Table des matières

[1. Introduction 3](#_Toc45994339)

[2. Présentation de l'entreprise : [Titre perso] 3](#_Toc45994340)

[2.1. Amiltone 3](#_Toc45994341)

[2.2. Mon point de vue 3](#_Toc45994342)

[2.3. L’équipe de travail 3](#_Toc45994343)

[3. Analyse du contexte : Le projet AmilApp 3](#_Toc45994344)

[3.1. AmilApp 3](#_Toc45994345)

[3.1.1. Les news 4](#_Toc45994346)

[3.1.2. Les événements 4](#_Toc45994347)

[3.1.3. Les sondages 4](#_Toc45994348)

[3.1.4. Les alertes 5](#_Toc45994349)

[3.1.5. Les améliorations futures 5](#_Toc45994350)

[3.2. Firebase 5](#_Toc45994351)

[3.2.1. Pourquoi l’avoir choisi ? 5](#_Toc45994352)

[3.2.2. Avantages et inconvénients de Firebase 5](#_Toc45994353)

[4. Problématique : Comment mettre en place une solution web réutilisable, modulaire et évolutive ? 6](#_Toc45994354)

[4.1. Les micro-services 6](#_Toc45994355)

[4.1.1. Définition 7](#_Toc45994356)

[4.1.2. Avantages 7](#_Toc45994357)

[4.1.3. Inconvénients 7](#_Toc45994358)

[4.1.4. Pourquoi les avoir choisis ? 7](#_Toc45994359)

[4.2. Le starter-kit 7](#_Toc45994360)

[5. Méthodes habituellement utilisées pour une situation présentant des similitudes : [Titre perso] 8](#_Toc45994361)

[5.1. Spring Boot 8](#_Toc45994362)

[5.1.1. Fonctionnalités 8](#_Toc45994363)

[5.1.2. Avantages et inconvénients par rapport au projet 8](#_Toc45994364)

[5.2. Hackathon-starter 9](#_Toc45994365)

[5.2.1. Fonctionnalités 9](#_Toc45994366)

[5.2.2. Avantages et inconvénients par rapport au projet 9](#_Toc45994367)

[5.3. Autres solutions 10](#_Toc45994368)

[6. Exposé des décisions prises et des interventions menées par le stagiaire pour résoudre le problème : [Titre perso] 10](#_Toc45994369)

[7. Démonstration d’une originalité dans l’élaboration et la mise en œuvre de la solution : [Titre perso] 10](#_Toc45994370)

[8. Analyse de l’approche choisie : [Titre perso] 10](#_Toc45994371)

[9. Réflexion sur le stage et le mémoire : [Titre perso] 10](#_Toc45994372)

[10. Conclusion 10](#_Toc45994373)

# Introduction

# Présentation de l'entreprise : [Titre perso]

## Amiltone

## Mon point de vue

## L’équipe de travail

# Analyse du contexte : Le projet AmilApp

## AmilApp

Dans cette partie, je vais présenter le projet sur lequel je travaille depuis bientôt deux ans. Comme je ne pars pas en mission chez les clients, je suis le seul développeur à être resté sur le projet depuis le début et les nouvelles personnes qui arrivent sur le projet sont, soit de nouveaux Amiltoniens qui vont se former aux technologies utilisées sur l’application, soit des Amiltoniens en inter-contrat.

La bonne entente ainsi que la bonne ambiance étant des valeurs fortes d’Amiltone, l’entreprise se doit d’avoir un moyen pratique de communiquer avec ses collaborateurs. AmilApp permet au département communication d’Amiltone de partager différentes informations avec les collaborateurs. L’application est divisée en plusieurs parties : les News (actualités), les Événements, les Sondages et les Alertes. L’avantage de cette application est de pouvoir centraliser toutes informations en lien avec Amiltone et permet de rendre les différents posts publiés sur l’application, habituellement communiqués via Teams ou par courriel, plus visibles pour tous les collaborateurs d’Amiltone. Les collaborateurs sont censé pouvoir se connecter avec le même compte qu’ils utilisent en entreprise. Mais à cause de la complexité du développement d’une telle fonctionnalité compatible avec Firebase, elle a été repoussée et c’est l’authentification intégrée de Firebase qui a été utilisée.

Après un développement de plus d’un an, AmilApp sort sur iOS, Android et web en version 1.0. Le site web reprend les fonctionnalités des applications mobiles. En plus de celles-ci, AmilApp dispose d’un backoffice administrateur pour pouvoir ajouter du contenu sur l’application. Pendant toute cette année de développement, j’étais en charge des parties backoffice administrateur et site web, codées en Angular 8. AmilApp ne dispose pas de backend à proprement parlé puisque nous utilisons la solution cloud de Google Firebase. J’expliquerai un peu plus tard pour nous l’avons choisi ainsi que ces avantages et inconvénients.

Je vais présenter ci-dessous les fonctionnalités principales d’AmilApp.

### Les news

Les News, ou Actualité, permettent d’informer les utilisateurs des dernières nouvelles chez Amiltone ou en lien avec les nouvelles technologies. Actuellement, une News peut afficher une image de présentation, des pièces jointes au format PDF et peut être reliée à un sondage ou à un événement. Plus tard, il sera possible d’y ajouter une galerie d’images, de cibler les personnes qui recevront la News, et même la possibilité de « liker » celle-ci. À la publication d’une News, les collaborateurs d’Amiltone ayant téléchargé l’application reçoivent une notification sur leur téléphone.

### Les événements

C’est dans la partie Événement qu’Amiltone communique sur les soirées d’agence, sorties ski, ou, plus local, les sessions sportives ou cinéma. Les administrateurs ont la possibilité de sélectionner une liste d’invités qui recevront une notification sur leur téléphone. Que ce soit l’application mobile ou le site web, les collaborateurs ont la possibilité d’indiquer leur présence ou non, tant que la date limite de réponse n’est pas dépassée.

### Les sondages

Les Sondages permettent à la Communication de l’entreprise de poser des questions aux collaborateurs pour ensuite avoir accès à des statistiques. Un sondage peut être de type ouvert, les utilisateurs répondent avec leurs propres mots ce qu’ils veulent ; de type multiple, les utilisateurs doivent choisir une ou plusieurs réponses prédéfinies ; ou de type date, les utilisateurs ont la possibilité de sélectionner une ou plusieurs dates prédéfinies, encore une fois. Les administrateurs ont la possibilité de cibler le sondage en sélectionnant une liste de personnes, ou d’agence, qui pourront participer à celui-ci.

### Les alertes

Les Alertes permettent de lancer des rappels aux collaborateurs. Les administrateurs doivent sélectionner une liste de personnes qui la recevront.

### Les améliorations futures

Il y a encore beaucoup d’améliorations de prévues pour AmilApp : les activités, les enquêtes, la possibilité pour les collaborateurs d’Amiltone de remplir leur rapport d’activité mensuel, une refonte complète du design de la partie administrateur du site web. Mais pour le moment, la prochaine grosse amélioration prévue est la création d’une API pour remplacer Firebase. J’explique ci-dessous pourquoi la décision de se séparer du service de Google a été prise.

## Firebase

Firebase est une solution proposée par Google qui permet la création de backend simplement et rapidement, le tout scalable et performant. Il permet aussi de gérer le stockage de fichier dans le cloud et prend en charge la gestion de comptes utilisateur.

Firestore est le service qui sert de base de données pour AmilApp et propose une actualisation des données en temps réel. Il est basé sur du NoSQL et ressemble beaucoup à MongoDB.

### Pourquoi l’avoir choisi ?

AmilApp n’était pas le premier projet à utiliser Firebase chez Amiltone. AmilCar, une petite application permettant la gestion du parc automobile d’Amiltone, a été le premier essai avec Firebase et a été développé en une semaine. Firebase a donc été choisi pour gagner du temps. Au début du projet, nous étions trois développeurs en alternance à travailler dessus et personne n’était là en même temps. S’il avait fallu qu’on développe le backend, le projet aurait été beaucoup plus long, sachant qu’à trois, on comptabilisait 26 jours de travail par mois au lieu de 60 si on avait été à temps plein.

### Avantages et inconvénients de Firebase

Firebase permet la mise en place d’un backend très rapidement et prend en charge l’authentification, qui est une partie souvent redondante et lourde à développer dans n’importe quelle application. De plus, l’utilisation de Firestore est assez simple et la documentation sur internet est plutôt bien faite.

Mais bien que Firebase soit très pratique pour une petite application, la solution proposée par Google ne semble pas assez mature lorsqu’il s’agit d’avoir des interactions plus complexes avec Firestore. En effet, sur AmilApp, on a besoin de récupérer certaines données avec des tris assez précis, et avec une API normale, lorsque que l’on fait une requête, les données reçues sont déjà ordonnées comme on le souhaite. Or, avec Firebase, certaines opérations, pourtant assez basiques dans un langage comme le SQL ou le NoSQL, n’existent tout simplement pas sur Firestore. J’ai donc été obligé de lancer plusieurs requêtes en même temps avec des tris différents pour avoir accès aux données que je souhaitais avoir précisément. Le code s’en retrouve beaucoup plus complexe dès lors que l’on souhaite intégrer l’actualisation des données en temps réel ou le « lazy-loading », pratique qui consiste à attendre une action de l’utilisateur pour charger des données et qui permet de grandement accélérer le chargement d’une application.

Enfin, la gestion du cache serveur de Firebase ne fonctionne correctement seulement si l’application est hébergée sur leur cloud, ce qui n’est pas le cas d’AmilApp. J’ai donc dû gérer moi-même le cache côté navigateur, mais certains bogues persistent à cause du lazy-loading, des requêtes envoyées simultanément pour récupérer les données correctement et l’actualisation des données en temps réel.

Avec tous ces inconvénients, le plus simple d’après moi est de se séparer de Firebase et créer notre propre API.

# Problématique : Comment mettre en place une solution web réutilisable, modulaire et évolutive ?

J’explique dans cette partie

## Les micro-services

Le but de cette partie est de faire un tour des différentes définitions et structurations de l’architecture micro-services.

Le but de l’architecture micro-services est de découper une application en plusieurs petits services, autonomes et indépendants les uns des autres, contrairement à une architecture monolithique où l’application forme un tout indissociable. La plupart du temps, chaque service expose une API REST qui est accessible par l’interface utilisateur ou n’importe quel autre micro-services. Chaque service a alors son propre environnement et peut être complètement différents des autres, que ce soit au niveau des technologies utilisées ou des versions de celles-ci. On peut très bien imaginer, par exemple, un micro-service codé en NodeJS avec une base de données NoSQL MongoDB, qui tourne sur un serveur Nginx et un autre micro-service codé en Java avec une base de données Postgresql, démarrée sur un serveur Apache. En général, on utilise Docker pour démarrer les micro-services.

### Définition

Avec le temps, les applications deviennent de plus en plus complexes au fur et à mesure que de nouvelles fonctionnalités sont ajoutées aux existantes. Avec cette complexité arrivent plusieurs désavantages :

* Il devient plus difficile de faire évoluer le projet proprement. Chaque évolution apportant son lot de nouvelles interactions avec le code existant, même avec des tests de code solides, le projet devient de moins en moins lisible et maintenable.
* Le projet perd en fiabilité à cause des potentiels « fix » qui rendent les fonctionnalités assez instables.
* Lors de l’ajout d’une nouvelle fonctionnalité, il se peut qu’il soit plus pratique de changer un morceau de la structure du projet. Mais plus le projet est gros, plus il est difficile de changer cette structure sans risquer de casser d’anciennes fonctionnalités.

Le but des micro-services est de diviser un gros projet en plusieurs petits projets, que l’on appelle **micro-services**. Voici une définition simple : « Microservices are small, autonomous services that work together »[[1]](#footnote-1) (les micro-services sont des petits services autonomes qui travaillent ensemble). Elle permet, sans être trop technique, de mettre en avant les deux principes fondateurs des micro-services :

* L’existence de services : c’est-à-dire de modules dont les finalités sont différentes. Un service peut travailler avec un autre service, mais chacun aura son propre objectif.
* Les services sont petits. De manière générale, un micro-service a une seule fonction.

### Avantages

### Inconvénients

### Pourquoi les avoir choisis ?

## Le starter-kit

# Méthodes habituellement utilisées pour une situation présentant des similitudes : [Titre perso]

Dans cette partie, je présente deux solutions : Spring Boot et Hackathon-starter, qui sont largement utilisées au sein de la communauté de développeurs pour démarrer un projet dans de plus brefs délais.

## Spring Boot

Spring est un framework open source bien connu et très utilisé dans le monde du Java pour les fonctionnalités qu’il propose : l’injection de dépendances, une gestion des instances de classes, des outils pour les applications web et encore beaucoup d’autres. Mais sa configuration reste très complexe et peut prendre beaucoup de temps. Les équipes de Spring ont alors créé Spring Boot.

### Fonctionnalités

Spring Boot est donc un framework qui rend la configuration de Spring plus rapide et beaucoup plus simple, en fournissant différentes annotations à placer dans le code, comme **@EnableAutoConfiguration**, qui, comme son nom l’indique, active l’auto-configuration pour toutes les dépendances présentes dans l’application.

Spring Boot est utilisé pour créer des applications micro-services et permet donc d’obtenir une application packagée, complètement autonome, comme le sont les micro-services, et prête pour la production (production ready) dès la création du projet. De plus, le framework intègre directement un serveur Web dans l’application, Apache Tomcat par exemple, qui sera démarré automatiquement au lancement de celle-ci.

Pour la création d’une application via Spring Boot, le site web Spring Initializr (https://start.spring.io) permet de générer très rapidement la structure du projet tout en y incluant les dépendances que vous aurez sélectionné sur le site.

Enfin, l’application contiendra automatiquement différentes routes pour afficher des statistiques (métrique) sur celle-ci, telles que l’utilisation de la mémoire, la liste des requêtes http, l’état de santé de l’application.

### Avantages et inconvénients par rapport au projet

Spring Boot bénéficie d’énormément d’avantages sur tous les points :

* La possibilité de créer un projet fonctionnel et production ready en très peu de temps grâce à Spring Initializr
* Axé sur les micro-services et extrêmement modulable grâce à l’auto-configuration des dépendances

## Hackathon-starter

Le hackathon est un événement pendant lequel des équipes de développeur se réunissent pour créer des projets informatiques dans un temps imparti souvent très cours : un weekend, une journée ou même une nuit. L’objectif est alors de coder le plus rapidement possible un logiciel ou une application qui sorte du lot. C’est donc tout naturellement que j’ai fais mes recherches sur l’existence d’un starter-kit destiné aux hackathons, qui allait permettre de gagner un temps précieux pendant ces événements. Mon choix s’est porté sur Hackaton-starter.

### Fonctionnalités

Hackathon-starter est un projet starter-kit écrit en JavaScript et NodeJS, hébergé sur la plateforme Github. Avec ses trente milles étoiles, il fait partie des projets qui en recueille le plus sur la plateforme. De base, il inclut énormément de fonctionnalités déjà codées :

* Toute la partie authentification, que ce soit en local, avec l’adresse email et le mot de passe, ou avec OAuth 1.0a et OAuth 2.0 pour s’authentifier via différents réseaux sociaux.
* Une gestion des notifications
* Une gestion des comptes utilisateurs
* Des pages web prédéfinies comme la page de profil par exemple

Le starter-kit utilise MongoDB pour la base de données et est compatible avec Docker.

### Avantages et inconvénients par rapport au projet

Les avantages à utiliser ce starter-kit sont assez nombreux, tout est déjà codé : la connexion avec la base de données, la partie authentification et même quelques pages web !

Malgré toutes ces fonctionnalités, ce projet ne conviendra pas à mon problème. Tout le code est écrit en JavaScript et je souhaite utiliser le TypeScript pour son typage très pratique et qui permet de garder un code plus propre et plus facilement maintenable lorsque de nouvelles personnes arrivent sur le projet. De plus, Hackathon-starter est très complet, trop complet même pour mon problème. Je n’ai pas besoin des pages web prédéfinies puisque chaque micro-service contiendra seulement une API Rest. Qui veut dire trop complet dit code inutile et donc suppression de code, ce qui prend énormément de temps. Il faut faire attention à ne pas casser des fonctionnalités que l’on souhaite garder.

## Autres solutions

J’ai pu trouver beaucoup de starter-kit différents sur Github mais quasiment aucun ne respectait les conditions de mon problème. Beaucoup ne sont pas codés en TypeScript, et pour ceux qui le sont, ce sont surtout des starter-kits pour Angular ou React. Le type de base de données utilisé ne me convient pas forcément non plus.

# Exposé des décisions prises et des interventions menées par le stagiaire pour résoudre le problème : [Titre perso]

# Démonstration d’une originalité dans l’élaboration et la mise en œuvre de la solution : [Titre perso]

# Analyse de l’approche choisie : [Titre perso]

# Réflexion sur le stage et le mémoire : [Titre perso]

# Conclusion

1. 18 juillet 2020 : <https://www.oreilly.com/library/view/building-microservices/9781491950340/ch01.html> [↑](#footnote-ref-1)