on nous a dit que la phase de recherche serait une phase primordiale à notre réussite, que c'était une phase importante du métier de développeur et avec ce projet j'en ai pleinement pris conscience.

2.5.2 Youtube : Les URL

Suite à la déconvenue que nous avons rencontrée avec Angular, j'ai décidé d'utiliser les API des plateformes tels que Youtube et Dailymotion.

Mais avant de vous présenter l'API Youtube, je pense qu'il est important d'éclairer un point.

Lorsque l'on affiche un élément dans une iframe, il suffit en général de placer son URL dans la source de l'iframe. Pas avec une vidéo, Youtube. En effet, l'URL nécessite une petite transformation.

Par défaut une URL Youtube ressemble à ceci :



A screenshot of a web browser showing a secure YouTube video page. The address bar displays a green padlock icon followed by the text "Sécurisé | https://www.youtube.com/watch?v=JGwWNGJdvx8".

Or ces URL ne peuvent pas être visionnés en dehors de la plateforme Youtube. Pour pouvoir la lire, il nous a fallu créer une petite fonction qui va créer une URL que nous pourrions visionner avec notre iframe. C'est-à-dire une URL de la forme suivante.



```
<iframe width="560" height="315" src="https://www.youtube.com/embed/JGwWNGJdvx8" frameborder="0" allowfullscreen></iframe>
```

2.5.3 Manipuler les url Youtube

Une fois le problème de lecture de vidéo Youtube réglé j'ai pu me concentrer sur l'utilisation de l'API. Il est assez aisément de construire une playlist Youtube avec l'API, il ne s'agit en effet que d'un simple jeu d'URL.

Si vous prenez une base d'URL Youtube comme celle que nous avons vue précédemment, vous pouvez derrière ajouter diverses options telles que l'option playlist par exemple et lister les identifiants des vidéos composant la playlist dans cette dernière.

L'identifiant d'une vidéo Youtube est cette petite partie de l'URL.



```
src="https://www.youtube.com/embed/JGwWNGJdvx8"
```

Il est également possible d'ajouter d'autres options telles que l'auto-player, ou alors faire disparaître le logo Youtube de la barre de lecture, ou la barre elle-même. Et ceux en manipulant simplement les URL Youtube.

Toutefois si on essaie d'y ajouter des identifiants Dailymotion cela ne marche plus. Cette fonctionnalité de l'API était également insuffisante dans le sens où elle ne pouvait pas nous transmettre l'état de la vidéo, si elle est en cours, terminée ou autre. Il fallait donc utiliser l'API de manière plus complète.

2.5.4 API iframe et player Youtube

L'API Youtube dispose d'une partie API Iframe qui permet de créer un player. Pour se faire on doit déclarer une div dans notre HTML, c'est dans cette div que l'API va pouvoir créer son player, son iframe.

Un player se construit de sorte à avoir un identifiant de vidéo à jouer, des options identiques à celles que j'ai mentionnées plus haut me permettant de spécifier que la vidéo devait être en auto-play.

Ainsi la vidéo se lance sans qu'aucune intervention extérieure ne soit demandée, ce qui était une exigence de notre client. Mais à ce player je peux également ajouter des listeners, des événements définis par l'API, tels qu'une fonction permettant de détecter les changements d'état de la vidéo.

```
setPlayer() {
  this.player = new YT.Player('player', {
    videoId: this.media,
    // , controls: 0},
    playerVars: {autoplay: 1},
    events: {
      'onStateChange': (event) => this.zone.run(() => this.onStateChange(event))
    }
  });
}
```

Une fois l'API trouvé, il a été assez aisément de la mettre en place pour lire une vidéo, mais nous ne voulions pas lire une seule vidéo mais bien plusieurs.

C'est à ce moment-là que je me suis rendu compte que tout ne se trouve pas toujours dans la documentation et qu'il était important de bien comprendre l'outil que nous utilisions.

Il a été assez difficile de mettre en place un player Youtube fonctionnel et correct. Dans un premier temps j'ai réussi à réaliser un player Youtube, mais malheureusement il s'est avéré que mon player utilisait non seulement le wrapper angular mais également l'API, ce qui à terme pouvait causer des problèmes. Il a donc fallu faire en sorte de n'utiliser que l'API Youtube.

Pour pouvoir utiliser l'API il faut l'initialiser, c'est-à-dire utiliser la méthode "onYouTubeIframeAPIReady". Cette fonction va charger l'API pour que nous puissions l'utiliser. Toutefois il faut prendre garde à ne pas la charger à chaque fois que nous appelons le composant. Pour cela on vérifie simplement qu'elle existe.

```
ngOnInit() {  
  if (window['onYouTubeIframeAPIReady'] === undefined) {  
    window['onYouTubeIframeAPIReady'] = () => {  
      this.setPlayer();  
    };  
  } else {  
    this.setPlayer();  
  }  
}
```

2.5.5 Préserver le contexte

Vous avez peut-être remarqué sur ce morceau de code, les arrow function. Ces fonctions nous permettent de conserver le contexte du composant. En effet à plusieurs reprises nous avons rencontré l'erreur « this is undefined » ce qui pour nous n'a pas de sens. Et bien à force de contracter des appels internes dans le composant, celui-ci perd le contexte et devient indéfini. Les arrow function empêchent cela de se produire.

Pour pouvoir utiliser l'API, il est également important d'utiliser le script de l'API ainsi que d'installer le composant Youtube.

Avec Angular cela se fait à l'aide de la commande "npm install package". Sans quoi certaines fonctionnalités de l'API ne peuvent être utilisées.

2.5.6 Input et Ouput

Afin de pouvoir changer la vidéo Youtube, car à ce stade nous ne jouons que des vidéos Youtube, nous utilisons la fonction "onStateChange" qui va émettre un évènement à la fin de la vidéo Youtube.

Comment ça émettre ? Une autre partie intéressante de la playlist multi-source était de faire communiquer le composant père avec son fils afin de pouvoir recevoir l'identifiant de la vidéo que je souhaite jouer.

Nous avons rapidement structuré la playlist en plusieurs composants. Le composant père était la web-TV dans son intégralité, c'est-à-dire avec les vidéos et les news. Et les composants fils tout ce qui la compose. C'est comme si le composant père était une bâtie et que les composants fils en étaient les différentes pièces. Ici elles sont les différentes fonctionnalités du composant père.

Toutefois il faut être capable de faire communiquer le père et ses fils.

Pour cela ont utilisé les **@Input** et **@Output**. Dans le composant fils on place ces deux annotations suivent du nom de variable.

```
@Output() endOfVideo: EventEmitter<boolean> = new EventEmitter<boolean>();
@Input() media;
```

L'input permet de récupérer la valeur d'une variable du composant père, en créant dans sa balise HTML une émission de variable de la même manière qu'on le ferait pour faire communiquer un ts et un HTML d'un même composant. L'output permet lui, au composant fils de communiquer avec le composant père en émettant un événement.

Ici on utilise l'input pour faire parvenir au composant fils l'identifiant de la vidéo qu'il a besoin de jouer.

Pour ce qui est de l'output, lorsque la fin de la vidéo est détectée par le joueur Youtube. Notre output va émettre un évènement qui va déclencher le passage à la vidéo suivante.

Cet évènement est capté par le père et déclenche une méthode de ce dernier.

```
<app-youtube-player *ngSwitchCase="youtube" [media]="currentURL" (endOfVideo)="nextVideo($event)">
  </app-youtube-player>
```

Mais même avec cela nous étions encore incapables de jouer la vidéo Youtube suivante. C'est alors que la fonction "ngZone" est entrée en scène.

Notre composant existe dans une boîte où l'on appelle Angular, tant que ces méthodes sont appelées par quelque chose se trouvant dans le composant tout se passe bien mais dans notre cas, la méthode qui alerte le composant parent n'est pas dans notre boîte angular. Cela est sans doute lié à l'iframe qui est généré par l'API Youtube. De ce fait, il a fallu faire savoir à Angular que le code doit s'exécuter dans la boîte. D'où "ngZone" qui va forcer l'exécution dans la boîte Angular.

Notre player Youtube fonctionne donc désormais parfaitement mais ce n'est pas un simple player Youtube que l'on désire. C'est un player multi-source, d'où l'utilisation de l'API Dailymotion.

2.5.7 Transmettre des informations entre composant via les URL

Avant de présenter l'API Dailymotion, je souhaite aborder un autre problème de transmission d'information.

Lorsque je me trouve dans la section viewChannel, qui liste mes channel, je peux cliquer afin de voir le contenu de la channel. Je fais alors appel au composant Channel, mais comment dire à ce composant qu'il doit récupérer les informations de tel ou tel channel. Comment peut-il savoir qu'elle channel j'ai sélectionné ?

Sur de nombreux sites on peut voir des informations passer dans les URL au fur et à mesure que l'on navigue entre les pages. J'ai donc entrepris de suivre la même idée en faisant passer l'identifiant de mon channel comme un paramètre de mon URL. Mais comment faire ?

J'ai rapidement trouvé qu'il fallait que j'utilise un Router contenu dans la librairie Angular du même nom.

```
async clickedChannel (channel: string, idVideoChannel: string) {  
  this.route.navigate(['/channel', channel, idVideoChannel]);  
}
```

Mais alors que faire passer des paramètres semble dans un premier temps facile cela s'est avéré un peu plus complexe. En effet j'ai dans un premier temps passé un paramètre au moment de la redirection, ainsi j'ajoutais bien un paramètre à mon URL. Problème cela ne semblait correspondre à aucune URL connu par mon navigateur.

Si on regarde plus attentivement le fichier de routing, c'est à dire le fichier où l'on va définir les chemins de chaque composant, on remarque qu'il faut spécifier lorsque l'on veut donner un paramètre à un chemin.

En effet un chemin possède deux partie, la référence au composant, le nom de la page et les paramètres de la page. J'ai donc pris soin de spécifié à ce fichier que la page Channel possède un paramètre correspondant à l'identifiant de la Channel qu'elle doit afficher.

Il faut donc que mon composant channel puisse récupérer l'identifiant de ma chaîne afin de pouvoir la lancer. Afin de le récupérer, on utilise également le Router mais cette fois pour récupérer un paramètre. Ici il faut bien faire attention à spécifier le même nom

que celui qu'on a inscrit dans le fichier de Routing sous peine de ne pas du tout récupérer ce dont on a besoin.



Mon composant Channel récupère désormais les éléments correspondant à la Channel que je lui ai demandé d'afficher.

Voir les différentes manière de faire communiquer les composants, que ce soit avec des URL ou avec les Input/Ouput m'a permis d'améliorer la manière dont j'écrivais mon code.

En effet avant de découvrir ces deux manières de faire, j'avais pour réflexe de tout faire dans un seul composant à partir du moment où il devait y avoir échange de données, or, le code devenait très vite illisible. En découplant bien les différentes fonctionnalité sans que cela m'empêche d'avoir les bonnes informations j'obtiens un code bien plus agréable à lire et de ce fait de meilleur qualité.

2.5.8 Dailymotion : Un peu de Back

Pour cette partie je me suis occupé de l'appel à l'API côté back également.

L'appel à l'API est bien plus simple que pour l'API YouTube qui nécessite des authentification.

Avec l'API Dailymotion on dispose d'une URL propre à l'API et ces à cette url que l'on spécifie ce que l'on a besoin. Dans notre cas on souhaite avoir des informations sur une vidéo ayant tel identifiant et récupérer les champs d'identifiant, de titre, de description et l'image de la vidéo.

Une fois cette URL construite on peut faire un get et on obtient alors un JSON composé des éléments que nous avons spécifié dans l'url. A partir de cela on a pu construire l'object media de source Dailymotion. Le champ source est d'ailleurs un champ que nous avons dû rajouter afin de permettre une uniformisation des données reçues. Cela permet de ne pas avoir à créer un objet Youtube, un objet Dailymotion ou un objet image.

On uniformise les données, cela est l'intérêt de gérer les API côté back plutôt que côté front.

2.5.9 Dailymotion Player

Mon composant a été construit de la même manière que le composant Youtube afin de pouvoir communiquer avec le composant père. C'est-à-dire que j'y ai placé des Input et des Ouput.

On retrouve de nombreuses similitudes dans son utilisation et sa déclaration avec le player Youtube.

En effet, pour créer le player on déclare une div en HTML, cette div sera également remplacée par le player Dailymotion. Le Dmplayer qui m'a d'ailleurs causé plusieurs difficultés comme je vous l'expliquerai un peu plus loin.

Lorsque l'on crée le player on lui spécifie un identifiant ainsi que des options. Des options un peu plus restreintes que celle de Youtube mais proposant également l'option auto-player.

Mais comme pour Youtube des imports sont nécessaires ainsi que l'ajout d'un script contenant les informations liées au player.

Afin de suivre l'évolution de l'état du player on utilise "ngOnChange", une fonction d'Angular qui permet de détecter les changements se déroulant dans le composant.

Il y a en réalité deux différences majeures. La gestion de l'évènement de fin ainsi que l'utilisation du sdk.

2.5.10 Dailymotion-sdk

Afin de pouvoir utiliser le player Dailymotion j'ai installé le sdk Dailymotion, toutefois après l'avoir importé tout ne s'est pas passé exactement comme prévu. En effet Angular était incapable de connaître DM mais pourquoi ?

À mes yeux cela n'avait pas de sens car la documentation était explicitée. Pour utiliser le Dmplayer il faut avoir le sdk, or je l'avais mais il faut s'interroger sur la nature du sdk.

Nous nous sommes rendu compte que dailymotion-sdk était une librairie fonctionnant en exécution sur un serveur node et pas depuis un navigateur client. De ce fait DM n'existe pas pour Angular, typescript n'arrivait pas à la reconnaître.

En fait, il n'existe pas de variable définie pour l'objet DM fournis par l'API contrairement à Youtube. Il m'appartient donc de créer cette variable.

Pour remédier à cela j'ai déclaré une constance DM de type any afin que typescript sache que cette variable existe.

Une fois DM déclaré, j'ai pu construire mon player et comme je vous l'expliquais un

peu plus haut ce dernier se construit de la même manière que Youtube.

C'est donc plus forts de l'expérience de l'API Youtube que nous avons pu se prémunir de certains problèmes tels que l'initialisation multiple du player où la conservation du contexte.

2.5.11 Dailymotion Evenement

Avec Dailymotion on n'intègre pas les évènement dans la player. On doit déclarer des listeners.

Ce listener écoute des évènements définis par l'API Youtobeus. Nous souhaitons ici encore et toujours savoir quand la vidéo se termine pour pouvoir émettre un évènement.

Pour écouter la fin de la vidéo c'est un listener de fin de vidéo qui doit être utilisé. On utilise évidemment l'arrow fonction lorsque l'on utilise ce listener afin de préserver le contexte.

2.5.12 Quel player ?

Nous avions désormais deux players fonctionnels mais chaque player correspond à un type de vidéo bien précise. Il me faut donc pouvoir dire à mon composant père quel composant utiliser et ce dans quelles circonstances.

Pour cela, on peut utiliser un ”*ngSwitch” qui proposera trois ”ngCase”, soit les trois sources possibles du média. Toutefois il semblerait que déclarer le type directement dans le ”ngCase” ne fonctionne pas. Afin de remédier à ce problème, j'ai déclaré des variables dans mon .ts que je transmets à mon HTML afin de pouvoir gérer les trois médias et donc utiliser le player approprié.

Désormais que tous ces problèmes ont été réglés notre Web-TV dispose d'un player multi-source fonctionnel faisant appel à diverses API.

Il a été très intéressant de travailler sur les différentes API et sur la communication des composants. Les composants permettant de découper le front en diverses fonctionnalités afin de simplifier le code et la compréhension de ce dernier mais en travaillant ainsi on pourra également ajouter, dans l'avenir, d'autres source à notre player.

Par exemple, en plus des vidéos Youtube et Dailymotion on peut également visionner des images mais on pourrait également ajouter un autre player de vidéo. La seule contrainte serait que ce player possède une API.

Conclusion

Au cours de mon entretien à Norsys, j'avais dû illustrer ma journée à Norsys avec l'image d'une carte. J'ai choisi celle d'une chenille entrant dans un labyrinthe. Au bout du bon chemin, elle trouve ses ailes. Aujourd'hui encore l'image correspond à mon aventure Norsys.

Je suis arrivée dans cette entreprise, un peu perdue, comme jetée dans l'inconnu que représente le monde de l'entreprise mais avec l'aide des collaborateurs de l'agence j'ai pu trouver le bon chemin en m'armant des bons outils.

Des outils que je connaissais déjà mais que j'ai découvert pour la plupart, notamment Angular que j'ai principalement utilisé au cours de mon projet ou encore les API qui m'ont tourmenté un petit moment. Mais au-delà de ces difficultés, je retiendrais le plaisir de voir un projet ainsi évoluer. Partir d'une page blanche et pouvoir faire évoluer ce projet jour après jour.

Aujourd'hui notre projet de Web-TV est en très bonne voie. Nous travaillons désormais sur la partie authentification pour, à terme, pouvoir gérer la gestion des chaînes et des médias ainsi que cette idée de chaîne publique et privée.

J'ai déjà bien avancé la partie gestion destinée à l'administrateur de l'application et l'authentification avance à grand pas.

J'ai énormément appris et me suis épanouie en participant à la création de cette Web TV. Cette expérience m'a enrichi et m'a fait grandir. J'en garderais un excellent souvenir et je suis à ce jour fière du travail fournit.

Glossaire

Vous trouverez ici la liste alphabetique des mots du vocabulaire spécialisé que j'utilise dans mon rapport.

API : API est un acronyme pour Applications Programming Interface. Une API est une interface de programmation qui permet de se « brancher » sur une application pour échanger des données

Application Web : Une web application désigne à l'origine une application pouvant être exécutée par le biais d'un navigateur Internet. Un webmail, un service bancaire en ligne ou un moteur de recherche sont donc des web applications.

Back/Backend : En informatique, un back-end (parfois aussi appelé un arrière-plan) est un terme désignant un étage de sortie d'un logiciel devant produire un résultat. On l'oppose au front-end (aussi appelé un frontal) qui lui est la partie visible de l'iceberg.

Dans un magasin, on trouve une arrière-boutique où sont stockés les articles, et un bureau qui assure le bon fonctionnement du magasin. Il s'agit du back-end, de tout ce qui se passe en arrière-plan sans que le client ne s'en rende compte.

Environnement de développement : C'est un environnement de programmation complet qui se présente sous la forme d'une application. L'outil aide les développeurs à concevoir et à documenter leurs codes comme un traitement de texte aide à produire des documents écrits.

Framework/micro-framework : C'est l'ensemble d'outils constituant les fondations d'un logiciel informatique ou d'applications web. Destiné autant à faciliter le travail qu'à augmenter la productivité du programmateur qui l'utilisera.

Exemple : Le framework d'un logiciel de dessin contiendra des pinceaux, des palettes de couleurs, etc...

Un micro-framework fait référence à une minimalisation d'un framework. C'est, par définition, un petit framework simple et rapide qui vous permet de monter une application

Front/Frontend : Lorsque l'on parle de « Front-End », il s'agit finalement des éléments du site que l'on voit à l'écran et avec lesquels on peut interagir.

Si on reprend l'idée du magasin évoqué pour le bak, on retrouve un service à la clientèle et des étalages. Il s'agit du front-end, de ce que le client voit.

HTML : C'est une abréviation de "L'Hypertext Markup Language". La signification de cet acronyme anglais se traduit en français par "langage de balisage d'hypertexte".

Ce langage informatique n'est pas réellement un langage de programmation mais un langage de balisage. En d'autres mots c'est un format de données qui permet de concevoir une page web.

JSON : JSON (JavaScript Object Notation) est un format d'échange de données en texte lisible. Il est utilisé pour représenter des structures de données et des objets simples dans un code qui repose sur un navigateur Web.

PostMan : Postman est un client REST proposé par Google. Il est disponible sous la forme d'une extension Chrome ou bien d'une application stand-alone.

Concrètement il permet de simuler des appels à l'API tel que des GET ou de DELETE.

Requête HTTP : L'acronyme HTTP signifie Hypertext Transfer Protocol (traduction : protocole de transfert hypertexte). Ce protocole définit la communication entre un client (exemple : navigateur) et un serveur sur le World Wide Web.

Story Mapping : Il permet de définir le besoin utilisateur complet d'un produit en phase de conception. Le principe consiste à définir les usages que les utilisateurs cible auront et ainsi concevoir plus efficacement le produit avec un mécanisme des priorités.

Bibliographie

Google Developer. Youtube Data API. Consulé à plusieurs reprise tout au long du mois de mai.
Disponible sur : <https://developers.google.com/youtube/v3/>

Google Developer. Youtube Player API. Consulé à plusieurs reprise tout au long du mois de mai. Disponible sur : https://developers.google.com/youtube/iframe_api_reference?hl=fr

Dailymotion Developer. Dailymotion Data API. Consulé à plusieurs reprise tout au long du mois de mai. Disponible sur : <https://developer.dailymotion.com/api>

Dailymotion Developer. Dailymotion Player API. Consulé à plusieurs reprise tout au long du mois de mai. Disponible sur : <https://developer.dailymotion.com/player>

Angular. Angular Material. Consulé à plusieurs reprise tout du stage. Disponible sur : <https://material.angular.io/components/component/tabs>

Bootstrap. Bootstrap Components. Consulté le 5 et 6 juin. Disponible sur : <https://getbootstrap.com/components/>

Docker. Docker Docs. Consulté la semaine du 1 mai. Disponible sur : <https://docs.docker.com/engine/installation/linux/fedora/>

NodeJs. Installing Node.js. Consulté la semaine du 1 mai. Disponible sur : <https://nodejs.org/en/download/package-manager/>

Yarn. Installation Yarn. Consulté la semaine du 1 mai. Disponible sur : <https://yarnpkg.com/lang/en/docs/install/>

Sumit Arora. Angular.Angular-Cli Wiki. Consulté la semaine du 1 mai et le 6 juin. Disponible sur : <https://github.com/angular/angular-cli/wiki>

Stephenparish. Convention de nommage des commits. Consulté la première semaine du moi d'avril. Disponible sur : <https://gist.github.com/stephenparish/9941e89d80e2bc58a153>

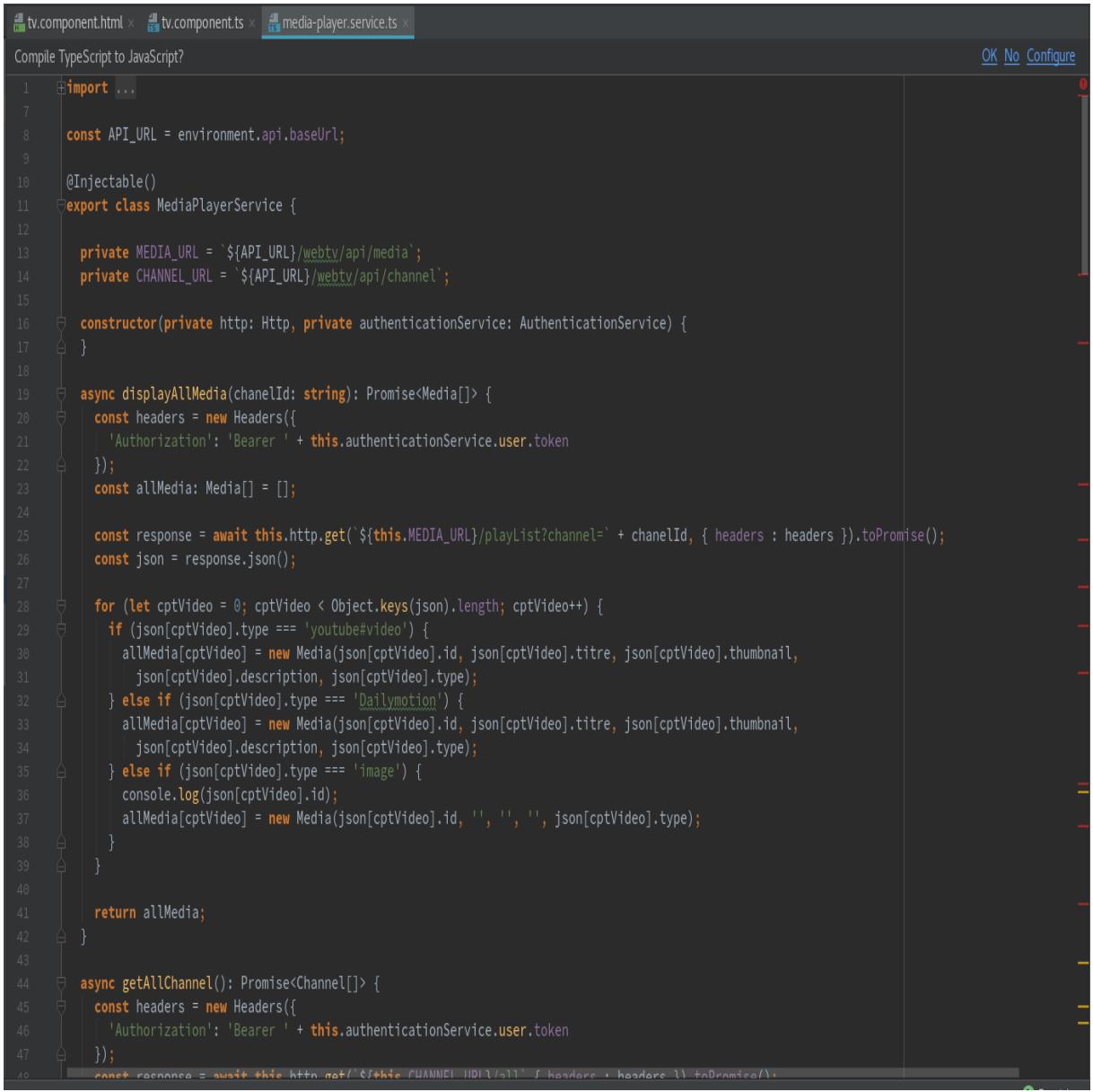
Maven. Maven. Consulté la première semaine du moi d'avril. Disponible sur : <https://maven.apache.org/>

Spring. Spring Framework. Consulté tout au long du stage. Disponible sur : <https://projects.spring.io/spring-framework/S>

Spring. Spring Initializr. Consulté lors de la semaine du 1 mai pour l'initialisation du projet. Disponible sur : <https://start.spring.io/>

A

Annexes



```

1 import ...
7
8 const API_URL = environment.api.baseUrl;
9
10 @Injectable()
11 export class MediaPlayerService {
12
13   private MEDIA_URL = `${API_URL}/webtv/api/media`;
14   private CHANNEL_URL = `${API_URL}/webtv/api/channel`;
15
16   constructor(private http: Http, private authenticationService: AuthenticationService) {
17   }
18
19   async displayAllMedia(channelId: string): Promise<Media[]> {
20     const headers = new Headers({
21       'Authorization': 'Bearer ' + this.authenticationService.user.token
22     });
23     const allMedia: Media[] = [];
24
25     const response = await this.http.get(`${this.MEDIA_URL}/playList?channel=` + channelId, { headers: headers }).toPromise();
26     const json = response.json();
27
28     for (let cptVideo = 0; cptVideo < Object.keys(json).length; cptVideo++) {
29       if (json[cptVideo].type === 'youtube#video') {
30         allMedia[cptVideo] = new Media(json[cptVideo].id, json[cptVideo].titre, json[cptVideo].thumbnail,
31           json[cptVideo].description, json[cptVideo].type);
32       } else if (json[cptVideo].type === 'Dailymotion') {
33         allMedia[cptVideo] = new Media(json[cptVideo].id, json[cptVideo].titre, json[cptVideo].thumbnail,
34           json[cptVideo].description, json[cptVideo].type);
35       } else if (json[cptVideo].type === 'image') {
36         console.log(json[cptVideo].id);
37         allMedia[cptVideo] = new Media(json[cptVideo].id, '', '', '', json[cptVideo].type);
38       }
39     }
40
41     return allMedia;
42   }
43
44   async getAllChannel(): Promise<Channel[]> {
45     const headers = new Headers({
46       'Authorization': 'Bearer ' + this.authenticationService.user.token
47     });
48     const response = await this.http.get(`${this.CHANNEL_URL}/all`, { headers: headers }).toPromise();
49
50     const json = response.json();
51
52     for (let cptChannel = 0; cptChannel < Object.keys(json).length; cptChannel++) {
53       const channel = json[cptChannel];
54
55       const channelObj = new Channel(
56         channel.id,
57         channel.name,
58         channel.description,
59         channel.thumbnail,
60         channel.type);
61
62       allChannels.push(channelObj);
63     }
64
65     return allChannels;
66   }
67
68   private handleError(error: any): Observable<never> {
69     console.error(error.message);
70     return Observable.throw(error.message);
71   }
72
73   private handleSuccess(response: Response): Observable<any> {
74     return Observable.of(response);
75   }
76
77   private handleJSON(response: Response): Observable<any> {
78     return response.json();
79   }
80
81   private handleText(response: Response): Observable<string> {
82     return response.text();
83   }
84
85   private handleBinary(response: Response): Observable<any> {
86     return response.blob();
87   }
88
89   private handleObject(response: Response): Observable<any> {
90     return response.json();
91   }
92
93   private handleFile(response: Response): Observable<any> {
94     return response.blob();
95   }
96
97   private handleImage(response: Response): Observable<any> {
98     return response.blob();
99   }
100
101   private handleScript(response: Response): Observable<any> {
102     return response.blob();
103   }
104
105   private handleJSONP(response: Response): Observable<any> {
106     return response.json();
107   }
108
109   private handleTextP(response: Response): Observable<string> {
109     return response.text();
110   }
111
112   private handleBinaryP(response: Response): Observable<any> {
113     return response.blob();
114   }
115
116   private handleObjectP(response: Response): Observable<any> {
117     return response.json();
118   }
119
120   private handleFileP(response: Response): Observable<any> {
121     return response.blob();
122   }
123
124   private handleImageP(response: Response): Observable<any> {
125     return response.blob();
126   }
127
128   private handleScriptP(response: Response): Observable<any> {
129     return response.blob();
130   }
131
132   private handleTextA(response: Response): Observable<string> {
133     return response.text();
134   }
135
136   private handleBinaryA(response: Response): Observable<any> {
137     return response.blob();
138   }
139
140   private handleObjectA(response: Response): Observable<any> {
141     return response.json();
142   }
143
144   private handleFileA(response: Response): Observable<any> {
145     return response.blob();
146   }
147
148   private handleImageA(response: Response): Observable<any> {
149     return response.blob();
150   }
151
152   private handleScriptA(response: Response): Observable<any> {
153     return response.blob();
154   }
155
156   private handleTextB(response: Response): Observable<string> {
157     return response.text();
158   }
159
160   private handleBinaryB(response: Response): Observable<any> {
161     return response.blob();
162   }
163
164   private handleObjectB(response: Response): Observable<any> {
165     return response.json();
166   }
167
168   private handleFileB(response: Response): Observable<any> {
169     return response.blob();
170   }
171
172   private handleImageB(response: Response): Observable<any> {
173     return response.blob();
174   }
175
176   private handleScriptB(response: Response): Observable<any> {
177     return response.blob();
178   }
179
180   private handleTextC(response: Response): Observable<string> {
181     return response.text();
182   }
183
184   private handleBinaryC(response: Response): Observable<any> {
185     return response.blob();
186   }
187
188   private handleObjectC(response: Response): Observable<any> {
189     return response.json();
190   }
191
192   private handleFileC(response: Response): Observable<any> {
193     return response.blob();
194   }
195
196   private handleImageC(response: Response): Observable<any> {
197     return response.blob();
198   }
199
200   private handleScriptC(response: Response): Observable<any> {
201     return response.blob();
202   }
203
204   private handleTextD(response: Response): Observable<string> {
205     return response.text();
206   }
207
208   private handleBinaryD(response: Response): Observable<any> {
209     return response.blob();
210   }
211
212   private handleObjectD(response: Response): Observable<any> {
213     return response.json();
214   }
215
216   private handleFileD(response: Response): Observable<any> {
217     return response.blob();
218   }
219
220   private handleImageD(response: Response): Observable<any> {
221     return response.blob();
222   }
223
224   private handleScriptD(response: Response): Observable<any> {
225     return response.blob();
226   }
227
228   private handleTextE(response: Response): Observable<string> {
229     return response.text();
230   }
231
232   private handleBinaryE(response: Response): Observable<any> {
233     return response.blob();
234   }
235
236   private handleObjectE(response: Response): Observable<any> {
237     return response.json();
238   }
239
240   private handleFileE(response: Response): Observable<any> {
241     return response.blob();
242   }
243
244   private handleImageE(response: Response): Observable<any> {
245     return response.blob();
246   }
247
248   private handleScriptE(response: Response): Observable<any> {
249     return response.blob();
250   }
251
252   private handleTextF(response: Response): Observable<string> {
253     return response.text();
254   }
255
256   private handleBinaryF(response: Response): Observable<any> {
257     return response.blob();
258   }
259
260   private handleObjectF(response: Response): Observable<any> {
261     return response.json();
262   }
263
264   private handleFileF(response: Response): Observable<any> {
265     return response.blob();
266   }
267
268   private handleImageF(response: Response): Observable<any> {
269     return response.blob();
270   }
271
272   private handleScriptF(response: Response): Observable<any> {
273     return response.blob();
274   }
275
276   private handleTextG(response: Response): Observable<string> {
277     return response.text();
278   }
279
280   private handleBinaryG(response: Response): Observable<any> {
281     return response.blob();
282   }
283
284   private handleObjectG(response: Response): Observable<any> {
285     return response.json();
286   }
287
288   private handleFileG(response: Response): Observable<any> {
289     return response.blob();
290   }
291
292   private handleImageG(response: Response): Observable<any> {
293     return response.blob();
294   }
295
296   private handleScriptG(response: Response): Observable<any> {
297     return response.blob();
298   }
299
300   private handleTextH(response: Response): Observable<string> {
301     return response.text();
302   }
303
304   private handleBinaryH(response: Response): Observable<any> {
305     return response.blob();
306   }
307
308   private handleObjectH(response: Response): Observable<any> {
309     return response.json();
310   }
311
312   private handleFileH(response: Response): Observable<any> {
313     return response.blob();
314   }
315
316   private handleImageH(response: Response): Observable<any> {
317     return response.blob();
318   }
319
320   private handleScriptH(response: Response): Observable<any> {
321     return response.blob();
322   }
323
324   private handleTextI(response: Response): Observable<string> {
325     return response.text();
326   }
327
328   private handleBinaryI(response: Response): Observable<any> {
329     return response.blob();
330   }
331
332   private handleObjectI(response: Response): Observable<any> {
333     return response.json();
334   }
335
336   private handleFileI(response: Response): Observable<any> {
337     return response.blob();
338   }
339
340   private handleImageI(response: Response): Observable<any> {
341     return response.blob();
342   }
343
344   private handleScriptI(response: Response): Observable<any> {
345     return response.blob();
346   }
347
348   private handleTextJ(response: Response): Observable<string> {
349     return response.text();
350   }
351
352   private handleBinaryJ(response: Response): Observable<any> {
353     return response.blob();
354   }
355
356   private handleObjectJ(response: Response): Observable<any> {
357     return response.json();
358   }
359
360   private handleFileJ(response: Response): Observable<any> {
361     return response.blob();
362   }
363
364   private handleImageJ(response: Response): Observable<any> {
365     return response.blob();
366   }
367
368   private handleScriptJ(response: Response): Observable<any> {
369     return response.blob();
370   }
371
372   private handleTextK(response: Response): Observable<string> {
373     return response.text();
374   }
375
376   private handleBinaryK(response: Response): Observable<any> {
377     return response.blob();
378   }
379
380   private handleObjectK(response: Response): Observable<any> {
381     return response.json();
382   }
383
384   private handleFileK(response: Response): Observable<any> {
385     return response.blob();
386   }
387
388   private handleImageK(response: Response): Observable<any> {
389     return response.blob();
390   }
391
392   private handleScriptK(response: Response): Observable<any> {
393     return response.blob();
394   }
395
396   private handleTextL(response: Response): Observable<string> {
397     return response.text();
398   }
399
400   private handleBinaryL(response: Response): Observable<any> {
401     return response.blob();
402   }
403
404   private handleObjectL(response: Response): Observable<any> {
405     return response.json();
406   }
407
408   private handleFileL(response: Response): Observable<any> {
409     return response.blob();
410   }
411
412   private handleImageL(response: Response): Observable<any> {
413     return response.blob();
414   }
415
416   private handleScriptL(response: Response): Observable<any> {
417     return response.blob();
418   }
419
420   private handleTextM(response: Response): Observable<string> {
421     return response.text();
422   }
423
424   private handleBinaryM(response: Response): Observable<any> {
425     return response.blob();
426   }
427
428   private handleObjectM(response: Response): Observable<any> {
429     return response.json();
430   }
431
432   private handleFileM(response: Response): Observable<any> {
433     return response.blob();
434   }
435
436   private handleImageM(response: Response): Observable<any> {
437     return response.blob();
438   }
439
440   private handleScriptM(response: Response): Observable<any> {
441     return response.blob();
442   }
443
444   private handleTextN(response: Response): Observable<string> {
445     return response.text();
446   }
447
448   private handleBinaryN(response: Response): Observable<any> {
449     return response.blob();
450   }
451
452   private handleObjectN(response: Response): Observable<any> {
453     return response.json();
454   }
455
456   private handleFileN(response: Response): Observable<any> {
457     return response.blob();
458   }
459
460   private handleImageN(response: Response): Observable<any> {
461     return response.blob();
462   }
463
464   private handleScriptN(response: Response): Observable<any> {
465     return response.blob();
466   }
467
468   private handleTextO(response: Response): Observable<string> {
469     return response.text();
470   }
471
472   private handleBinaryO(response: Response): Observable<any> {
473     return response.blob();
474   }
475
476   private handleObjectO(response: Response): Observable<any> {
477     return response.json();
478   }
479
480   private handleFileO(response: Response): Observable<any> {
481     return response.blob();
482   }
483
484   private handleImageO(response: Response): Observable<any> {
485     return response.blob();
486   }
487
488   private handleScriptO(response: Response): Observable<any> {
489     return response.blob();
490   }
491
492   private handleTextP(response: Response): Observable<string> {
493     return response.text();
494   }
495
496   private handleBinaryP(response: Response): Observable<any> {
497     return response.blob();
498   }
499
500   private handleObjectP(response: Response): Observable<any> {
501     return response.json();
502   }
503
504   private handleFileP(response: Response): Observable<any> {
505     return response.blob();
506   }
507
508   private handleImageP(response: Response): Observable<any> {
509     return response.blob();
510   }
511
512   private handleScriptP(response: Response): Observable<any> {
513     return response.blob();
514   }
515
516   private handleTextQ(response: Response): Observable<string> {
517     return response.text();
518   }
519
520   private handleBinaryQ(response: Response): Observable<any> {
521     return response.blob();
522   }
523
524   private handleObjectQ(response: Response): Observable<any> {
525     return response.json();
526   }
527
528   private handleFileQ(response: Response): Observable<any> {
529     return response.blob();
530   }
531
532   private handleImageQ(response: Response): Observable<any> {
533     return response.blob();
534   }
535
536   private handleScriptQ(response: Response): Observable<any> {
537     return response.blob();
538   }
539
540   private handleTextR(response: Response): Observable<string> {
541     return response.text();
542   }
543
544   private handleBinaryR(response: Response): Observable<any> {
545     return response.blob();
546   }
547
548   private handleObjectR(response: Response): Observable<any> {
549     return response.json();
550   }
551
552   private handleFileR(response: Response): Observable<any> {
553     return response.blob();
554   }
555
556   private handleImageR(response: Response): Observable<any> {
557     return response.blob();
558   }
559
560   private handleScriptR(response: Response): Observable<any> {
561     return response.blob();
562   }
563
564   private handleTextS(response: Response): Observable<string> {
565     return response.text();
566   }
567
568   private handleBinaryS(response: Response): Observable<any> {
569     return response.blob();
570   }
571
572   private handleObjectS(response: Response): Observable<any> {
573     return response.json();
574   }
575
576   private handleFileS(response: Response): Observable<any> {
577     return response.blob();
578   }
579
580   private handleImageS(response: Response): Observable<any> {
581     return response.blob();
582   }
583
584   private handleScriptS(response: Response): Observable<any> {
585     return response.blob();
586   }
587
588   private handleTextT(response: Response): Observable<string> {
589     return response.text();
590   }
591
592   private handleBinaryT(response: Response): Observable<any> {
593     return response.blob();
594   }
595
596   private handleObjectT(response: Response): Observable<any> {
597     return response.json();
598   }
599
600   private handleFileT(response: Response): Observable<any> {
601     return response.blob();
602   }
603
604   private handleImageT(response: Response): Observable<any> {
605     return response.blob();
606   }
607
608   private handleScriptT(response: Response): Observable<any> {
609     return response.blob();
610   }
611
612   private handleTextU(response: Response): Observable<string> {
613     return response.text();
614   }
615
616   private handleBinaryU(response: Response): Observable<any> {
617     return response.blob();
618   }
619
620   private handleObjectU(response: Response): Observable<any> {
621     return response.json();
622   }
623
624   private handleFileU(response: Response): Observable<any> {
625     return response.blob();
626   }
627
628   private handleImageU(response: Response): Observable<any> {
629     return response.blob();
630   }
631
632   private handleScriptU(response: Response): Observable<any> {
633     return response.blob();
634   }
635
636   private handleTextV(response: Response): Observable<string> {
637     return response.text();
638   }
639
640   private handleBinaryV(response: Response): Observable<any> {
641     return response.blob();
642   }
643
644   private handleObjectV(response: Response): Observable<any> {
645     return response.json();
646   }
647
648   private handleFileV(response: Response): Observable<any> {
649     return response.blob();
650   }
651
652   private handleImageV(response: Response): Observable<any> {
653     return response.blob();
654   }
655
656   private handleScriptV(response: Response): Observable<any> {
657     return response.blob();
658   }
659
660   private handleTextW(response: Response): Observable<string> {
661     return response.text();
662   }
663
664   private handleBinaryW(response: Response): Observable<any> {
665     return response.blob();
666   }
667
668   private handleObjectW(response: Response): Observable<any> {
669     return response.json();
670   }
671
672   private handleFileW(response: Response): Observable<any> {
673     return response.blob();
674   }
675
676   private handleImageW(response: Response): Observable<any> {
677     return response.blob();
678   }
679
680   private handleScriptW(response: Response): Observable<any> {
681     return response.blob();
682   }
683
684   private handleTextX(response: Response): Observable<string> {
685     return response.text();
686   }
687
688   private handleBinaryX(response: Response): Observable<any> {
689     return response.blob();
690   }
691
692   private handleObjectX(response: Response): Observable<any> {
693     return response.json();
694   }
695
696   private handleFileX(response: Response): Observable<any> {
697     return response.blob();
698   }
699
700   private handleImageX(response: Response): Observable<any> {
701     return response.blob();
702   }
703
704   private handleScriptX(response: Response): Observable<any> {
705     return response.blob();
706   }
707
708   private handleTextY(response: Response): Observable<string> {
709     return response.text();
710   }
711
712   private handleBinaryY(response: Response): Observable<any> {
713     return response.blob();
714   }
715
716   private handleObjectY(response: Response): Observable<any> {
717     return response.json();
718   }
719
720   private handleFileY(response: Response): Observable<any> {
721     return response.blob();
722   }
723
724   private handleImageY(response: Response): Observable<any> {
725     return response.blob();
726   }
727
728   private handleScriptY(response: Response): Observable<any> {
729     return response.blob();
730   }
731
732   private handleTextZ(response: Response): Observable<string> {
733     return response.text();
734   }
735
736   private handleBinaryZ(response: Response): Observable<any> {
737     return response.blob();
738   }
739
740   private handleObjectZ(response: Response): Observable<any> {
741     return response.json();
742   }
743
744   private handleFileZ(response: Response): Observable<any> {
745     return response.blob();
746   }
747
748   private handleImageZ(response: Response): Observable<any> {
749     return response.blob();
750   }
751
752   private handleScriptZ(response: Response): Observable<any> {
753     return response.blob();
754   }
755
756   private handleTextAA(response: Response): Observable<string> {
757     return response.text();
758   }
759
760   private handleBinaryAA(response: Response): Observable<any> {
761     return response.blob();
762   }
763
764   private handleObjectAA(response: Response): Observable<any> {
765     return response.json();
766   }
767
768   private handleFileAA(response: Response): Observable<any> {
769     return response.blob();
770   }
771
772   private handleImageAA(response: Response): Observable<any> {
773     return response.blob();
774   }
775
776   private handleScriptAA(response: Response): Observable<any> {
777     return response.blob();
778   }
779
780   private handleTextBB(response: Response): Observable<string> {
781     return response.text();
782   }
783
784   private handleBinaryBB(response: Response): Observable<any> {
785     return response.blob();
786   }
787
788   private handleObjectBB(response: Response): Observable<any> {
789     return response.json();
790   }
791
792   private handleFileBB(response: Response): Observable<any> {
793     return response.blob();
794   }
795
796   private handleImageBB(response: Response): Observable<any> {
797     return response.blob();
798   }
799
800   private handleScriptBB(response: Response): Observable<any> {
801     return response.blob();
802   }
803
804   private handleTextCC(response: Response): Observable<string> {
805     return response.text();
806   }
807
808   private handleBinaryCC(response: Response): Observable<any> {
809     return response.blob();
810   }
811
812   private handleObjectCC(response: Response): Observable<any> {
813     return response.json();
814   }
815
816   private handleFileCC(response: Response): Observable<any> {
817     return response.blob();
818   }
819
820   private handleImageCC(response: Response): Observable<any> {
821     return response.blob();
822   }
823
824   private handleScriptCC(response: Response): Observable<any> {
825     return response.blob();
826   }
827
828   private handleTextDD(response: Response): Observable<string> {
829     return response.text();
830   }
831
832   private handleBinaryDD(response: Response): Observable<any> {
833     return response.blob();
834   }
835
836   private handleObjectDD(response: Response): Observable<any> {
837     return response.json();
838   }
839
840   private handleFileDD(response: Response): Observable<any> {
841     return response.blob();
842   }
843
844   private handleImageDD(response: Response): Observable<any> {
845     return response.blob();
846   }
847
848   private handleScriptDD(response: Response): Observable<any> {
849     return response.blob();
850   }
851
852   private handleTextEE(response: Response): Observable<string> {
853     return response.text();
854   }
855
856   private handleBinaryEE(response: Response): Observable<any> {
857     return response.blob();
858   }
859
860   private handleObjectEE(response: Response): Observable<any> {
861     return response.json();
862   }
863
864   private handleFileEE(response: Response): Observable<any> {
865     return response.blob();
866   }
867
868   private handleImageEE(response: Response): Observable<any> {
869     return response.blob();
870   }
871
872   private handleScriptEE(response: Response): Observable<any> {
873     return response.blob();
874   }
875
876   private handleTextFF(response: Response): Observable<string> {
877     return response.text();
878   }
879
880   private handleBinaryFF(response: Response): Observable<any> {
881     return response.blob();
882   }
883
884   private handleObjectFF(response: Response): Observable<any> {
885     return response.json();
886   }
887
888   private handleFileFF(response: Response): Observable<any> {
889     return response.blob();
890   }
891
892   private handleImageFF(response: Response): Observable<any> {
893     return response.blob();
894   }
895
896   private handleScriptFF(response: Response): Observable<any> {
897     return response.blob();
898   }
899
900   private handleTextGG(response: Response): Observable<string> {
901     return response.text();
902   }
903
904   private handleBinaryGG(response: Response): Observable<any> {
905     return response.blob();
906   }
907
908   private handleObjectGG(response: Response): Observable<any> {
909     return response.json();
910   }
911
912   private handleFileGG(response: Response): Observable<any> {
913     return response.blob();
914   }
915
916   private handleImageGG(response: Response): Observable<any> {
917     return response.blob();
918   }
919
920   private handleScriptGG(response: Response): Observable<any> {
921     return response.blob();
922   }
923
924   private handleTextHH(response: Response): Observable<string> {
925     return response.text();
926   }
927
928   private handleBinaryHH(response: Response): Observable<any> {
929     return response.blob();
930   }
931
932   private handleObjectHH(response: Response): Observable<any> {
933     return response.json();
934   }
935
936   private handleFileHH(response: Response): Observable<any> {
937     return response.blob();
938   }
939
940   private handleImageHH(response: Response): Observable<any> {
941     return response.blob();
942   }
943
944   private handleScriptHH(response: Response): Observable<any> {
945     return response.blob();
946   }
947
948   private handleTextII(response: Response): Observable<string> {
949     return response.text();
950   }
951
952   private handleBinaryII(response: Response): Observable<any> {
953     return response.blob();
954   }
955
956   private handleObjectII(response: Response): Observable<any> {
957     return response.json();
958   }
959
960   private handleFileII(response: Response): Observable<any> {
961     return response.blob();
962   }
963
964   private handleImageII(response: Response): Observable<any> {
965     return response.blob();
966   }
967
968   private handleScriptII(response: Response): Observable<any> {
969     return response.blob();
970   }
971
972   private handleTextJJ(response: Response): Observable<string> {
973     return response.text();
974   }
975
976   private handleBinaryJJ(response: Response): Observable<any> {
977     return response.blob();
978   }
979
980   private handleObjectJJ(response: Response): Observable<any> {
981     return response.json();
982   }
983
984   private handleFileJJ(response: Response): Observable<any> {
985     return response.blob();
986   }
987
988   private handleImageJJ(response: Response): Observable<any> {
989     return response.blob();
990   }
991
992   private handleScriptJJ(response: Response): Observable<any> {
993     return response.blob();
994   }
995
996   private handleTextKK(response: Response): Observable<string> {
997     return response.text();
998   }
999
1000  private handleBinaryKK(response: Response): Observable<any> {
1001    return response.blob();
1002  }
1003
1004  private handleObjectKK(response: Response): Observable<any> {
1005    return response.json();
1006  }
1007
1008  private handleFileKK(response: Response): Observable<any> {
1009    return response.blob();
1010  }
1011
1012  private handleImageKK(response: Response): Observable<any> {
1013    return response.blob();
1014  }
1015
1016  private handleScriptKK(response: Response): Observable<any> {
1017    return response.blob();
1018  }
1019
1020  private handleTextLL(response: Response): Observable<string> {
1021    return response.text();
1022  }
1023
1024  private handleBinaryLL(response: Response): Observable<any> {
1025    return response.blob();
1026  }
1027
1028  private handleObjectLL(response: Response): Observable<any> {
1029    return response.json();
1030  }
1031
1032  private handleFileLL(response: Response): Observable<any> {
1033    return response.blob();
1034  }
1035
1036  private handleImageLL(response: Response): Observable<any> {
1037    return response.blob();
1038  }
1039
1040  private handleScriptLL(response: Response): Observable<any> {
1041    return response.blob();
1042  }
1043
1044  private handleTextMM(response: Response): Observable<string> {
1045    return response.text();
1046  }
1047
1048  private handleBinaryMM(response: Response): Observable<any> {
1049    return response.blob();
1050  }
1051
1052  private handleObjectMM(response: Response): Observable<any> {
1053    return response.json();
1054  }
1055
1056  private handleFileMM(response: Response): Observable<any> {
1057    return response.blob();
1058  }
1059
1060  private handleImageMM(response: Response): Observable<any> {
1061    return response.blob();
1062  }
1063
1064  private handleScriptMM(response: Response): Observable<any> {
1065    return response.blob();
1066  }
1067
1068  private handleTextNN(response: Response): Observable<string> {
1069    return response.text();
1070  }
1071
1072  private handleBinaryNN(response: Response): Observable<any> {
1073    return response.blob();
1074  }
1075
1076  private handleObjectNN(response: Response): Observable<any> {
1077    return response.json();
1078  }
1079
1080  private handleFileNN(response: Response): Observable<any> {
1081    return response.blob();
1082  }
1083
1084  private handleImageNN(response: Response): Observable<any> {
1085    return response.blob();
1086  }
1087
1088  private handleScriptNN(response: Response): Observable<any> {
1089    return response.blob();
1090  }
1091
1092  private handleTextOO(response: Response): Observable<string> {
1093    return response.text();
1094  }
1095
1096  private handleBinaryOO(response: Response): Observable<any> {
1097    return response.blob();
1098  }
1099
1100  private handleObjectOO(response: Response): Observable<any> {
1101    return response.json();
1102  }
1103
1104  private handleFileOO(response: Response): Observable<any> {
1105    return response.blob();
1106  }
1107
1108  private handleImageOO(response: Response): Observable<any> {
1109    return response.blob();
1110  }
1111
1112  private handleScriptOO(response: Response): Observable<any> {
1113    return response.blob();
1114  }
1115
1116  private handleTextPP(response: Response): Observable<string> {
1117    return response.text();
1118  }
1119
1120  private handleBinaryPP(response: Response): Observable<any> {
1121    return response.blob();
1122  }
1123
1124  private handleObjectPP(response: Response): Observable<any> {
1125    return response.json();
1126  }
1127
1128  private handleFilePP(response: Response): Observable<any> {
1129    return response.blob();
1130  }
1131
1132  private handleImagePP(response: Response): Observable<any> {
1133    return response.blob();
1134  }
1135
1136  private handleScriptPP(response: Response): Observable<any> {
1137    return response.blob();
1138  }
1139
1140  private handleTextQQ(response: Response): Observable<string> {
1141    return response.text();
1142  }
1143
1144  private handleBinaryQQ(response: Response): Observable<any> {
1145    return response.blob();
1146  }
1147
1148  private handleObjectQQ(response: Response): Observable<any> {
1149    return response.json();
1150  }
1151
1152  private handleFileQQ(response: Response): Observable<any> {
1153    return response.blob();
1154  }
1155
1156  private handleImageQQ(response: Response): Observable<any> {
1157    return response.blob();
1158  }
1159
1160  private handleScriptQQ(response: Response): Observable<any> {
1161    return response.blob();
1162  }
1163
1164  private handleTextRR(response: Response): Observable<string> {
1165    return response.text();
1166  }
1167
1168  private handleBinaryRR(response: Response): Observable<any> {
1169    return response.blob();
1170  }
1171
1172  private handleObjectRR(response: Response): Observable<any> {
1173    return response.json();
1174  }
1175
1176  private handleFileRR(response: Response): Observable<any> {
1177    return response.blob();
1178  }
1179
1180  private handleImageRR(response: Response): Observable<any> {
1181    return response.blob();
1182  }
1183
1184  private handleScriptRR(response: Response): Observable<any> {
1185    return response.blob();
1186  }
1187
1188  private handleTextSS(response: Response): Observable<string> {
1189    return response.text();
1190  }
1191
1192  private handleBinarySS(response: Response): Observable<any> {
1193    return response.blob();
1194  }
1195
1196  private handleObjectSS(response: Response): Observable<any> {
1197    return response.json();
1198  }
1199
1200  private handleFileSS(response: Response): Observable<any> {
1201    return response.blob();
1202  }
1203
1204 
```

```

tv.component.html x tv.component.ts x media-player.service.ts x
Compile TypeScript to JavaScript?
OK No Configure
42 }
43
44 async getAllChannel(): Promise<Channel[]> {
45   const headers = new Headers({
46     'Authorization': 'Bearer ' + this.authenticationService.user.token
47   });
48   const response = await this.http.get(`${this.CHANNEL_URL}/all`, { headers : headers }).toPromise();
49   const json = response.json();
50
51   const channels: Channel[] = [];
52
53   for (let cptVideo = 0; cptVideo < Object.keys(json).length; cptVideo++) {
54     channels[cptVideo] = new Channel(json[cptVideo].id, json[cptVideo].nameTV, json[cptVideo].photo, json[cptVideo].description);
55   }
56
57   return channels;
58 }
59
60 async getVideoId(channelId: string): Promise<Media[]> {
61   const headers = new Headers({
62     'Authorization': 'Bearer ' + this.authenticationService.user.token
63   });
64   const response = await this.http.get(`${this.MEDIA_URL}/XByChannel?channel=' + channelId + '&taille=4', { headers : headers }).toPromise();
65   const json = response.json();
66
67   const allVideo: Media[] = [];
68
69   if (Object.keys(json).length >= 4) {
70     for (let cptMedia = 0; cptMedia < 4; cptMedia++) {
71       if (json[cptMedia].type === 'youtube#video') {
72         allVideo[cptMedia] = new Media(this.createYoutubeUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
73           json[cptMedia].description, json[cptMedia].type);
74       } else if (json[0].type === 'Dailymotion') {
75         allVideo[cptMedia] = new Media(this.createDailymotionUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
76           json[cptMedia].description, json[cptMedia].type);
77       }
78     }
79   } else {
80     for (let cptMedia = 0; cptMedia < Object.keys(json).length; cptMedia++) {
81       if (json[cptMedia].type === 'youtube#video') {
82         allVideo[cptMedia] = new Media(this.createYoutubeUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
83           json[cptMedia].description, json[cptMedia].type);
84       }
85     }
86   }
87 }
88 
```

FIGURE A.2 – Second extrait du service chargé des médias et des channels

```

tv.component.html x tv.component.ts x media-player.service.ts x
Compile TypeScript to JavaScript?
OK No Configure
57     return channels;
58   }
59
60   async getVideoId(channelId: string): Promise<Media[]> {
61     const headers = new Headers([
62       'Authorization': `Bearer ${this.authenticationService.user.token}`
63     ]);
64     const response = await this.http.get(`${this.MEDIA_URL}/XByChannel?channel=${channelId}&taille=4`, { headers });
65     const json = response.json();
66
67     const allVideo: Media[] = [];
68
69     if (Object.keys(json).length >= 4) {
70       for (let cptMedia = 0; cptMedia < 4; cptMedia++) {
71         if (json[cptMedia].type === 'youtube#video') {
72           allVideo[cptMedia] = new Media(this.createYoutubeUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
73                                         json[cptMedia].description, json[cptMedia].type);
74         } else if (json[cptMedia].type === 'Dailymotion') {
75           allVideo[cptMedia] = new Media(this.createDailymotionUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
76                                         json[cptMedia].description, json[cptMedia].type);
77         }
78       }
79     } else {
80       for (let cptMedia = 0; cptMedia < Object.keys(json).length; cptMedia++) {
81         if (json[cptMedia].type === 'youtube#video') {
82           allVideo[cptMedia] = new Media(this.createYoutubeUrl(json[cptMedia].id), json[cptMedia].titre, json[cptMedia].thumbnail,
83                                         json[cptMedia].description, json[cptMedia].type);
84         }
85       }
86       for (let cptAdd = allVideo.length; cptAdd < 4; cptAdd++) {
87         allVideo[cptAdd] = new Media(' ', ' ', ' ', ' ', ' ');
88       }
89     }
90     return allVideo;
91   }
92
93   async getVitrine(): Promise<string> {
94     const headers = new Headers([
95       'Authorization': `Bearer ${this.authenticationService.user.token}`
96     ]);
97     const response = await this.http.get(`${this.MEDIA_URL}/vitrine`, { headers });
98     const json = response.json();
99     if (json.type === 'youtube#video') {

```

FIGURE A.3 – Troisième extrait du service chargé des médias et des channels

tv.component.html x tv.component.ts x media-player.service.ts x

OK No Configure

```
Compile TypeScript to JavaScript?
```

```
93  ↴ async getVitrine(): Promise<string> {
94    ↴ const headers = new Headers({
95      'Authorization': 'Bearer ' + this.authenticationService.user.token
96    });
97    ↴ const response = await this.http.get(`${this.MEDIA_URL}/vitrine`, { headers : headers }).toPromise();
98    ↴ const json = response.json();
99    if (json.type === 'youtube#video') {
100      return this.createYoutubeUrl(json.id);
101    } else {
102      return this.createDailymotionUrl(json.id);
103    }
104  }
105
106  ↴ async getFirstVideoChanel(channelId: string): Promise<string> {
107    ↴ const headers = new Headers({
108      'Authorization': 'Bearer ' + this.authenticationService.user.token
109    });
110    ↴ const response = await this.http.get(`${this.MEDIA_URL}/FirstByChannel?channel=' + channelId + '&type=youtube', { headers : headers }).toPromise();
111    const json = response.json();
112
113    return this.createYoutubeUrl(json.id);
114
115  }
116
117  ↴ async getFirstDescriptionVideo(channelId: string): Promise<string> {
118    ↴ const headers = new Headers({
119      'Authorization': 'Bearer ' + this.authenticationService.user.token
120    });
121    ↴ const response = await this.http.get(`${this.MEDIA_URL}/XByChannel?channel=' + channelId + '&taille=1', { headers : headers }).toPromise();
122    const json = response.json();
123    return json[0].description;
124  }
125
126  ↴ async getUpdateDescription(videoId: string): Promise<string> {
127    ↴ const headers = new Headers({
128      'Authorization': 'Bearer ' + this.authenticationService.user.token
129    });
130    ↴ const response = await this.http.get(`${this.MEDIA_URL}/?id=' + videoId, { headers : headers }).toPromise();
131    const json = response.json();
132    console.log('get : ' + json.description);
133
134    return json.description;
135 }
```

FIGURE A.4 – Quatrième extrait du service chargé des médias et des channels

```

tv.component.html x tv.component.ts x media-player.service.ts x
Compile TypeScript to JavaScript?
OK No Configure
133
134     return json.description;
135 }
136
137 async getAllVideoChaneId(chanelName: string): Promise<Media[]> {
138     const headers = new Headers({
139         'Authorization': 'Bearer ' + this.authenticationService.user.token
140     });
141     const response = await this.http.get(`.${this.MEDIA_URL}/playList?channel=` + chanelName,{ headers : headers }).toPromise();
142     const json = response.json();
143
144     const allVideo: Media[] = [];
145     let cptIsVideo = 0;
146
147     for (let cptVideo = 0; cptVideo < Object.keys(json).length; cptVideo++) {
148         if (json[cptVideo].type === 'youtube#video') {
149             allVideo[cptIsVideo] = new Media(this.createYoutubeUrl(json[cptVideo].id), json[cptVideo].titre, json[cptVideo].thumbnail,
150             json[cptVideo].description, json[cptVideo].type);
151             cptIsVideo++;
152         } else if (json[cptVideo].type === 'Dailymotion') {
153             allVideo[cptIsVideo] = new Media(this.createDailymotionUrl(json[cptVideo].id), json[cptVideo].titre, json[cptVideo].thumbnail,
154             json[cptVideo].description, json[cptVideo].type);
155             cptIsVideo++;
156         }
157     }
158     return allVideo;
159 }
160
161
162 async getFirstImageChanel(channelId: string): Promise<string> {
163
164     const headers = new Headers({
165         'Authorization': 'Bearer ' + this.authenticationService.user.token
166     });
167     const response = await this.http.get(`.${this.MEDIA_URL}/FirstByChannel?channel=` + channelId+`&type=image`,{ headers : headers }).toPromise();
168     const json = response.json();
169     const url = json.id;
170     return url;
171 }
172
173 async getAllChanelImage(channelId: string): Promise<string[]> {
174     const headers = new Headers({

```

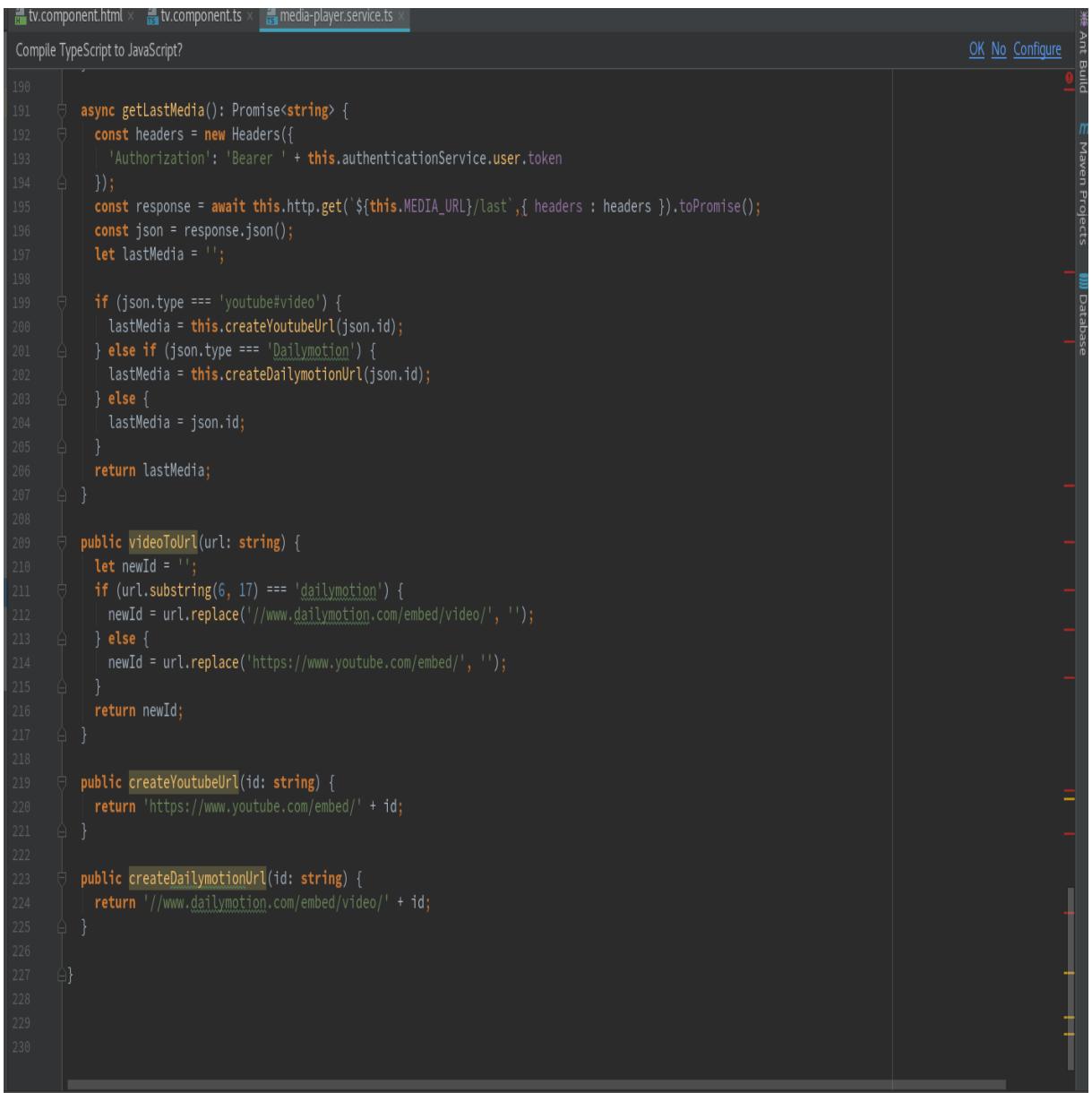
FIGURE A.5 – Cinquième extrait du service chargé des médias et des channels

```

160
161
162     async getFirstImageChanel(channelId: string): Promise<string> {
163
164         const headers = new Headers([
165             'Authorization': 'Bearer ' + this.authenticationService.user.token
166         ]);
167         const response = await this.http.get(`${this.MEDIA_URL}/FirstByChannel?channel=' + channelId + '&type=image',{ headers : headers }).toPromise();
168         const json = response.json();
169         const url = json.id;
170         return url;
171     }
172
173     async getAllChanelImage(channelId: string): Promise<string[]> {
174         const headers = new Headers([
175             'Authorization': 'Bearer ' + this.authenticationService.user.token
176         ]);
177         const response = await this.http.get(`${this.MEDIA_URL}/playList?channel= + channelId,{ headers : headers }).toPromise();
178         const json = response.json();
179
180         const allVideo: string[] = [];
181         let cpt = 0;
182         for (let cptVideo = 0; cptVideo < Object.keys(json).length; cptVideo++) {
183             if (json[cptVideo].type === 'image') {
184                 allVideo[cpt] = json[cptVideo].id;
185                 cpt++;
186             }
187         }
188         return allVideo;
189     }
190
191     async getLastMedia(): Promise<string> {
192         const headers = new Headers([
193             'Authorization': 'Bearer ' + this.authenticationService.user.token
194         ]);
195         const response = await this.http.get(`${this.MEDIA_URL}/last',{ headers : headers }).toPromise();
196         const json = response.json();
197         let lastMedia = '';
198
199         if (json.type === 'youtube#video') {
200             lastMedia = this.createYoutubeUrl(json.id);
201         } else if (json.type === 'Dailymotion') {
202             lastMedia = this.createDailymotionUrl(json.id);
203         }

```

FIGURE A.6 – Sixième extrait du service chargé des médias et des channels



The screenshot shows a code editor window with three tabs at the top: `tv.component.html`, `tv.component.ts`, and `media-player.service.ts`. The `media-player.service.ts` tab is active, displaying the following TypeScript code:

```

190
191     async getLastMedia(): Promise<string> {
192         const headers = new Headers({
193             'Authorization': `Bearer ${this.authenticationService.user.token}`
194         });
195         const response = await this.http.get(`${this.MEDIA_URL}/last`, { headers: headers }).toPromise();
196         const json = response.json();
197         let lastMedia = '';
198
199         if (json.type === 'youtube#video') {
200             lastMedia = this.createYoutubeUrl(json.id);
201         } else if (json.type === 'Dailymotion') {
202             lastMedia = this.createDailymotionUrl(json.id);
203         } else {
204             lastMedia = json.id;
205         }
206         return lastMedia;
207     }
208
209     public videoToUrl(url: string) {
210         let newId = '';
211         if (url.substring(6, 17) === 'dailymotion') {
212             newId = url.replace('://www.dailymotion.com/embed/video/', '');
213         } else {
214             newId = url.replace('https://www.youtube.com/embed/', '');
215         }
216         return newId;
217     }
218
219     public createYoutubeUrl(id: string) {
220         return `https://www.youtube.com/embed/${id}`;
221     }
222
223     public createDailymotionUrl(id: string) {
224         return `://www.dailymotion.com/embed/video/${id}`;
225     }
226
227 }
228
229
230

```

The code implements a service for managing media. It includes methods to get the last media item from a specific URL, convert URLs to a standard format, and generate URLs for both YouTube and Dailymotion videos.

FIGURE A.7 – Septième extrait du service chargé des médias et des channels

```

view-channels.component.html x

1
2
3  <div class="container">
4    <div class="row">
5
6
7      <div *ngIf="display !== true" class="col-md-offset-5 col-sm-offset-4 loadingTop15">
8        
9      </div>
10
11      <div *ngIf="display === true">
12        <div class="col-sm-12 col-md-12 col-lg-12" *ngFor="let media of medias">
13          <div class="page-header borderTitleHeader">
14            <h1 class="channelName">
15              {{media.nom}}
16            </h1>
17            <button type="button" class="btn btn-default btn-sm" (click)="clickedChanel(media.id)">
18              <span class="glyphicon glyphicon-play-circle"></span>
19              Visionner la chaîne
20            </button>
21
22            </div>
23
24          <!--*ngIf= "showImg[pos]" -->
25          <div class="col-sm-3 col-md-3" *ngFor="let video of media.videos">
26            <div *ngIf="video != null" class="thumbnail">
27              <img [src]=`assets/norsyswebtv/video/thumbnails/${video.id}.jpg` height="200" width="200">
28              <div class="caption">
29                <h3>Vidéo {{pos + 1}}</h3>
30              </div>
31            </div>
32          </div>
33        </div>
34      </div>
35    </div>
36  </div>
37
38  <ann-footer *ngIf="showHeaderFooter"></ann-footer>
39
40
41
42
43

```

FIGURE A.8 – HTML du composant ViewChannel chargé d'afficher les chaines et quatre vidéo

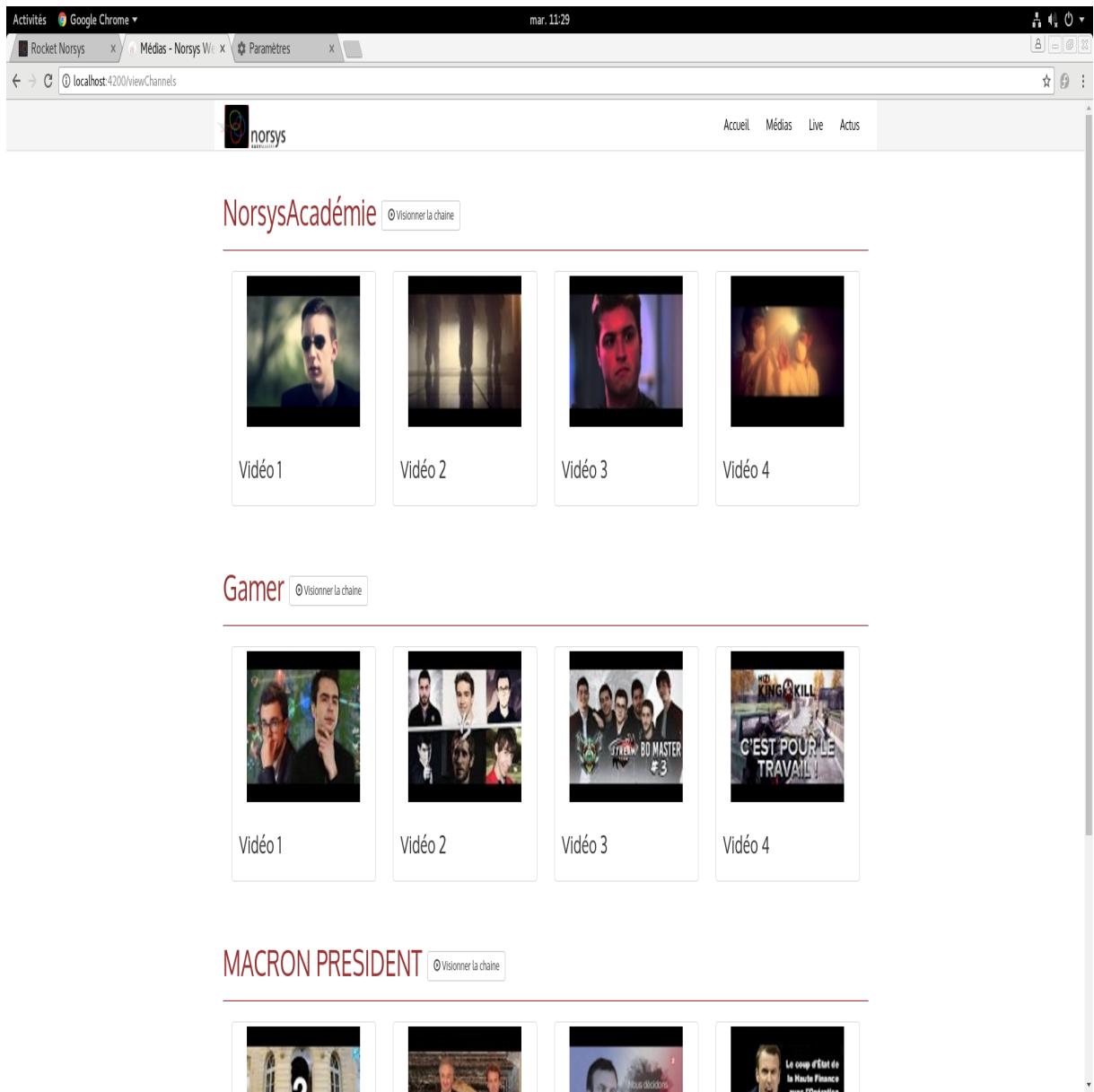


FIGURE A.9 – ViewChannel.ts

FIGURE A.10 – Extrait premier du composant Youtube.ts

youtube-player.component.ts

Compile TypeScript to JavaScript?

```
38  } ngDoCheck(): void {
39  }
40  }
41  ngAfterViewInit(): void {
42  }
43  }
44  }
45  ngOnChanges(changes: SimpleChanges) {
46    if (changes.media && !changes.media.isFirstChange()) {
47      this.media = changes.media.currentValue;
48      if (this.player) {
49        this.player.loadVideoById(this.media);
50      } else {
51        this.setPlayer();
52      }
53    }
54  }
55  }
56  }
57  setPlayer() {
58    this.player = new YT.Player('player', {
59      videoId: this.media,
60      // , controls: 0},
61      playerVars: {autoplay: 1},
62      events: {
63        'onStateChange': (event) => this.zone.run(() => this.onStateChange(event))
64      }
65    });
66  }
67  }
68  onStateChange(event) {
69    if (this.player.getPlayerState() === YT.PlayerState.ENDED) {
70      this.endOfVideo.emit(false);
71    }
72  }
73  }
74  }
75  }
```

FIGURE A.11 – Second extrait du composant Youtube.ts

```
app.routes.ts x navbar.component.html x channel.component.ts x view-channels.component.ts x app-routing.module.ts x media-player.service.ts
Compile TypeScript to JavaScript?
1 import ...
2 export const routerConfig: Routes = [
3   {
4     path: 'login',
5     pathMatch: 'full',
6     component: NoLoginComponent
7   },
8   {
9     path: 'webTV/:channelId',
10    pathMatch: 'full',
11    component: TvComponent,
12    data: { title: 'WebTV - Norsys WebTV'}
13  },
14  {
15    path: '',
16    pathMatch: 'full',
17    component: HomeComponent,
18    data: { title: 'Accueil - Norsys WebTV'}
19  },
20  {
21    path: 'viewChannels',
22    pathMatch: 'full',
23    component: ViewChannelsComponent,
24    data: { title: 'Médias - Norsys WebTV'}
25  },
26  {
27    path: 'channel/:channelId',
28    pathMatch: 'full',
29    component: ChannelComponent,
30    data: { title: 'Chaine - Norsys WebTV'}
31  },
32  {
33    path: 'imageTV/:channelId',
34    pathMatch: 'full',
35    component: ImageTvComponent,
36    data: { title: 'ImageTV - Norsys WebTV'}
37  },
38  {
39    path: 'viewNews',
40    pathMatch: 'full'
41  }
42 ]
```

FIGURE A.12 – Un fichier contenant les routes des différent composant

```

1 import ...
14
15 let DM: any = '';
16
17 @Component({
18   selector: 'app-dailymotion-player',
19   templateUrl: './dailymotion-player.component.html',
20   styleUrls: ['./dailymotion-player.component.scss'],
21 })
22 export class DailymotionPlayerComponent implements OnInit, OnChanges, DoCheck, AfterViewInit {
23
24
25
26   @Output() endOfVideo: EventEmitter<boolean> = new EventEmitter<boolean>();
27   @Input() media;
28   dailyPlayer: any;
29
30   constructor(private element: ElementRef, private zones: NgZone) {
31   }
32
33
34   ngOnInit() {
35     const scriptTag = document.createElement('script');
36     scriptTag.setAttribute('src', 'https://api.dmcn.net/all.js');
37     this.element.nativeElement.appendChild(scriptTag);
38
39     if (window['dmAsyncInit'] === undefined) {
40       window['dmAsyncInit'] = () => {
41         DM = window['DM'];
42         DM.init({apiKey: '25b52ebaa2f4cf0abda', status: true, cookie: true});
43         this.setPlayerDaily();
44       };
45     } else {
46       this.setPlayerDaily();
47     }
48   }
49
50   ngOnChanges(changes: SimpleChanges) {
51     if (changes.media && !changes.media.isFirstChange()) {
52       console.log('first change ' + this.media);
53       this.media = changes.media.currentValue;
54       if (this.dailyPlayer) {
55         this.dailyPlayer.load(this.media);
56       }
57     }
58   }
59
60   ngDoCheck() {
61   }
62
63   ngAfterViewInit() {
64   }
65
66   setPlayerDaily() {
67   }
68
69   load(media) {
70   }
71
72   play() {
73   }
74
75   pause() {
76   }
77
78   stop() {
79   }
80
81   seek(position) {
82   }
83
84   volume(volume) {
85   }
86
87   fullScreen() {
88   }
89
90   close() {
91   }
92
93   destroy() {
94   }
95
96   getDuration() {
97   }
98
99   getVolume() {
100 }
101
102   getMedia() {
103   }
104
105   getProgress() {
106   }
107
108   getSeekPosition() {
109   }
110
111   getFullScreen() {
112   }
113
114   getVolumeLevel() {
115   }
116
117   getMuted() {
118   }
119
120   getIsPlaying() {
121   }
122
123   getIsFullscreened() {
124   }
125
126   getIsMuted() {
127   }
128
129   getIsVolumeLocked() {
130   }
131
132   getIsVolumeMuted() {
133   }
134
135   getIsVolumeMuted() {
136   }
137
138   getIsVolumeMuted() {
139   }
140
141   getIsVolumeMuted() {
142   }
143
144   getIsVolumeMuted() {
145   }
146
147   getIsVolumeMuted() {
148   }
149
150   getIsVolumeMuted() {
151   }
152
153   getIsVolumeMuted() {
154   }
155
156   getIsVolumeMuted() {
157   }
158
159   getIsVolumeMuted() {
160   }
161
162   getIsVolumeMuted() {
163   }
164
165   getIsVolumeMuted() {
166   }
167
168   getIsVolumeMuted() {
169   }
170
171   getIsVolumeMuted() {
172   }
173
174   getIsVolumeMuted() {
175   }
176
177   getIsVolumeMuted() {
178   }
179
180   getIsVolumeMuted() {
181   }
182
183   getIsVolumeMuted() {
184   }
185
186   getIsVolumeMuted() {
187   }
188
189   getIsVolumeMuted() {
190   }
191
192   getIsVolumeMuted() {
193   }
194
195   getIsVolumeMuted() {
196   }
197
198   getIsVolumeMuted() {
199   }
199
200   getIsVolumeMuted() {
201   }
202
203   getIsVolumeMuted() {
204   }
205
206   getIsVolumeMuted() {
207   }
208
209   getIsVolumeMuted() {
210   }
211
212   getIsVolumeMuted() {
213   }
214
215   getIsVolumeMuted() {
216   }
217
218   getIsVolumeMuted() {
219   }
219
220   getIsVolumeMuted() {
221   }
221
222   getIsVolumeMuted() {
223   }
223
224   getIsVolumeMuted() {
225   }
225
226   getIsVolumeMuted() {
227   }
227
228   getIsVolumeMuted() {
229   }
229
230   getIsVolumeMuted() {
231   }
231
232   getIsVolumeMuted() {
233   }
233
234   getIsVolumeMuted() {
235   }
235
236   getIsVolumeMuted() {
237   }
237
238   getIsVolumeMuted() {
239   }
239
240   getIsVolumeMuted() {
241   }
241
242   getIsVolumeMuted() {
243   }
243
244   getIsVolumeMuted() {
245   }
245
246   getIsVolumeMuted() {
247   }
247
248   getIsVolumeMuted() {
249   }
249
250   getIsVolumeMuted() {
251   }
251
252   getIsVolumeMuted() {
253   }
253
254   getIsVolumeMuted() {
255   }
255
256   getIsVolumeMuted() {
257   }
257
258   getIsVolumeMuted() {
259   }
259
260   getIsVolumeMuted() {
261   }
261
262   getIsVolumeMuted() {
263   }
263
264   getIsVolumeMuted() {
265   }
265
266   getIsVolumeMuted() {
267   }
267
268   getIsVolumeMuted() {
269   }
269
270   getIsVolumeMuted() {
271   }
271
272   getIsVolumeMuted() {
273   }
273
274   getIsVolumeMuted() {
275   }
275
276   getIsVolumeMuted() {
277   }
277
278   getIsVolumeMuted() {
279   }
279
280   getIsVolumeMuted() {
281   }
281
282   getIsVolumeMuted() {
283   }
283
284   getIsVolumeMuted() {
285   }
285
286   getIsVolumeMuted() {
287   }
287
288   getIsVolumeMuted() {
289   }
289
290   getIsVolumeMuted() {
291   }
291
292   getIsVolumeMuted() {
293   }
293
294   getIsVolumeMuted() {
295   }
295
296   getIsVolumeMuted() {
297   }
297
298   getIsVolumeMuted() {
299   }
299
300   getIsVolumeMuted() {
301   }
301
302   getIsVolumeMuted() {
303   }
303
304   getIsVolumeMuted() {
305   }
305
306   getIsVolumeMuted() {
307   }
307
308   getIsVolumeMuted() {
309   }
309
310   getIsVolumeMuted() {
311   }
311
312   getIsVolumeMuted() {
313   }
313
314   getIsVolumeMuted() {
315   }
315
316   getIsVolumeMuted() {
317   }
317
318   getIsVolumeMuted() {
319   }
319
320   getIsVolumeMuted() {
321   }
321
322   getIsVolumeMuted() {
323   }
323
324   getIsVolumeMuted() {
325   }
325
326   getIsVolumeMuted() {
327   }
327
328   getIsVolumeMuted() {
329   }
329
330   getIsVolumeMuted() {
331   }
331
332   getIsVolumeMuted() {
333   }
333
334   getIsVolumeMuted() {
335   }
335
336   getIsVolumeMuted() {
337   }
337
338   getIsVolumeMuted() {
339   }
339
340   getIsVolumeMuted() {
341   }
341
342   getIsVolumeMuted() {
343   }
343
344   getIsVolumeMuted() {
345   }
345
346   getIsVolumeMuted() {
347   }
347
348   getIsVolumeMuted() {
349   }
349
350   getIsVolumeMuted() {
351   }
351
352   getIsVolumeMuted() {
353   }
353
354   getIsVolumeMuted() {
355   }
355
356   getIsVolumeMuted() {
357   }
357
358   getIsVolumeMuted() {
359   }
359
360   getIsVolumeMuted() {
361   }
361
362   getIsVolumeMuted() {
363   }
363
364   getIsVolumeMuted() {
365   }
365
366   getIsVolumeMuted() {
367   }
367
368   getIsVolumeMuted() {
369   }
369
370   getIsVolumeMuted() {
371   }
371
372   getIsVolumeMuted() {
373   }
373
374   getIsVolumeMuted() {
375   }
375
376   getIsVolumeMuted() {
377   }
377
378   getIsVolumeMuted() {
379   }
379
380   getIsVolumeMuted() {
381   }
381
382   getIsVolumeMuted() {
383   }
383
384   getIsVolumeMuted() {
385   }
385
386   getIsVolumeMuted() {
387   }
387
388   getIsVolumeMuted() {
389   }
389
390   getIsVolumeMuted() {
391   }
391
392   getIsVolumeMuted() {
393   }
393
394   getIsVolumeMuted() {
395   }
395
396   getIsVolumeMuted() {
397   }
397
398   getIsVolumeMuted() {
399   }
399
400   getIsVolumeMuted() {
401   }
401
402   getIsVolumeMuted() {
403   }
403
404   getIsVolumeMuted() {
405   }
405
406   getIsVolumeMuted() {
407   }
407
408   getIsVolumeMuted() {
409   }
409
410   getIsVolumeMuted() {
411   }
411
412   getIsVolumeMuted() {
413   }
413
414   getIsVolumeMuted() {
415   }
415
416   getIsVolumeMuted() {
417   }
417
418   getIsVolumeMuted() {
419   }
419
420   getIsVolumeMuted() {
421   }
421
422   getIsVolumeMuted() {
423   }
423
424   getIsVolumeMuted() {
425   }
425
426   getIsVolumeMuted() {
427   }
427
428   getIsVolumeMuted() {
429   }
429
430   getIsVolumeMuted() {
431   }
431
432   getIsVolumeMuted() {
433   }
433
434   getIsVolumeMuted() {
435   }
435
436   getIsVolumeMuted() {
437   }
437
438   getIsVolumeMuted() {
439   }
439
440   getIsVolumeMuted() {
441   }
441
442   getIsVolumeMuted() {
443   }
443
444   getIsVolumeMuted() {
445   }
445
446   getIsVolumeMuted() {
447   }
447
448   getIsVolumeMuted() {
449   }
449
450   getIsVolumeMuted() {
451   }
451
452   getIsVolumeMuted() {
453   }
453
454   getIsVolumeMuted() {
455   }
455
456   getIsVolumeMuted() {
457   }
457
458   getIsVolumeMuted() {
459   }
459
460   getIsVolumeMuted() {
461   }
461
462   getIsVolumeMuted() {
463   }
463
464   getIsVolumeMuted() {
465   }
465
466   getIsVolumeMuted() {
467   }
467
468   getIsVolumeMuted() {
469   }
469
470   getIsVolumeMuted() {
471   }
471
472   getIsVolumeMuted() {
473   }
473
474   getIsVolumeMuted() {
475   }
475
476   getIsVolumeMuted() {
477   }
477
478   getIsVolumeMuted() {
479   }
479
480   getIsVolumeMuted() {
481   }
481
482   getIsVolumeMuted() {
483   }
483
484   getIsVolumeMuted() {
485   }
485
486   getIsVolumeMuted() {
487   }
487
488   getIsVolumeMuted() {
489   }
489
490   getIsVolumeMuted() {
491   }
491
492   getIsVolumeMuted() {
493   }
493
494   getIsVolumeMuted() {
495   }
495
496   getIsVolumeMuted() {
497   }
497
498   getIsVolumeMuted() {
499   }
499
500   getIsVolumeMuted() {
501   }
501
502   getIsVolumeMuted() {
503   }
503
504   getIsVolumeMuted() {
505   }
505
506   getIsVolumeMuted() {
507   }
507
508   getIsVolumeMuted() {
509   }
509
510   getIsVolumeMuted() {
511   }
511
512   getIsVolumeMuted() {
513   }
513
514   getIsVolumeMuted() {
515   }
515
516   getIsVolumeMuted() {
517   }
517
518   getIsVolumeMuted() {
519   }
519
520   getIsVolumeMuted() {
521   }
521
522   getIsVolumeMuted() {
523   }
523
524   getIsVolumeMuted() {
525   }
525
526   getIsVolumeMuted() {
527   }
527
528   getIsVolumeMuted() {
529   }
529
530   getIsVolumeMuted() {
531   }
531
532   getIsVolumeMuted() {
533   }
533
534   getIsVolumeMuted() {
535   }
535
536   getIsVolumeMuted() {
537   }
537
538   getIsVolumeMuted() {
539   }
539
540   getIsVolumeMuted() {
541   }
541
542   getIsVolumeMuted() {
543   }
543
544   getIsVolumeMuted() {
545   }
545
546   getIsVolumeMuted() {
547   }
547
548   getIsVolumeMuted() {
549   }
549
550   getIsVolumeMuted() {
551   }
551
552   getIsVolumeMuted() {
553   }
553
554   getIsVolumeMuted() {
555   }
555
556   getIsVolumeMuted() {
557   }
557
558   getIsVolumeMuted() {
559   }
559
560   getIsVolumeMuted() {
561   }
561
562   getIsVolumeMuted() {
563   }
563
564   getIsVolumeMuted() {
565   }
565
566   getIsVolumeMuted() {
567   }
567
568   getIsVolumeMuted() {
569   }
569
570   getIsVolumeMuted() {
571   }
571
572   getIsVolumeMuted() {
573   }
573
574   getIsVolumeMuted() {
575   }
575
576   getIsVolumeMuted() {
577   }
577
578   getIsVolumeMuted() {
579   }
579
580   getIsVolumeMuted() {
581   }
581
582   getIsVolumeMuted() {
583   }
583
584   getIsVolumeMuted() {
585   }
585
586   getIsVolumeMuted() {
587   }
587
588   getIsVolumeMuted() {
589   }
589
590   getIsVolumeMuted() {
591   }
591
592   getIsVolumeMuted() {
593   }
593
594   getIsVolumeMuted() {
595   }
595
596   getIsVolumeMuted() {
597   }
597
598   getIsVolumeMuted() {
599   }
599
600   getIsVolumeMuted() {
601   }
601
602   getIsVolumeMuted() {
603   }
603
604   getIsVolumeMuted() {
605   }
605
606   getIsVolumeMuted() {
607   }
607
608   getIsVolumeMuted() {
609   }
609
610   getIsVolumeMuted() {
611   }
611
612   getIsVolumeMuted() {
613   }
613
614   getIsVolumeMuted() {
615   }
615
616   getIsVolumeMuted() {
617   }
617
618   getIsVolumeMuted() {
619   }
619
620   getIsVolumeMuted() {
621   }
621
622   getIsVolumeMuted() {
623   }
623
624   getIsVolumeMuted() {
625   }
625
626   getIsVolumeMuted() {
627   }
627
628   getIsVolumeMuted() {
629   }
629
630   getIsVolumeMuted() {
631   }
631
632   getIsVolumeMuted() {
633   }
633
634   getIsVolumeMuted() {
635   }
635
636   getIsVolumeMuted() {
637   }
637
638   getIsVolumeMuted() {
639   }
639
640   getIsVolumeMuted() {
641   }
641
642   getIsVolumeMuted() {
643   }
643
644   getIsVolumeMuted() {
645   }
645
646   getIsVolumeMuted() {
647   }
647
648   getIsVolumeMuted() {
649   }
649
650   getIsVolumeMuted() {
651   }
651
652   getIsVolumeMuted() {
653   }
653
654   getIsVolumeMuted() {
655   }
655
656   getIsVolumeMuted() {
657   }
657
658   getIsVolumeMuted() {
659   }
659
660   getIsVolumeMuted() {
661   }
661
662   getIsVolumeMuted() {
663   }
663
664   getIsVolumeMuted() {
665   }
665
666   getIsVolumeMuted() {
667   }
667
668   getIsVolumeMuted() {
669   }
669
670   getIsVolumeMuted() {
671   }
671
672   getIsVolumeMuted() {
673   }
673
674   getIsVolumeMuted() {
675   }
675
676   getIsVolumeMuted() {
677   }
677
678   getIsVolumeMuted() {
679   }
679
680   getIsVolumeMuted() {
681   }
681
682   getIsVolumeMuted() {
683   }
683
684   getIsVolumeMuted() {
685   }
685
686   getIsVolumeMuted() {
687   }
687
688   getIsVolumeMuted() {
689   }
689
690   getIsVolumeMuted() {
691   }
691
692   getIsVolumeMuted() {
693   }
693
694   getIsVolumeMuted() {
695   }
695
696   getIsVolumeMuted() {
697   }
697
698   getIsVolumeMuted() {
699   }
699
700   getIsVolumeMuted() {
701   }
701
702   getIsVolumeMuted() {
703   }
703
704   getIsVolumeMuted() {
705   }
705
706   getIsVolumeMuted() {
707   }
707
708   getIsVolumeMuted() {
709   }
709
710   getIsVolumeMuted() {
711   }
711
712   getIsVolumeMuted() {
713   }
713
714   getIsVolumeMuted() {
715   }
715
716   getIsVolumeMuted() {
717   }
717
718   getIsVolumeMuted() {
719   }
719
720   getIsVolumeMuted() {
721   }
721
722   getIsVolumeMuted() {
723   }
723
724   getIsVolumeMuted() {
725   }
725
726   getIsVolumeMuted() {
727   }
727
728   getIsVolumeMuted() {
729   }
729
730   getIsVolumeMuted() {
731   }
731
732   getIsVolumeMuted() {
733   }
733
734   getIsVolumeMuted() {
735   }
735
736   getIsVolumeMuted() {
737   }
737
738   getIsVolumeMuted() {
739   }
739
740   getIsVolumeMuted() {
741   }
741
742   getIsVolumeMuted() {
743   }
743
744   getIsVolumeMuted() {
745   }
745
746   getIsVolumeMuted() {
747   }
747
748   getIsVolumeMuted() {
749   }
749
750   getIsVolumeMuted() {
751   }
751
752   getIsVolumeMuted() {
753   }
753
754   getIsVolumeMuted() {
755   }
755
756   getIsVolumeMuted() {
757   }
757
758   getIsVolumeMuted() {
759   }
759
760   getIsVolumeMuted() {
761   }
761
762   getIsVolumeMuted() {
763   }
763
764   getIsVolumeMuted() {
765   }
765
766   getIsVolumeMuted() {
767   }
767
768   getIsVolumeMuted() {
769   }
769
770   getIsVolumeMuted() {
771   }
771
772   getIsVolumeMuted() {
773   }
773
774   getIsVolumeMuted() {
775   }
775
776   getIsVolumeMuted() {
777   }
777
778   getIsVolumeMuted() {
779   }
779
780   getIsVolumeMuted() {
781   }
781
782   getIsVolumeMuted() {
783   }
783
784   getIsVolumeMuted() {
785   }
785
786   getIsVolumeMuted() {
787   }
787
788   getIsVolumeMuted() {
789   }
789
790   getIsVolumeMuted() {
791   }
791
792   getIsVolumeMuted() {
793   }
793
794   getIsVolumeMuted() {
795   }
795
796   getIsVolumeMuted() {
797   }
797
798   getIsVolumeMuted() {
799   }
799
800   getIsVolumeMuted() {
801   }
801
802   getIsVolumeMuted() {
803   }
803
804   getIsVolumeMuted() {
805   }
805
806   getIsVolumeMuted() {
807   }
807
808   getIsVolumeMuted() {
809   }
809
810   getIsVolumeMuted() {
811   }
811
812   getIsVolumeMuted() {
813   }
813
814   getIsVolumeMuted() {
815   }
815
816   getIsVolumeMuted() {
817   }
817
818   getIsVolumeMuted() {
819   }
819
820   getIsVolumeMuted() {
821   }
821
822   getIsVolumeMuted() {
823   }
823
824   getIsVolumeMuted() {
825   }
825
826   getIsVolumeMuted() {
827   }
827
828   getIsVolumeMuted() {
829   }
829
830   getIsVolumeMuted() {
831   }
831
832   getIsVolumeMuted() {
833   }
833
834   getIsVolumeMuted() {
835   }
835
836   getIsVolumeMuted() {
837   }
837
838   getIsVolumeMuted() {
839   }
839
840   getIsVolumeMuted() {
841   }
841
842   getIsVolumeMuted() {
843   }
843
844   getIsVolumeMuted() {
845   }
845
846   getIsVolumeMuted() {
847   }
847
848   getIsVolumeMuted() {
849   }
849
850   getIsVolumeMuted() {
851   }
851
852   getIsVolumeMuted() {
853   }
853
854   getIsVolumeMuted() {
855   }
855
856   getIsVolumeMuted() {
857   }
857
858   getIsVolumeMuted() {
859   }
859
860   getIsVolumeMuted() {
861   }
861
862   getIsVolumeMuted() {
863   }
863
864   getIsVolumeMuted() {
865   }
865
866   getIsVolumeMuted() {
867   }
867
868   getIsVolumeMuted() {
869   }
869
870   getIsVolumeMuted() {
871   }
871
872   getIsVolumeMuted() {
873   }
873
874   getIsVolumeMuted() {
875   }
875
876   getIsVolumeMuted() {
877   }
877
878   getIsVolumeMuted() {
879   }
879
880   getIsVolumeMuted() {
881   }
881
882   getIsVolumeMuted() {
883   }
883
884   getIsVolumeMuted() {
885   }
885
886   getIsVolumeMuted() {
887   }
887
888   getIsVolumeMuted() {
889   }
889
890   getIsVolumeMuted() {
891   }
891
892   getIsVolumeMuted() {
893   }
893
894   getIsVolumeMuted() {
895   }
895
896   getIsVolumeMuted() {
897   }
897
898   getIsVolumeMuted() {
899   }
899
900   getIsVolumeMuted() {
901   }
901
902   getIsVolumeMuted() {
903   }
903
904   getIsVolumeMuted() {
905   }
905
906   getIsVolumeMuted() {
907   }
907
908   getIsVolumeMuted() {
909   }
909
910   getIsVolumeMuted() {
911   }
911
912   getIsVolumeMuted() {
913   }
913
914   getIsVolumeMuted() {
915   }
915
916   getIsVolumeMuted() {
917   }
917
918   getIsVolumeMuted() {
919   }
919
920   getIsVolumeMuted() {
921   }
921
922   getIsVolumeMuted() {
923   }
923
924   getIsVolumeMuted() {
925   }
925
926   getIsVolumeMuted() {
927   }
927
928   getIsVolumeMuted() {
929   }
929
930   getIsVolumeMuted() {
931   }
931
932   getIsVolumeMuted() {
933   }
933
934   getIsVolumeMuted() {
935   }
935
936   getIsVolumeMuted() {
937   }
937
938   getIsVolumeMuted() {
939   }
939
940   getIsVolumeMuted() {
941   }
941
942   getIsVolumeMuted() {
943   }
943
944   getIsVolumeMuted() {
945   }
945
946   getIsVolumeMuted() {
947   }
947
948   getIsVolumeMuted() {
949   }
949
950   getIsVolumeMuted() {
951   }
951
952   getIsVolumeMuted() {
953   }
953
954   getIsVolumeMuted() {
955   }
955
956   getIsVolumeMuted() {
957   }
957
958   getIsVolumeMuted() {
959   }
959
960   getIsVolumeMuted() {
961   }
961
962   getIsVolumeMuted() {
963   }
963
964   getIsVolumeMuted() {
965   }
965
966   getIsVolumeMuted() {
967   }
967
968   getIsVolumeMuted() {
969   }
969
970   getIsVolumeMuted() {
971   }
971
972   getIsVolumeMuted() {
973   }
973
974   getIsVolumeMuted() {
975   }
975
976   getIsVolumeMuted() {
977   }
977
978   getIsVolumeMuted() {
979   }
979
980   getIsVolumeMuted() {
981   }
981
982   getIsVolumeMuted() {
983   }
983
984   getIsVolumeMuted() {
985   }
985
986   getIsVolumeMuted() {
987   }
987
988   getIsVolumeMuted() {
989   }
989
990   getIsVolumeMuted() {
991   }
991
992   getIsVolumeMuted() {
993   }
993
994   getIsVolumeMuted() {
995   }
995
996   getIsVolumeMuted() {
997   }
997
998   getIsVolumeMuted() {
999   }
999
1000   getIsVolumeMuted() {
1001   }
1001
1002   getIsVolumeMuted() {
1003   }
1003
1004   getIsVolumeMuted() {
1005   }
1005
1006   getIsVolumeMuted() {
1007   }
1007
1008   getIsVolumeMuted() {
1009   }
1009
1010   getIsVolumeMuted() {
1011   }
1011
1012   getIsVolumeMuted() {
1013   }
1013
1014   getIsVolumeMuted() {
1015   }
1015
1016   getIsVolumeMuted() {
1017   }
1017
1018   getIsVolumeMuted() {
1019   }
1019
1020   getIsVolumeMuted() {
1021   }
1021
1022   getIsVolumeMuted() {
1023   }
1023
1024   getIsVolumeMuted() {
1025   }
1025
1026   getIsVolumeMuted() {
1027   }
1027
1028   getIsVolumeMuted() {
1029   }
1029
1030   getIsVolumeMuted() {
1031   }
1031
1032   getIsVolumeMuted() {
1033   }
1033
1034   getIsVolumeMuted() {
1035   }
1035
1036   getIsVolumeMuted() {
1037   }
1037
1038   getIsVolumeMuted() {
1039   }
1039
1040   getIsVolumeMuted() {
1041   }
1041
1042   getIsVolumeMuted() {
1043   }
1043
1044   getIsVolumeMuted() {
1045   }
1045
1046   getIsVolumeMuted() {
1047   }
1047
1048   getIsVolumeMuted() {
1049   }
1049
1050   getIsVolumeMuted() {
1051   }
1051
1052   getIsVolumeMuted() {
1053   }
1053
1054   getIsVolumeMuted() {
1055   }
1055
1056   getIsVolumeMuted() {
1057   }
1057
1058   getIsVolumeMuted() {
1059   }
1059
1060   getIsVolumeMuted() {
1061   }
1061
1062   getIsVolumeMuted() {
1063   }
1063
1064   getIsVolumeMuted() {
1065   }
1065
1066   getIsVolumeMuted() {
1067   }
1067
1068   getIsVolumeMuted() {
1069   }
1069
1070   getIsVolumeMuted() {
1071   }
1071
1072   getIsVolumeMuted() {
1073   }
1073
1074   getIsVolumeMuted() {
1075   }
1075
1076   getIsVolumeMuted() {
1077   }
1077
1078   getIsVolumeMuted() {
1079   }
1079
1080   getIsVolumeMuted() {
1081   }
1081
1082   getIsVolumeMuted() {
1083   }
1083
1084   getIsVolumeMuted() {
1085   }
1085
1086   getIsVolumeMuted() {
1087   }
1087
1088   getIsVolumeMuted() {
1089   }
1089
1090   getIsVolumeMuted() {
1091   }
1091
1092   getIsVolumeMuted() {
1093   }
1093
1094   getIsVolumeMuted() {
1095   }
1095
1096   getIsVolumeMuted() {
1097   }
1097
1098   getIsVolumeMuted() {
1099   }
1099
1100   getIsVolumeMuted() {
1101   }
1101
1102   getIsVolumeMuted() {
1103   }
1103
1104   getIsVolumeMuted() {
1105   }
1105
1106   getIsVolumeMuted() {
1107   }
1107
1108   getIsVolumeMuted() {
1109   }
1109
1110   getIsVolumeMuted() {
1111   }
1111
1112   getIsVolumeMuted() {
1113   }
1113
1114   getIsVolumeMuted() {
1115   }
1115
1116   getIsVolumeMuted() {
1117   }
1117
1118   getIsVolumeMuted() {
1119   }
1119
1120   getIsVolumeMuted() {
1121   }
1121
1122   getIsVolumeMuted() {
1123   }
1123
1124   getIsVolumeMuted() {
1125   }
1125
1126   getIsVolumeMuted() {
1127   }
1127
1128   getIsVolumeMuted() {
1129   }
1129
1130   getIsVolumeMuted() {
1131   }
1131
1132   getIsVolumeMuted() {
1133   }
1133
1134   getIsVolumeMuted() {
1135   }
1135
1136   getIsVolumeMuted() {
1137   }
1137
1138   getIsVolumeMuted() {
1139   }
1139
1140   getIsVolumeMuted() {
1141   }
1141
1142   getIsVolumeMuted() {
1143   }
1143
1144   getIsVolumeMuted() {
1145   }
1145
1146   getIsVolumeMuted() {
1147   }
1147
1148   getIsVolumeMuted() {
1149   }
1149
1150   getIsVolumeMuted() {
1151   }
1151
1152   getIsVolumeMuted() {
1153   }
1153
1154   getIsVolumeMuted() {
1155   }
1155
1156   getIsVolumeMuted() {
1157   }
1157
1158   getIsVolumeMuted() {
1159   }
1159
1160   getIsVolumeMuted() {
1161   }
1161
1162   getIsVolumeMuted() {
1163   }
1163
1
```

```

dailymotion-player.component.ts
Compile TypeScript to JavaScript?
OK No Configure

47 }
48 }
49
50 ↗ ngOnChanges(changes: SimpleChanges) {
51   if (changes.media && !changes.media.isFirstChange()) {
52     console.log('first change ' + this.media);
53     this.media = changes.media.currentValue;
54     if (this.dailyPlayer) {
55       this.dailyPlayer.load(this.media);
56       console.log('load', {autoplay: true});
57     } else {
58       console.log('set');
59       this.setPlayerDaily();
60     }
61   }
62 }
63
64 ↗ ngDoCheck(): void {
65 }
66
67 ↗ ngAfterViewInit(): void {
68 }
69
70 ↗ setPlayerDaily() {
71   this.dailyPlayer = DM.player('player', {
72     video: this.media,
73     params: {
74       autoplay: true
75     }
76   });
77   this.dailyPlayer.addEventListener('video_end', () => this.sendToTv(), event);
78 }
79
80 ↗ sendToTv() {
81   this.endOfVideo.emit(false);
82 }
83
84

```

FIGURE A.14 – Second extrait du composant Dailymotion.ts

The screenshot shows a code editor window with the file 'tv.component.html' open. The code is written in HTML and includes several Angular directives like *ngIf and *ngSwitchCase. The editor has syntax highlighting and shows line numbers from 1 to 35. The code structure is as follows:

```
tv.component.html
div.container-fluid div.row div.embed-responsive embed-responsive-16by9 div
1 <app-navbar *ngIf="showHeaderFooter"></app-navbar>
2
3 <div class="container-fluid">
4
5   <div class="row">
6     <div class="embed-responsive embed-responsive-16by9">
7
8       <!--<div [ngSwitch]="myMedia[currentId].type">-->
9       <div [ngSwitch]="myMedia[currentId].type" *ngIf="myMedia[currentId] != undefined">
10
11         <app-youtube-player *ngSwitchCase="youtube" [media]="currentURL" (endOfVideo)="nextVideo($event)">
12           </app-youtube-player>
13
14         <app-dailymotion-player *ngSwitchCase="dailymotion" [media]="currentURL" (endOfVideo)="nextVideo($event)">
15           </app-dailymotion-player>
16
17         <app-image *ngSwitchCase="image" [media]="currentURL" (endOfVideo)="nextVideo($event)">
18           </app-image>
19
20       </div>
21
22     </div>
23
24   </div>
25
26 </div>
27
28
29 <app-news-bar></app-news-bar>
30
31 </div>
32 <app-footer *ngIf="showHeaderFooter"></app-footer>
33
34
35
```

FIGURE A.15 – HTML du composant TV

FIGURE A.16 – Composant TV.ts

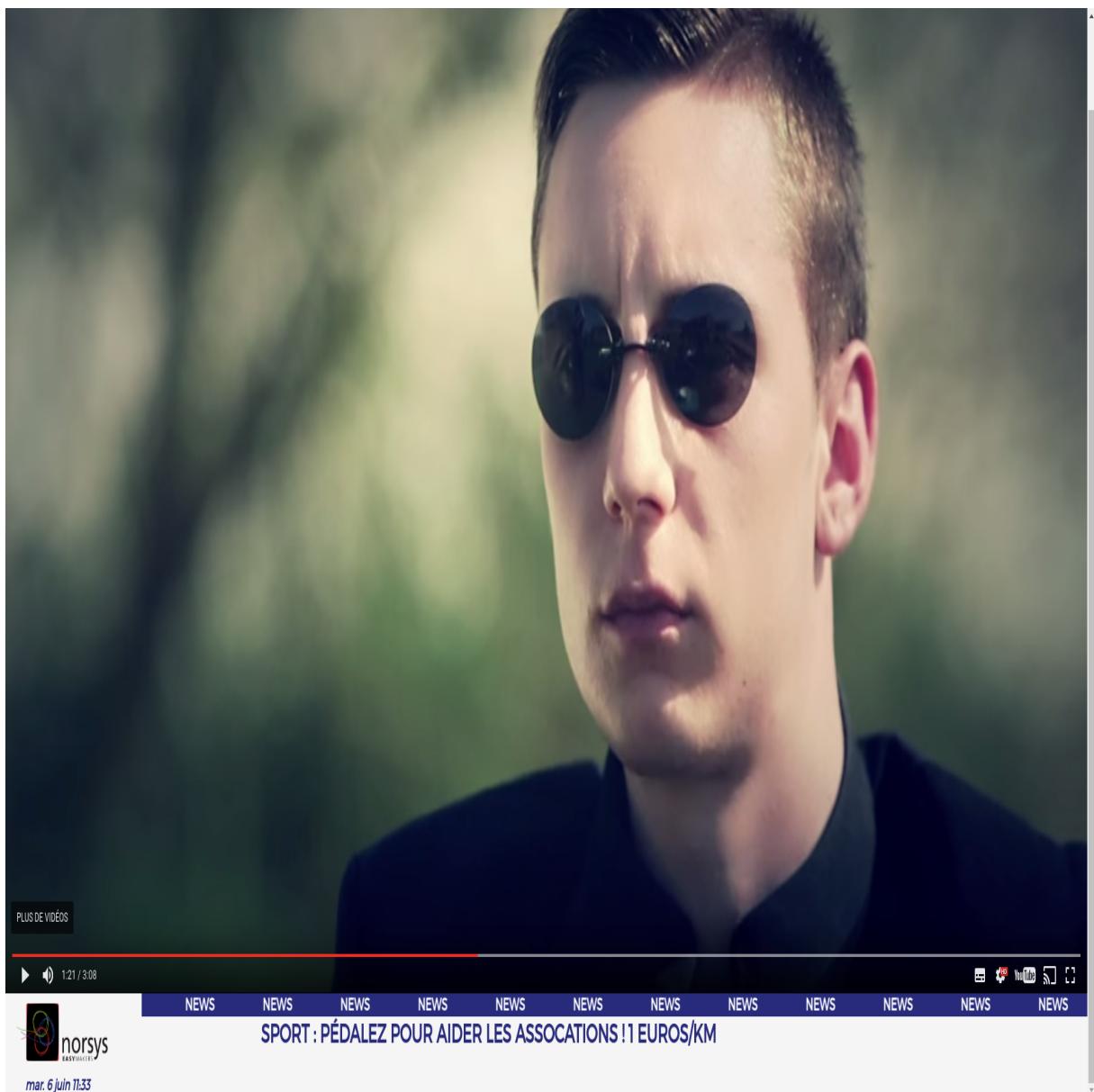


FIGURE A.17 – Un Web-TV qui fonctionne