Table des matières

[1. Introduction 3](#_Toc47290614)

[2. Présentation de l'entreprise : [Titre perso] 3](#_Toc47290615)

[2.1. Amiltone 3](#_Toc47290616)

[2.1.1. Chiffres clés 4](#_Toc47290617)

[2.1.2. Les Factories 4](#_Toc47290618)

[2.2. Mon point de vue 6](#_Toc47290619)

[2.3. Mon alternance 6](#_Toc47290620)

[2.3.1. L’équipe sur le projet 6](#_Toc47290621)

[2.3.2. Utilisation de la méthode Agile 7](#_Toc47290622)

[2.3.3. Mes missions 7](#_Toc47290623)

[3. Analyse du contexte : Le projet AmilApp 9](#_Toc47290624)

[3.1. AmilApp 9](#_Toc47290625)

[3.1.1. Les news 9](#_Toc47290626)

[3.1.2. Les événements 10](#_Toc47290627)

[3.1.3. Les sondages 10](#_Toc47290628)

[3.1.4. Les alertes 10](#_Toc47290629)

[3.1.5. Les améliorations futures 10](#_Toc47290630)

[3.2. Firebase 10](#_Toc47290631)

[3.2.1. Pourquoi l’avoir choisi ? 11](#_Toc47290632)

[3.2.2. Avantages et inconvénients de Firebase 11](#_Toc47290633)

[3.3. Analyse personnelle du contexte de stage de fin d’étude 11](#_Toc47290634)

[4. Problématique : Comment mettre en place une solution web réutilisable, modulaire et évolutive ? 12](#_Toc47290635)

[4.1. Les micro-services 12](#_Toc47290636)

[4.1.1. Définition 12](#_Toc47290637)

[4.1.2. Avantages 13](#_Toc47290638)

[4.1.3. Inconvénients 13](#_Toc47290639)

[4.1.4. Pourquoi les avoir choisis ? 13](#_Toc47290640)

[4.2. Le starter-kit 13](#_Toc47290641)

[5. Méthodes habituellement utilisées pour une situation présentant des similitudes : [Titre perso] 13](#_Toc47290642)

[5.1. Spring Boot 14](#_Toc47290643)

[5.1.1. Fonctionnalités 14](#_Toc47290644)

[5.1.2. Avantages et inconvénients par rapport au projet 14](#_Toc47290645)

[5.2. Hackathon-starter 14](#_Toc47290646)

[5.2.1. Fonctionnalités 15](#_Toc47290647)

[5.2.2. Avantages et inconvénients par rapport au projet 15](#_Toc47290648)

[5.3. Autres solutions 15](#_Toc47290649)

[6. Exposé des décisions prises et des interventions menées par le stagiaire pour résoudre le problème : [Titre perso] 16](#_Toc47290650)

[7. Démonstration d’une originalité dans l’élaboration et la mise en œuvre de la solution : [Titre perso] 16](#_Toc47290651)

[8. Analyse de l’approche choisie : [Titre perso] 16](#_Toc47290652)

[9. Réflexion sur le stage et le mémoire : [Titre perso] 16](#_Toc47290653)

[10. Conclusion 16](#_Toc47290654)

# Introduction

# Présentation de l'entreprise : [Titre perso]

## Amiltone

Amiltone est une société de services numériques (ESN). L’écosystème d’expertise de la Digital Factory, que j’expliquerai un peu plus tard, permet à Amiltone d’accompagner ses clients dans leurs problématiques de développement et de compétitivité grâce à une approche globale des systèmes d’informations. Combinant la valeur ajoutée, l’innovation et la performance des services fournis, Amiltone propose plusieurs solutions : le conseil, l’intégration de systèmes et le développement de solutions métiers adaptées aux besoins spécifiques de chacun de ses clients.

### Chiffres clés

Amiltone a été fondée en 2012 à Lyon, où se trouve actuellement le siège social. L’entreprise comptabilise 270 employés en 2019 et a pour objectif d’atteindre les 300 collaborateurs en 2020. Le chiffre d’affaire de l’entreprise était de 2.7 millions d’euros en 2014 et était de 16 millions d’euros en 2019. Aujourd’hui, Amiltone c’est sept agences en France : Lyon, Paris, Niort, Nantes, Bordeaux, Grenoble et Aix-en-Provence.

### Les Factories

#### La Digital Factory

La Digital Factory est le pôle d’expertise qui accompagne les clients d’Amiltone dans leur transformation digitale. L’objectif de cette entité est d’analyser l’entreprise, le fonctionnement et les outils déjà en place chez le client pour proposer une solution adéquate, stable et évolutive permettant de répondre aux besoins et d’anticiper les évolutions et les changements futurs, des grands groupes comme des petites structures pour que leurs systèmes d’information deviennent l’un des moteurs de leur croissance.

#### La Mobile Factory

La Mobile Factory est l’équipe experte en développement d’applications mobiles natives sur Android et iOS. Amiltone fait le choix de développer en langage natif. En effet, seule une application développée nativement permet d’obtenir une vitesse d’exécution optimale sur n’importe quel terminal. La réactivité est l’atout majeur du natif sur tous les autres types de développement comme le web embarqué ou le cross-platform.

Le natif permet également d’utiliser l’ensemble des éléments d’interface et d’ergonomie propres au système tout en assurant leur évolutivité. L’utilisateur retrouvera, selon son système, une interface et des usages qu’il maitrise et qu’il retrouve dans les applications livrées de base avec son smartphone.

À noter que plusieurs fonctionnalités ne sont pour l’instant réalisables qu’avec du code natif et que maintenir une application hybride avec des parties de code natif augmente considérablement la charge de travail.

La maintenabilité de l’application à moyen et long terme est donc plus sûre et plus simple en utilisant le natif, puisque l’évolution conjointe des Software Development Kit (SDK) et du système permet d’utiliser directement l’ensemble des nouveaux éléments.

Le choix des applications cross-platform ajoute également des risques supplémentaires, comme l’arrêt du support de la technologie utilisée, ou un retard de mise à jour par rapport aux dernières versions des systèmes mobiles, ces derniers évoluant très rapidement.

Partir sur du natif permet d’assurer la longévité de l’application et une maintenance simplifiée.

Amiltone estime que le coût initial d’un projet, bien que plus long à développer en phase de conception, rejoint sur le moyen terme avec la maintenance et les évolutions le coût d’un projet cross-platform.

#### La Web Factory

La Web Factory est la pierre angulaire de l’offre globale d’Amiltone. En effet, l’utilisation de technologies dédiées à la création d’un backend d’administration des données permet de connecter entre toutes les briques d’un système d’information.

Pour chaque nouveau projet, un comité d’experts est chargé d’identifier les technologies Web les plus appropriées. Les réponses aux questions de langages et de base de données permettent de créer une fondation solide basée sur des technologies adaptées et spécifiquement sélectionnées pour leur intérêt sur un sujet spécifique.

En outre, l’expertise de la Web Factory ne se limite pas au développement d’un système qui répond parfaitement à un besoin métier. Son rôle est également de respecter les normes de développement afin d’améliorer le référencement naturel, l’interconnexion simplifiée avec les API ou l’intégration de standards de communication multi-canaux tels que Facebook ou Twitter.

#### La Data Factory

La Data Factory permet de répondre à une problématique commune à de nombreux clients d’Amiltone, quelle que soit leur taille et leur champ d’activité : comment gérer leurs données ? Cette question complexe doit être résolue à plusieurs niveaux. Tout d’abord, le but de la Data Factory est de comprendre les activités du client afin de déterminer avec lui les indicateurs pertinents qui lui permettront de contrôler plus facilement son activité. La deuxième étape consiste à extraire des données pour répondre à ce besoin. Très souvent, les données sont stockées à différents emplacements du Service Informatique (SI) et sur différentes technologies (Base de données, email, Excel par exemple). L’étape suivante consiste à consolider ces données dans un Datawarehouse, avec éventuellement l’ajout de Datamart pour différencier plusieurs secteurs, pour permettre leur utilisation simple et rapide dans la dernière étape qui est l’affichage d’indicateurs pertinents et visuellement percutants pour que les décideurs puissent réagir en un coup d’œil.

La Data Factory offre également des services de surveillances de log serveurs et une surveillance complète d’un flux métier.

Afin d’aller toujours plus loin dans l’analyse des données, la Data Factory est accompagnée par un expert de renommée internationale spécialisé dans l’intégration de briques algorithmiques pour des usages tels que l’analyse prédictive ou le machine learning.

#### Expertise et qualité

La qualité logicielle et l’expertise technique étant les principaux axes de la stratégies d’Amiltone, la Digital Factory a mis en place une veille particulière sur des problématiques telles que :

* La maintenabilité du code
* L’objectif zéro défaut
* Le contrôle du cycle de vie du logiciel
* L’amélioration continue
* L’organisation projet

Les équipes techniques sont impliquées dans ce processus de veille et scrutent en permanence les outils et techniques de demain afin d’améliorer la qualité des projets et la satisfaction des clients.

## Mon point de vue

### Une entreprise impliquée envers ses collaborateurs

Depuis mon arrivée chez Amiltone, j’ai toujours travaillé sur des projets internes. J’ai pu observer les différentes méthodes de travail en équipe et elles n’ont pas toujours été optimales. La priorité étant pour les projets clients, les projets internes étaient délaissés. J’ai eu l’occasion de travailler sur une application sans chef de projet, sans référent technique et tous les collaborateurs qui travaillaient avec moi, ainsi que moi-même, étions débutant sur les technologies utilisées. Nous nous faisions les merge request entre nous et finalement, nous ne pouvions pas vraiment progresser puisque personne ne nous indiquait nos erreurs. Ce fut probablement le dernier projet sans référant technique, sans chef de projet.

Amiltone a su donner de l’importance aux applications internes et améliorer les différents process sur ces projets. Chacun d’eux est maintenant pris en charge par un chef de projet et un référant technique est assigné pour traiter les revues de code.

## Mon alternance

Cela fait maintenant deux années que je suis en alternance chez Amiltone à Lyon sur le projet AmilApp.

### L’équipe sur le projet

Depuis le début de projet, un grand nombre de collaborateurs ont pu travailler sur AmilApp. Nous avons d’abord commencé trois développeurs, tous alternants, chacun sur une plateforme différente : un sur iOS, le système d’exploitation de l’iPhone, un sur Android, celui de Google, et moi-même sur la partie administrateur de l’application, en web, et chacun supervisé par le responsable de chaque factory, pour relire nos merge request.

Une fois la première version d’AmilApp sortie, au bout d’un an, je me suis retrouvé seul sur le projet le temps de développer la partie utilisateur du projet sur le web. Puis, dès le mois de Novembre, l’idée de développer notre propre backend pour remplacer Firebase florissait et je commençais alors à développer l’API avec un autre collaborateur. L’équipe s’agrandit assez rapidement puisque beaucoup de nouveaux collaborateurs arrive chez Amiltone, nous sommes sept sur le projet au mois de Juillet 2020 : deux développeurs sur le backend, deux autres sur les fronts utilisateur et administrateur, deux sur l’application Android et un dernier sur l’application iOS.

Globalement, la taille de l’équipe varie beaucoup en fonction de la charge de travail à effectuer sur le projet et des missions clientes. En effet, tous les développeurs peuvent être envoyé en mission à n’importe quel moment, l’équipe d’AmilApp n’est jamais fixe.

### Utilisation de la méthode Agile

Malgré le fait qu’AmilApp soit un projet interne et qu’il n’y a pas réellement de client final, l’équipe essaye au maximum de fonctionner en mode agile. Les « sprint » sont renouvelés toutes les deux semaines afin de permettre une meilleure réactivité aux possibles évolutions et améliorations. Comme AmilApp est un projet interne, il n’y a à priori pas de deadline mais plutôt des objectifs, comme par exemple, faire en sorte que l’application soit présentable à certains salons auxquels Amiltone a participé.

Plusieurs fois dans l’histoire du projet, de nouvelles évolutions conséquentes ont été demandées, comme de nouvelles pages sur le site web ou de nouveaux écrans sur les applications mobiles. À chacune d’elle, l’équipe organisait plusieurs réunions avec le chef de projet, les collaborateurs qui se sont occupés des maquettes et les développeurs, qui nous ont permis de discuter des nouvelles maquettes, de ce qui était faisable ou non, de l’ergonomie de celles-ci, ou même parfois des fonctionnalités qui peuvent manquer de cohérence de temps en temps.

D’autres réunion sont ensuite organisées pour que les développeurs puissent estimer les nouvelles tâches. Etant sur le projet depuis le début, c’est moi qui suis en charge de créer les différents tickets du projet web sur le tableau Jira.

De temps en temps, trop peu à mon avis, des revues de sprint sont organisées avec le « product owner », qui n’est autre que mon maître de stage Alexandre Buffy, pour lui présenter les avancées du projet. Mais effectivement, ces réunions sont surtout organisées lorsqu’il y a de grosses avancées sur le projet.

Le chef de projet permet ici d’harmoniser la communication entre les différentes équipes de développement ainsi que l’équipe en charge des maquettes et le « product owner ».

### Mes missions

Comme dit précédemment, ma mission sur le projet AmilApp a été dans un premier temps de développer l’application web. C’est dans cette année de développement que j’ai pu réellement progresser, grâce au développeur leader de la Web Factory qui a fait mes revues de code. Malgré le côté frustrant que ces revues de code peuvent avoir lorsqu’il y a beaucoup de choses qui ne vont pas techniquement dans le code, je me suis vite rendu compte que c’est grâce à celles-ci j’ai pu m’améliorer. Rapidement, après le début du projet, il n’y avait déjà plus beaucoup de code à corriger pendant les revues.

Une fois la première version d’AmilApp déployée, j’ai commencé le développement du front utilisateur web. J’étais alors seul sur le projet et j’ai pu prendre la décision de repartir de zéro afin de mettre à jour les différentes versions des paquets utilisés sur le projet. Effectivement, j’ai trouvé plus simple de prendre cette option plutôt que de continuer sur l’existant, sachant que nous étions partis sur l’ancien starter-kit maison en Angular. Il m’a permis de rapidement commencer le projet la première année mais je me suis vite rendu compte que beaucoup de code n’était pas utilisé et pour moi, c’était perdre du temps de nettoyer un projet qui, de toute façon, n’était pas à jour avec Angular.

J’ai alors pu expérimenter certaines choses sur le projet pour essayer de le rendre le plus propre et le plus générique possible, c’est-à-dire que plusieurs fonctionnalités utilisent le même code, les mêmes fonctions, les mêmes classes, indépendamment du type de donnée. Après plus d’un an passé, ce code est toujours utilisé et une fois maitrisé par les nouveaux collaborateurs, permet d’ajouter de nouvelles pages web similaires aux précédentes très rapidement et avec un minimum de code.

J’ai ensuite pris la décision de reprendre le code de la partie administrateur pour l’intégrer complètement à la partie front utilisateur afin d’être cohérent sur les projets et surtout ne pas avoir à déployer deux projet web différents pour une même application. L’intégration s’est faite rapidement grâce au code générique dont j’ai parlé plus haut.

Le front utilisateur maintenant déployé, grâce à mon expérience acquise sur le projet, j’ai pu exprimer mon ressenti par rapport à Firebase ainsi que mon envie de nous en séparer à mon chef de projet. La décision est alors prise de nous détacher complétement de Firebase et de créer notre propre API en micro-services. Grâce à celle-ci, il sera éventuellement possible de déployer l’application pour d’autres client en mode édition logicielle Saas (Software as a service, Logiciel en tant que service).

Etant la personne la plus ancienne sur le projet et celle ayant le plus d’expérience, j’ai maintenant la responsabilité de relire et de valider, ou non, le code Angular de mes collègues sur le projet AmilApp. En général, chaque personne de l’équipe peut choisir le ticket, la fonctionnalité sur laquelle il veut travailler, mais régulièrement, mes collègues viennent me demander ce qu’ils peuvent commencer lorsque les tâches sont plus ardues. Dans ce cas-là, je n’hésite pas à régulièrement demander si tout se passe bien et fourni de l’aide si besoin tout en essayant de faire en sorte qu’ils comprennent le code du mieux possible.

# 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.

## Analyse personnelle du contexte de stage de fin d’étude

# 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

Les avantages de l’architecture micro-services sont nombreux :

* Comme chaque service ne représente qu’une petite partie du code d’un projet, il y a naturellement moins de code que dans un gros projet avec une architecture monolithique. Moins il y a de code, plus les chances que des bugs surviennent sont faibles. De plus, les tests unitaires seront exécutés beaucoup plus rapidement.
* Optimisé pour la mise à l’échelle horizontale, c’est-à-dire que lorsque les capacités de la machine qui héberge l’application sont trop limitées pour supporter le nombre de requêtes envoyées par les utilisateurs, l’application est facilement déployable sur plusieurs serveurs distincts de façon à réduire la charge de travail entre ceux-ci. Et contrairement à la mise à l’échelle verticale, l’application reste toujours disponible pendant l’ajout de ressources supplémentaires et si un des serveurs tombe en panne, l’application reste disponible.
* Offre une meilleure résilience,
* Est plus adapté au fonctionnement d’équipe en mode agile. En effet, dans mon cas, comme une seule équipe travaille sur tous les services, je peux par exemple me permettre de livrer uniquement un seul service fonctionnel en fin de sprint, ce qui peut être bloquant dans un projet monolithique où certains services peuvent être dépendants des autres.

### 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)