UQAC
Université du Québec
à Chicoutimi

UNE PROPOSITION DE MODÈLE DÉCISIONNEL MULTICRITÈRE POUR LA SÉLECTION DU FRAMEWORK NODE.JS

PAR MÁRIO DANILO ALVES DA SILVA

MÉMOIRE PRÉSENTÉ À L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI EN VUE DE L'OBTENTION DU GRADE DE MAÎTRE ÈS SCIENCES (M.Sc.) EN INFORMATIQUE

QUÉBEC, CANADA

RÉSUMÉ

Au fur et à mesure que Node.js évolue et mûrit, de plus en plus de frameworks sont créés pour cette technologie. Cependant, avec une telle gamme de frameworks disponibles, le choix du framework le plus approprié en fonction de sa fonctionnalité ou de ses caractéristiques peut entraîner des complications. La prise de décision des développeurs devient un problème. Ce mémoire fournit des orientations et des informations de prise de décision pour les développeurs. Un modèle multicritère de sélection des frameworks Node.js est présenté et une méthodologie d'application de ce modèle est décrite. Cette méthodologie présente une application d'une technique d'aide à la décision multicritère et est basée sur la méthode multicritère AHP. Enfin, une étude de cas est présentée pour démontrer l'efficacité du modèle.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS

AHP Processus d'analyse hiérarchique

API Interface de programmation d'applications

CLI Interface de ligne de commande

CMS Système de gestion de contenu

CRUD Créer, Lire, Mettre à jour et Supprimer

DEX Shell expert décisionnel

DTO Objet de transfert de données

ELECTRE ELimination Et Choix Traduisant la REalité

ETL Extraction, transformation, chargement

EW-LMS Moyen facile d'évaluer les LMS

HAW Pondération adaptative hiérarchique

HTML HyperText Markup Language

HTTP Protocole de transfert hypertexte

JSON JavaScript Object Notation

LMS Système de gestion de l'apprentissage

MCDM Prise de décision multicritères

MODM Prise de décision à objectifs multiples

MADM Prise de décision à attributs multiples

MVC Modèle-vue-contrôleur

PROMETHEE Méthode d'organisation du classement des préférences pour l'évaluation de l'enrichissement

REST Transfert d'état représentatif

SAW Pondération additive simple

TFN Nombre flou triangulaire

TODIM Prise de décision multicritère interactive

TOPSIS Technique d'Ordre de Préférence par Similitude à la Solution Idéale

URL Uniform Resource Locator

WAS Somme moyenne pondérée

WPM La méthode du produit pondéré

WSM La méthode de la somme pondérée

XML Langage de balisage extensible

DÉDICACE

A mes parents et mon épouse,

J'ai la chance d'appartenir à une famille unie. Merci d'être là pour moi et de m'encourager dans mes projets et mes rêves. Merci de m'épauler dans les moments les plus difficiles et de me donner la motivation nécessaire pour suivre ma maîtrise. Sachez que j'en suis très reconnaissant.

REMERCIEMENTS

Je tiens à remercier mon superviseur Fabio Petrillo, professeur au Département d'informatique et de mathématiques de l'Université du Québec à Chicoutimi, pour l'opportunité, la compréhension et le dévouement qu'il a eu avec moi.

Je tiens à remercier mes parents, Mariomar et Cláudia, pour leur soutien à chaque instant de ma vie. J'aimerais également remercier mon épouse Ana Flávia qui a été partenaire lors de ce travail.

Je tiens aussi à remercier les collègues du laboratoire SmArtSE qui m'ont aidé directement et indirectement avec beaucoup de doutes. Je remercie également l'UQAC pour cette opportunité.

Enfin, je voudrais remercier à Dieu de m'avoir mis sur un chemin heureux.

CHAPITRE I

INTRODUCTION

1.1 CONTEXTE

L'utilisation croissante de la technologie a apporté une immense innovation dans les applications utilisées sur les appareils mobiles et sur les sites Web. Cependant, les besoins du marché en logiciels ont beaucoup changé au fil des ans. Par conséquent, les entreprises ont dû évoluer pour répondre à la demande afin de fournir des résultats agiles et précieux à leurs clients. Pour répondre à cette demande, les entreprises investissent dans le développement Web. Il s'agit actuellement d'une activité de 14 000 milliards de dollars américains et devrait croître en moyenne de 5% par an [1]. En général, le développement Web est divisé en deux blocs : le front-end [1] et le back-end [2]. Le premier bloc est responsable des fonctionnalités côté client, qui interagissent directement sur le Web, comme le design qui doit être transmis par la page Web, les effets visuels, les animations et autres [2]. L'autre bloc est le côté serveur, responsable de tout ce que nous interagissons indirectement sur le Web, comme les bases de données, Interface de programmation d'applications (API) pour la communication avec le front-end, la logique de sécurité et les règles métier [3].

Le développement back-end et front-end utilisent tous deux largement les frameworks [3] [4]. Un framework est un outil optionnel dans le développement d'applications, cependant,

---

1. En français, nous pouvons utiliser le mot *frontal* pour *front-end*. Par contre, nous avons les garder le mot *front-end* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

2. En français, nous pouvons utiliser le mot *dorsal* pour *back-end*. Par contre, nous avons les garder le mot *back-end* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

3. En français, nous pouvons utiliser le mot *cadriciel* pour *framework*. Par contre, nous avons les garder le mot *framework* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

c'est un outil disponible pour aider à développer et améliorer la vitesse de développement [5]. Les frameworks permettent de gagner du temps dans le développement d'applications, car ils fournissent des fonctionnalités prêtes à l'emploi, augmentent la productivité des développeurs de logiciels et facilitent l'apprentissage [6]. Ainsi, les frameworks sont utilisés par diverses technologies et langages de programmation [2].

Pour le développement back-end, Node.js est une technologie largement répandue et utilisée ces dernières années parmi les développeurs, selon une étude de Stack Overflow (2021) Node.js est la sixième technologie le plus populaire parmi les développeurs, de même qu'est également utilisée par la moitié des développeurs de logiciels interrogés, en tête du classement des technologies avec 51,4% [8]. Depuis sa sortie en 2009, Node.js joue un rôle important dans l'utilisation du langage JavaScript pour le développement d'applications back-end [9]. En raison de cette importance et de ses avantages, pour la technologie Node.js, plusieurs frameworks sont disponibles pour le développement back-end [10, 11], et cette grande diversité rend difficile la prise de décision des développeurs pour décider lequel utiliser [12].

1.2 PROBLÈME ET MOTIVATIONS

Dans la conception d'une nouvelle application, le choix d'un framework est une action complexe [2]. Actuellement, avec l'existence de plusieurs frameworks de développement back-end disponibles [10, 11], choisir le framework le plus approprié en fonction de ses fonctionnalités [13] ou de ses caractéristiques peut entraîner des complications [14, 15]. En bref, il est déroutant d'absorber toutes ces différences, de sorte que la décision de sélectionner le framework qui convient le mieux à application est un problème.

De cette façon, l'affirmation de soi dans l'incorporation d'un framework qui répond le mieux aux exigences du projet peut faire la différence et contribuer au succès du projet. Considérant les équipes de développeurs qui ont des connaissances différentes, ainsi que les différents détails des fonctionnalités mises en œuvre par les frameworks, ainsi que les différences entre les caractéristiques des frameworks, nous proposons un modèle de prise de décision.

1.3 OBJECTIF

Notre travail consiste à proposer un modèle décisionnel multicritère pour la sélection du framework Node.js, afin d'appuyer le choix du framework le plus adapté à l'application. Pour atteindre cet objectif, nous faisons une analyse de la littérature multivocale qui inclut la littérature grise [16] sur les frameworks de développement back-end pour la technologie Node.js. L'objectif est de recueillir les critères de sélection des frameworks utilisés dans la communauté et dans l'industrie.

Plus précisément, nous faisons une revue systématique des articles, blogs et sites Web pertinents de professionnels qui partagent leurs idées et leur expérience professionnelle de la pratique des logiciels [17]. Premièrement, nous collectons les critères de sélection appliqués lors du choix d'un framework. En parallèle, nous identifions les frameworks Node.js les plus cités. Ensuite, nous discutons des critères utilisés, ainsi que de l'analyse et de l'interprétation des résultats obtenus.

Ensuite, nous proposons notre modèle décisionnel multicritère pour la sélection du framework Node.js en utilisant une méthode d'aide à la décision multicritère, dans le but d'aider et de guider le décideur dans la structuration, l'évaluation et le choix d'alternatives

pour le problème en question [18]. Enfin, nous faisons une application du modèle multicritère et la conclusion.

1.4 CONTRIBUTIONS

Dans cet objectif, ce travail met en évidence les termes de contributions suivantes. Premièrement, nous proposons un modèle décisionnel multicritère pour la sélection du framework Node.js afin d'aider à la sélection d'un framework parmi plusieurs disponibles. Notre modèle définit des critères et une méthode mathématique multicritère pour l'aide à la décision, présente une méthodologie et fournit une structure de décision pour sélectionner le framework qui convient le mieux au besoin et au contexte de l'application. Notre approche complète également une cartographie des principaux frameworks Node.js disponibles. Nous mettons en évidence les 30 frameworks les plus recommandés pour la technologie Node.js. Nos contributions sont listées ci-dessous :

- Une cartographie des frameworks Node.js clés.
- Une proposition de modèle décisionnel multicritère pour la sélection du framework Node.js.

1.5 ORGANISATION

Ce mémoire est organisé comme suit. L'objectif de ce chapitre est de présenter le projet de recherche sur un modèle décisionnel multicritère pour la sélection du framework Node.js. Le deuxième chapitre est composé de l'état de l'art et clarifie en détail les idées initiales pertinentes. Le troisième chapitre est composé du point de vue de la revue de la littérature multivocale. Les principaux critères de sélection des frameworks y sont décrits, ainsi que leurs principaux contextes d'application. Nous présentons également les frameworks

les plus cités. L'objectif du troisième chapitre est de présenter notre proposition de modèle décisionnel multicritère pour la sélection du framework Node.js. De même que les critères de sélection ainsi que la description de la méthode mathématique et la méthodologie de choix d'un framework de développement back-end. Le quatrième chapitre décrit une étude de cas pour l'application du modèle. Enfin, nous présentons la conclusion générale de ce mémoire.

CHAPITRE II

ÉTAT DE L'ART

Dans ce chapitre nous abordons l'état de l'art, afin de clarifier les concepts fondamentaux et pertinents de notre travail. Le chapitre est divisé en quatre parties, la première apporte une brève introduction au concept de frameworks Web, qui est l'objet de prédilection de notre modèle. La deuxième partie traite de la technologie de développement back-end Node.js. La troisième partie aborde la définition des méthodes d'aide à la décision multicritères. Enfin, la quatrième partie présente la synthèse des travaux connexes.

2.1 FRAMEWORKS WEB

Un framework est un ensemble d'outils qui fournissent des fonctionnalités communes pour un type particulier d'application [19] et résume certains des aspects les plus complexes du développement [11], ainsi qu'une solution de haut niveau pour la réutilisation des logiciels [20]. De cette façon, il garantit un niveau de qualité supérieur du produit final, puisqu'une partie importante du code de l'application, qui se trouve dans le framework, est déjà testée [21].

Un framework fournit des outils pour interagir avec l'application, y compris le routage des URL (Uniform Resource Locator) pour gérer les requêtes HTTP (Protocole de transfert hypertexte) [19], l'interaction avec les bases de données [2], les sessions d'appui et l'autorisation des utilisateurs, améliorer la sécurité contre les attaques Web [15] et formatage de la sortie [20], par exemple HyperText Markup Language (HTML), JavaScript Object Notation (JSON), Langage de balisage extensible (XML). Les frameworks permettent aux développeurs de se mettre rapidement au travail sur les résultats commerciaux, de gagner du temps de développe-

ment, d'augmenter la productivité des développeurs de logiciels et de faciliter l'apprentissage [6].

Il existe plusieurs types de frameworks Web et chacun offre une expérience de programmation différente [15]. Parmi ces types figurent le Modèle-vue-contrôleur (MVC) , le Full-Stack [4] [10, 11], le Transfert d'état représentatif (REST) API [10, 11], le micro-cadriciel [10]. Un framework est un outil facultatif dans le développement d'applications, cependant, c'est un outil disponible pour aider à développer et à améliorer la vitesse de développement [5].

2.2 NODE.JS

Node.js est une technologie qui a été largement utilisée pour le développement back-end, selon la recherche Stack Overflow (2020), Node.js est en tête du classement technologique avec 51,4 %, car il est utilisé par la moitié des développeurs de logiciels répondants, de même que la sixième technologie la plus populaire parmi les développeurs [7].

Depuis sa sortie en 2009, Node.js a joué un rôle important dans le développement d'applications back-end utilisant le langage JavaScript [9]. Node.js est une technologie de développement Web à code source ouvert ainsi qu'un environnement d'exécution à thread unique [22]. Node.js est construit sur le moteur JavaScript V8 de Chrome. V8 est un moteur haute performance et à code source ouvert de Google écrit en C++, qui implémente ECMAScript [5]

4. En français, nous pouvons utiliser le mot *pile complète* pour *full-stack*. Par contre, nous avons les garder le mot *full-stack* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

5. ECMAScript est un langage de programmation basé sur des scripts normalisé par Ecma International dans les spécifications ECMA-262 et ISO/IEC 16262, disponible sur https://tc39.es/ecma262.

et WebAssembly [6], qui compile et exécute le code source JavaScript, qui gère l'allocation de mémoire pour les objets et qui récupère les objets qui ne sont plus nécessaires [23]. Le moteur V8 est adapté au processus de développement d'applications Web Node.js [22].

Node.js utilise un modèle d'architecture d'entrée et de sortie (E/S) non bloquant [3] et utilise une programmation asynchrone pilotée par les événements qui est conçue pour les services réseau [24], ce qui le rend léger et efficace. L'avantage important de la programmation événementielle et asynchrone est qu'elle utilise pleinement les ressources système [24]. L'idée de thread unique est basée sur une boucle d'événements qui est responsable de l'exécution du code utilisateur, y compris les rappels de fonction pour répondre aux événements de manière asynchrone [25], ce qui signifie que bien que ces fonctions semblent être enregistrées, selon un ordre séquentiel dans la structure du code, elles ne dépendent pas de l'ordre dans lequel ils apparaissent, mais attendez que l'événement correspondant soit déclenché.

Selon les auteurs [26, 27], Node.js est utilisé pour créer des systèmes back-end évolutifs. L'une des caractéristiques importantes de Node.js est qu'il s'agit d'un framework multiplate-forme [22], parce qu'il est développé avec la bonne structure, qui peut être empaqueté dans un exécutable contenant toutes les dépendances.

Node.js utilise l'un des plus grands écosystèmes de bibliothèque JavaScript au monde, NPM [7] avec sa structure simple aide l'écosystème Node.js à proliférer [22]. Node.js est populaire auprès de plusieurs entreprises internationales renommées, selon StackShare [28], plus de 8 062 entreprises utilisent Node.js, y compris des géants tels que Netflix, Uber, Twitter, Glovo, PayPal et Asana.

---

6. WebAssembly est un format d'instruction binaire pour une machine virtuelle basée sur une pile, une compilation portable pour les langages de programmation, permettant le déploiement Web pour les applications client et serveur, disponible sur https://webassembly.org.

7. NPM est un gestionnaire de paquets à code source ouvert et une collection publique de paquets pour Node.js, disponible sur https://www.npmjs.com/.

2.3 MÉTHODES DE PRISE DE DÉCISION MULTICRITÈRES

Une décision est prise sur la base de jugements intuitifs la plupart du temps [29], où divers éléments intégraux du problème sont traités dans une analyse non systématique. La prise de décision est une activité intrinsèquement complexe et potentiellement l'une des plus controversées [30], car nous devons choisir non seulement entre des alternatives possibles, mais aussi entre des points de vue. Dans la plupart des cas, les décisions sont favorables et satisfaisantes lorsque les décisions sont prises sur la base de plusieurs critères [31].

Cela a conduit à l'émergence des méthodes de Prise de décision multicritères (MCDM), qui sont considérées comme des outils mathématiques, efficaces pour résoudre des problèmes dans lesquels il existe des critères contradictoires [32], en raison de la complexité du processus de prise de décision entre l'information et la nécessité d'atteindre une plus grande assurance dans la prise de décision [33].

Il existe deux groupes principaux de modèles MCDM, le premier est Prise de décision à objectifs multiples (MODM) qui est utilisé pour la conception, le second est Prise de décision à attributs multiples (MADM) qui est utilisé pour sélectionner la meilleure option. Selon Zeleny [34], en général, le premier modèle est défini dans un espace continu et le deuxième modèle est défini dans un espace décisionnel discontinu. D'après Triantaphyllou [35], ces modèles ne tentent pas de calculer une solution optimale, mais ils tentent plutôt de déterminer par classement un classement des stocks pertinents qui est « optimal » par rapport à divers critères.

En théorie, de nombreuses méthodes ont été développées et sont à l'étude pour résoudre des problèmes dans de nombreuses variantes [36]. Pour résoudre des problèmes multicritères, il existe plusieurs méthodes telles que le Processus d'analyse hiérarchique (AHP), le Méthode d'organisation du classement des préférences pour l'évaluation de l'enrichisse-

ment (PROMETHEE), ELimination Et Choix Traduisant la REalité (ELECTRE), La méthode du produit pondéré (WPM), Technique d'Ordre de Préférence par Similitude à la Solution Idéale (TOPSIS) et Prise de décision multicritère interactive (TODIM).

2.4 TRAVAUX CONNEXES

Sélection du logiciel ETL. Pour le processus d'extraction, de transformation et de chargement (ETL) de données, chaque logiciel utilise ses propres méthodes, l'évaluation de ces logiciels est donc problématique. La sélection du bon logiciel ETL est très importante pour le succès ou l'échec des projets d'intelligence d'affaires. Pour le problème de prise de décision multicritère, Hanine et al. [37] utilisent les méthodes AHP et TOPSIS qui permettent de sélectionner facilement le bon logiciel ETL et pour les évaluations de ses substituts. Le but de l'utilisation d'AHP est d'analyser la structure du problème de sélection du logiciel ETL et d'obtenir des poids des critères sélectionnés. La technique TOPSIS est utilisée pour calculer les classements des alternatives.

Sélection du fournisseur d'informatique en nuage. Muthuraman et al. [38] utilisent la méthode Fuzzy AHP pour sélectionner les services d'informatique en nuage. Ces dernières années, il y a eu une augmentation du nombre de services disponibles, les fournisseurs de nuage ont de vastes concepts, de sorte qu'un cadre d'évaluation aide à estimer la compétence d'un fournisseur de ressources dans l'accomplissement d'une tâche, permettant aux utilisateurs de sélectionner les meilleures ressources dans l'infrastructure de nuage. Les comparaisons suivent la version Fuzzy AHP et utilisent également le Nombre flou triangulaire (TFN) pour donner des valeurs à chaque couple de paramètres. Cette comparaison est faite par le chef de projet sur une échelle de 1 à 5. Les estimations de confiance obtenues montrent un écart pour les paramètres qui ne sont pas directement proportionnels aux attributs contributifs.

Sélection de logiciels de dossier médical électronique à code source ouvert. Les logiciels de dossiers médicaux offrent un grand nombre de fonctionnalités personnalisables, une sélection inappropriée de logiciels qui ne répondent pas aux besoins de l'organisation peut entraîner des décisions stratégiques incorrectes et par conséquent une perte économique. Zaidan et al. [36] utilisent les techniques MCDM pour sélectionner un logiciel de dossier médical électronique à code source ouvert. Une étude pratique est réalisée pour évaluer le logiciel et la méthode AHP est utilisée pour la sélection. La méthode AHP est intégrée à différentes techniques MCDM, notamment WPM, La méthode de la somme pondérée (WSM), Pondération additive simple (SAW), Pondération adaptative hiérarchique (HAW) et TOPSIS.

Sélection d'un système de gestion de l'apprentissage à code source ouvert. Abdullateef et al. [39] mènent une étude pour explorer et enquêter sur le problème de la sélection inappropriée des Système de gestion de l'apprentissage (LMS). Ces logiciels sont utilisés pour organiser, mettre en œuvre et soutenir l'éducation, fournir du matériel d'apprentissage en ligne et créer un environnement d'apprentissage collaboratif. L'étude est réalisée en trois étapes, dans la première étape, ils vérifient les logiciels LMS à code source ouvert, 23 logiciels LMS ont trouvés. Ensuite, ils sélectionnent 11 articles pour obtenir les critères d'évaluation et les décrivent. Enfin, ils analysent les techniques MCDM appliquées à la sélection de logiciels LMS. Ces techniques et outils incluent le système Shell expert décisionnel (DEX), Moyen facile d'évaluer les LMS (EW-LMS) et AHP.

Sélection de logiciels d'exploration de données et d'aide à la décision. Collier et al. [40] fournit des informations d'orientation et de prise de décision pour le professionnel dans la sélection de logiciels d'exploration de données. Ils présentent un framework d'évaluation des outils d'exploration de données et décrivent une méthodologie comprenant les phases suivantes : présélection de l'outil, identification des critères de sélection supplémentaires, pondération des critères de sélection, notation de l'outil, évaluation de la notation et sélection

de l'outil. La méthodologie d'application de ce framework est basé sur la méthode de la matrice de décision, c'est une technique qualitative utilisée pour classer les options multidimensionnelles d'un ensemble d'options. Enfin, ils présentent une étude de cas pour démontrer l'efficacité de la méthode.

Sélection de paquets logiciels. Jadhav et Sonar [41] effectuent une revue systématique de 64 articles publiés dans des revues et des actes de conférence qui traitent de la façon d'évaluer et de sélectionner des progiciels qui répondent aux exigences d'une entreprise. Cette revue littéraire étudie les méthodologies de sélection des progiciels, les techniques d'évaluation des logiciels, les critères d'évaluation des logiciels et les systèmes qui aident les décideurs d'évaluer les progiciels. Différentes techniques d'évaluation de progiciels sont proposées dans la littérature, des techniques d'aide à la décision multicritère sont présentes. Les techniques d'évaluation discutées dans les articles examinés sont AHP, analyse des ressources, Somme moyenne pondérée (WAS) et Fuzzy. Ils concluent que l'AHP est largement utilisé pour évaluer les progiciels, qu'il n'y a pas de liste commune de critères génériques pour l'évaluation des logiciels et leur signification et enfin, qu'il est nécessaire de développer un framework qui inclut la méthodologie de sélection des logiciels, la technique d'évaluation, critères d'évaluation et système pour aider les décideurs à choisir un logiciel.

CHAPITRE III
PERSPECTIVE LITTÉRAIRE

Pour la perspective littéraire, nous définissons nos questions de recherche, puis nous faisons une revue de la littérature grise pour trouver les critères utilisés pour la sélection des frameworks, en même temps identifier les frameworks les plus cités.

Ce chapitre est divisé en quatre parties, qui consistent à décrire les motivations de la recherche, à décrire la méthodologie, à montrer les résultats des questions de recherche et à faire les dernières considérations sur les critères de sélection trouvés dans la perspective littéraire.

3.1 MOTIVATIONS

Il existe plusieurs frameworks pour presque tous les langages de programmation [2], de même que plusieurs frameworks disponibles pour le développement back-end utilisant la technologie Node.js [10, 11]. Alors que lorsqu'une technologie est aussi populaire que Node.js, elle a généralement une communauté active de développeurs [42], de nouveaux frameworks émergent constamment et cette liste semble s'allonger de plus en plus [14].

Bien que les frameworks fassent partie de la panoplie d'outils de la plupart des développeurs, leurs caractéristiques peuvent apporter des complications [14, 15] lorsqu'il faut choisir leurs avantages réels pour le projet.

Nous nous concentrons sur les critères de sélection des frameworks et les frameworks clés, identifiant ce qui rend ces frameworks populaires, avec l'intention d'aider à guider la

décision d'en choisir un. Nous explorons la perspective de la littérature pour mieux comprendre les caractéristiques, les qualités et les capacités des frameworks et leur popularité.

3.2 MÉTHODOLOGIE

Dans cette section, nous décrivons notre méthodologie choisie. Pour effectuer notre revue de la littérature, nous avons mené nos recherches en utilisant des sources de littérature grise. L'inclusion de la littérature grise peut apporter des avantages substantiels dans certains domaines du génie logiciel [43] et pose certains défis [44], car les preuves qu'elles contiennent sont souvent basées sur l'expérience et l'opinion. La littérature grise en génie logiciel peut être définie comme tout matériel qui n'est ni officiellement évalué par des pairs ni officiellement publié [17]. Dans notre travail, nous utilisons des articles écrits dans des blogs technologiques et des manuels des frameworks comme artefacts. La méthodologie que nous utilisons pour notre revue systématique de la littérature comprend : le processus de recherche et de sélection d'articles sur les frameworks Node.js, l'extraction et la synthèse des données.

Nous basons notre recherche sur deux questions de recherche pour atteindre notre objectif. Ces questions sont :

- QR1 : Quels sont les critères de sélection d'un framework ?
- QR2 : Quels sont les framewoks Node.js plus cités ?

En répondant à la question QR1, nous pouvons avoir une meilleure compréhension des critères et également des informations essentielles pour construire notre modèle multicritère. En répondant à la question QR2, nous identifions quels sont les frameworks les plus cités ou recommandés pour Node.js, ce qui nous permet d'analyser les caractéristiques qui les rendent particuliers.

Notre processus de recherche et de sélection se déroule en plusieurs étapes. Premièrement, nous effectuons une recherche avancée avec le mot-clé suivant *« nodejs framework best OR top OR web OR comparing OR select OR choose OR choosing OR backend OR back-end -frontend -front-end »* dans le moteur de recherche Google. Cette requête nous permet d'obtenir les résultats de plusieurs articles qui abordent le sujet choix du framework Node.js. Sur la base de ces résultats, nous sélectionnons les articles les plus pertinents selon nos critères de sélection.

Pour la recherche des critères de sélection des frameworks et des principaux frameworks Node.js, nous appliquons des critères de sélection d'exclusion et d'inclusion pour soustraire les articles non pertinents et incorporer les articles essentiels. Nous énumérons ci-dessous les critères d'exclusion et d'inclusion utilisés.

1. Inclure : Tout sur les études comparatives entre les frameworks Node.js et l'analyse critique à leur sujet.
2. Exclure : Toutes les études qui ne sont pas liées au framework Node.js.
3. Exclure : Toutes les études liées au framework front-end.

En raison de la nature des sources de données impliquées, nous créons un ensemble de données avec des copies d'articles sélectionnés pour la conservation des références bibliographiques. L'ensemble de données, les scripts et tout le matériel de cette étude se trouvent dans notre dépôt de réplication [8].

Pour l'extraction des données, nous créons une structure de classification et collectons les données de chaque article. Ensuite, nous compilons dix-neuf articles de blog que nous trouvons parmi les résultats de recherche. Les articles trouvés se situent entre les années 2017

8. Disponible sur : https://github.com/Mario-UQAC/dataset

et 2021. Le processus utilisé pour considérer le choix d'un framework consiste à identifier les critères et les paramètres de sélection des frameworks parmi les principaux recommandés.

L'activité de synthèse de données a pour objectif principal de comprendre, d'analyser et de classer les articles actuels [45]. Plus précisément, nous effectuons une combinaison d'analyses de contenu et interprétons les résultats de l'analyse de contenu.

3.3 RÉSULTATS

La section des résultats de la question de recherche est présentée en deux parties. Ce sont les principaux frameworks Node.js et les critères de sélection des frameworks.

3.3.1 QR1 : QUELS SONT LES CRITÈRES DE SÉLECTION D'UN FRAMEWORK ?

Les critères présentés ici sont le résultat de notre première question de recherche qui nous aide à avoir des informations et les méthodes utilisées pour sélectionner les frameworks Node.js proposés dans les articles de recherche. Les critères suivants servent de base à la proposition du modèle multicritère de sélection des frameworks Node.js.

Parmi les paramètres abordés pour le critère « communauté », on peut souligner la « popularité », qui est un paramètre de préférence chez les développeurs [46, 19, 47, 48, 11, 49] ainsi que, les étoiles du dépôt GitHub [9] [50, 51, 52], les dépendants sur GitHub [51], les suiveurs sur Stack Overflow [10] [46, 50] et téléchargements hebdomadaires [50, 51, 52] d'un framework sur la page NPM Trends [11]. Avec une grande communauté, la facilité d'obtenir des

9. GitHub est une plateforme d'hébergement de code source et de fichiers avec contrôle de version à l'aide de Git, disponible sur https://github.com/.

10. Stack Overflow est un site de questions et de réponses destiné aux professionnels et passionnés dans le domaine de la programmation informatique, disponible sur https://stackoverflow.com/.

11. NPM Trends est une page officielle qui compare le nombre de téléchargements de paquet au fil du temps, disponible sur https://www.npmtrends.com/.

conseils augmente [46, 50, 48] et il est plus facile de trouver des bibliothèques écrites pour fonctionner avec le framework [50, 53, 54]. Tous ces paramètres de « communauté » discutés sont destinés à aider les développeurs à sélectionner un framework plus populaire.

Le critère que nous identifions comme « le but », apparaît dans 14 articles. Autrement dit, certains frameworks sont créés dans le but de résoudre certains problèmes [15]. Réfléchissez à l'objectif du projet [46, 50, 55, 56, 14], essayez de clarifier et de déterminer les exigences du projet [53, 56, 11, 49]. Certains des paramètres que nous pouvons considérer sont : quel type d'architecture [50, 57], comme API, REST, MVC, microservices, sans serveur et temps réel (WebSockets), nous devons considérer si nous allons écrire l'application avec le langage TypeScript [50], qui est un langage de programmation qui permet la saisie statique [58]. Ces paramètres montrent comment les frameworks sont utilisés entre les développeurs.

Le critère « style » fait référence à la flexibilité de personnalisation d'un framework [46, 55]. Fait référence à un framework qui offre des fonctionnalités recommandées [59, 53, 15], les rendant ainsi « opiniâtres » ou « non opiniâtres » par rapport au code [50, 15]. Ils peuvent également structurer le code et les fichiers de manière spécifique afin que nous puissions profiter des fonctionnalités proposées [50, 10, 11]. De plus, nous devons choisir ou décider d'intégrer des paquets tiers pour les fonctionnalités nécessaires [10, 54]. De cette façon, le framework peut inclure des outils comme une Interface de ligne de commande (CLI) [12], la validation, la journalisation, l'authentification, les abstractions de base de données et l'injection de dépendances [50, 15].

La « performance » est un critère cité par 8 articles. En somme, ils citent le chargement des ressources système [55, 56, 52], si le framework peut répondre aux besoins d'efficacité d'affaires [19, 11] et de rapidité [15].

12. Une CLI traite les commandes d'un programme informatique sous la forme de lignes de texte.

L'effort d'apprentissage d'un framework [47, 15], la facilité d'utilisation [11], les capacités d'installation [52], la simplicité de construction [55, 54, 52] et d'exécution rapide des applications [54], la simplicité de test [55] et la formation de nouvelles personnes [19] font partie des paramètres discutés pour le critère « utilisabilité ».

Le critère « évolutivité » porte sur la capacité à faire évoluer l'application [15]. Le framework ne doit pas limiter les possibilités d'exécution du projet [53]. Autrement dit, le framework doit permettre de répartir la charge d'une application sur différents serveurs.

La « documentation » est un critère pertinent, cependant la documentation ne se crée pas de la même manière [50]. Certains paramètres à prendre en compte sont : la qualité [50, 15], elle permet une prise en charge facile [19] en proposant des tutoriels [15] ou du matériel pédagogique [53], des exemples pratiques [50] et des moteurs de recherche [50].

Nous collectons quelques paramètres pour le critère « maintenabilité », il fait référence à la santé du framework, il est indispensable de savoir si le framework est en bon état et s'il continue d'être maintenu régulièrement [50]. Parmi les paramètres présentés dans ce critère, nous avons la licence [15], les releases [13] [50, 55, 51], les releases indiquent si le framework est en cours de développement actif ou non [15], les issues [14] [50, 51], les pull requests [15] [50], les contributeurs [50], date de création [51, 48] et les dépendances NPM, c'est-à-dire de combien de paquets dans NPM dépend le framework [50, 51].

---

13. En français, nous pouvons utiliser le mot *livraison* pour *release*. Par contre, nous avons les garder le mot *release* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

14. En français, nous pouvons utiliser le mot *problèmes* pour *issues*. Par contre, nous avons les garder le mot *issues* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

15. En français, nous pouvons utiliser le mot *demande de tirage* pour *pull request*. Par contre, nous avons les garder le mot *pull request* pour son originalité et surtout parce qu'il est aussi utilisé comme tel dans les publications en français.

La connaissance et la familiarité [15] avec le framework sont des paramètres du critère « expérience », et sont basées sur l'expérience personnelle du développeur [42], donc il augmente la productivité dans la création et la maintenance des applications et interfère dans le coût de développement [55, 56].

Parmi les paramètres discutés pour le critère « sécurité », nous avons, en premier lieu, si les frameworks fournissent un appui pour faire face aux attaques Web [15]. Ensuite, si les vulnérabilités Node.js affectent le framework [52].

Le critère des « promesses » n'est mentionné qu'une seule fois. Par conséquent, il est important que le framework prenne en charge les fonctions de routage asynchrones [50].

Ainsi, nous présentons ci-dessous les résultats de notre première question de recherche sous forme de tableau 3.1.

TABLEAU 3.1 : Liste des critères plus cités dans les blogs

Critères	Sources	Total
communauté	[46], [50], [55], [42], [59], [53], [19], [51], [56], [54], [52], [47], [48], [11], [49], [15]	16
le but	[46], [50], [55], [57], [42], [53], [56], [10], [47], [48], [11], [14], [49], [15]	14
style	[46], [50], [55], [59], [53], [10], [54], [11], [15]	9
performance	[46], [55], [19], [56], [52], [47], [11], [15]	8
utilisabilité	[55], [19], [54], [52], [47], [11], [15]	7
évolutivité	[46], [55], [53], [56], [47], [15]	6
documentation	[46], [50], [55], [53], [19], [15]	6
maintenabilité	[50], [55], [51], [48], [15]	5
expérience	[50], [42], [56], [15]	4
sécurité	[52], [15]	2
promesses	[50]	1

3.3.2 QR2 : QUELS SONT LES FRAMEWOKS NODE.JS PLUS CITÉS ?

Nous présentons ici les résultats de notre deuxième question de recherche qui nous aide à cartographier les principaux frameworks Node.js. De plus, cela nous permet d'analyser les caractéristiques qui les rendent populaires. Les frameworks les plus cités ou indiqués parmi tous les articles sont listés dans le tableau ci-dessous 3.2. De plus, l'annexe A montre un tableau avec les résultats détaillés avec les articles qui citent chaque framework.

Tous ces frameworks mentionnés sont des frameworks Node.js. Chaque framework offre aux développeurs ses principales fonctionnalités [59], mais tous ne sont pas construits avec la même intention [19]. Par exemple, parmi les dix les plus cités, Express, Koa et Hapi sont des frameworks permettant de développer des applications capables de gérer des requêtes via HTTP [19]. Certains d'entre eux supportent une architecture logicielle, LoopBack pour le développement d'API REST [59], Sails [48] et Adonis [59] les plus utilisés pour le MVC, enfin, Feathers utilisé pour les applications temps réel [42]. Meteor est un framework Full-Stack qui fournit une structure et des fonctionnalités pour votre application [11]. Total.js est l'un des nombreux frameworks qui offre une expérience unique dans un Système de gestion de contenu (CMS) [14]. D'autres ont plus de fonctionnalités, comme Nest.js, et peuvent décider d'utiliser les fonctionnalités et d'organiser le code de manière avisée [53].

Parmi les trente frameworks recensés, le framework Express n'est pas mentionné dans un seul article, il a donc eu 17 nominations. Ensuite, le framework Sails a 15 nominations et peu de temps après, le framework Koa avec 14 citations.

TABLEAU 3.2 : Liste des frameworks Node.js plus cités dans les blogs

Framework	Total
Express	17
Sails	15
Koa	14
Hapi	12
Meteor	12
LoopBack	11
Nest.js	11
Adonis	8
Total.js	7
Feathers	7
Socket.io	6
Derby.js	5
Restify.js	4
Mean.js	3
Keystone.js	3
Kraken.js	2
Strapi	2
Next.js	2
Nuxt	2
Mocha.js	2
Fastify	1
Mojito	1
Electron.js	1
Diet.js	1
ActionHero.Js	1
Sequelize	1
Molecular	1
Flatiron.Js	1
FoalTS	1
VulcanJS	1

3.4 CONSIDÉRATIONS FINALES

Dans cette section, nous discutons des résultats de nos questions de recherche. Nous mettons en évidence les 30 frameworks les plus recommandés pour la technologie Node.js (QR2 3.3.2). Nous identifions un ensemble composé de 13 critères, ainsi que de 33 paramètres liés au thème du choix d'un framework à utiliser dans les projets logiciels que les développeurs peuvent utiliser. Nous créons une compilation des critères et de leurs paramètres dans le tableau 3.3. Les critères et paramètres les plus importants adoptés pour évaluer et sélectionner un framework Node.js (QR1 3.3.1) sont « Communauté », « Le but » et « Style ». Par contre, « Promesses » et « Sécurité » sont les critères les moins cités.

TABLEAU 3.3 : Liste des critères et leurs paramètres
© Mário Danilo Alves da Silva, 2022

Critères	Paramètres
Communauté	Étoile dans GitHub
	Dépendant dans GitHub
	NPM téléchargement
	Plateformes technologiques
Le but	Architecture
	TypeScrip
Style	Fonctionnalités incluses
	CLI
	La journalisation
	Abstractions
	Opiniâtre
Performance	Requêtes par seconde
Utilisabilité	Mise en place
	Création
	Mettre à disposition en production
Évolutivité	Outils d'évolutivité
Documentation	Tutoriels
	Exemples pratiques
	Moteurs de recherche
	Bonne structure
Maintenabilité	Releases
	Pull requests
	Issues
	Dépendances NPM
	Date de création
	Licence
	Contributeurs
	Activité de contribution
	Répartition des cotisations
Expérience	Expérience personnelle
Sécurité	Outils standards
	Outils tiers
	Recommande les bonnes pratiques
Promesses	

Certains des critères et paramètres adoptés pour évaluer et sélectionner un framework Node.js sont similaires à ceux trouvés dans d'autres travaux. Le critère « style » est similaire au critère « fonctionnalité » dans les articles [37, 36, 39, 40, 41], qui utilisent ce critère pour sélectionner ETL, dossier médical électronique, LMS, logiciel d'exploration de données et paquets logiciels. La fonctionnalité est la capacité du logiciel à fournir des fonctions qui répondent aux besoins des utilisateurs lors de l'utilisation du logiciel dans des conditions spécifiques [60]. La plupart de ces systèmes partagent un ensemble commun de fonctionnalités de base.

Actuellement, la tendance est de rechercher un framework qui a une grande popularité dans la communauté. Selon les auteurs [46, 50, 48], avec une grande communauté qui apporte fréquemment des contributions, cela augmente la facilité d'obtenir des conseils et de trouver des bibliothèques écrites pour fonctionner avec le framework [50, 53, 54]. Les travaux de [37, 41] mettent également en évidence la popularité du fournisseur, le qualifiant de sous-critère du critère « fournisseur ».

Un autre facteur qui prend beaucoup d'importance est « Le but ». Cela signifie que pendant que nous évaluons ces frameworks les uns par rapport aux autres, le choix du framework dépend de l'objectif de l'application [46, 50, 55, 56, 14].

Le critère « utilisabilité » n'est pas seulement évoqué dans nos travaux, mais aussi dans les travaux de [37, 36, 39, 40, 41], où ils mettent en avant facilité d'utilisation, facilité d'installation, satisfaction, courbe d'apprentissage, facilité de disponibilité à l'usage. L'utilisabilité établit à quel point un système est efficace, pratique et facile à apprendre [61].

Le travail de Zaidan et al. [36], fait référence à deux critères, le critère d'appui utilisateur et le critère d'appui développeur, ces critères sont similaires aux paramètres du critère « documentation » trouvés, en somme, ils traitent des différentes sources que les utilisateurs

ou développeurs préfèrent se renseigner sur un système et obtenir de l'aide. Jadhav et Sonar [41] traitent les paramètres du tutoriel et les exemples pratiques comme critères de sélection, évaluant la disponibilité du tutoriel et du guide de dépannage pour apprendre à utiliser le paquet logiciel.

Une évaluation générale de la sécurité des frameworks dépasse l'étendue de cette recherche, l'évaluation de la sécurité nécessite une analyse approfondie sous de nombreux aspects [62]. Jadhav et Sonar [41] et Abdullateef et al. [39] utilisent le critère de « sécurité » pour vérifier l'amplitude des politiques de sécurité prises en charge par le paquet logiciel, telles que l'identification de l'utilisateur, l'audit, le chiffrement des données.

Les paramètres de « maintenabilité » nécessitent une évaluation à long terme dans le monde réel [39]. Les modifications peuvent inclure des corrections, des améliorations ou une adaptation du logiciel aux changements de l'environnement, des exigences et des spécifications fonctionnelles [60]. Hanine et al. [37] traitent du coût de la maintenance. Jadhav et Sonar [41] traitent de la taille moyenne des unités de code indépendantes et du niveau d'indépendance entre les modules.

Cependant, Jadhav et Sonar [41] font référence à la capacité de séparer les applications en paquets pouvant être distribués sur différents serveurs en tant que « maintenabilité », en plus de spécifier un critère « évolutivité » qui fait référence à la capacité du paquet logiciel à gérer un nombre croissant d'utilisateurs et charge de transaction plus élevée. Ces deux dernières définitions sont liées à notre critère « évolutivité ».

Le critère de « performance » est lié à la question de savoir si le framework peut répondre aux besoins d'efficacité d'affaires [19, 11]. Les auteurs [37, 39] traitent ce critère comme la stabilité et la fiabilité du logiciel. Les auteurs Jadhav et Sonar [41] utilisent un ensemble de critères pour le critère « performance » pour sélectionner les paquets logiciels, les critères

traitent de la variété de la plateforme, de l'architecture logicielle, de l'accès aux données hétérogènes, de la taille des données, de l'efficacité, de l'interopérabilité et de la robustesse.

L'importance d'un critère n'est pas nécessairement proportionnel au nombre de paramètres ou au nombre de citations. En précisant l'importance des critères de sélection, on peut dire que certains paramètres ne sont pas des facteurs pour sélectionner un framework. Les résultats de ce travail peuvent être utiles aux professionnels qui ont besoin de sélectionner un framework Node.js. La liste de critères peut être utilisée efficacement comme liste de contrôle pour vérifier que toutes les caractéristiques potentiellement importantes du framework sont correctement évaluées.

CHAPITRE IV
LE MODÈLE DE PRISE DE DÉCISION

Ce chapitre traite de notre proposition de modèle décisionnel multicritère. Notre modèle vise à soutenir la prise de décision pour la sélection du framework Node.js. Le chapitre est divisé en quatre parties, la première apporte une brève introduction à la méthodologie multicritère AHP, qui est largement utilisée par les analystes et les décideurs dans la résolution de problèmes complexes. La deuxième partie traite de la sélection des critères du modèle, qui s'appuie sur les résultats obtenus dans le chapitre 3. La troisième partie aborde la définition des critères du modèle. Enfin, la quatrième partie montre la synthèse de la méthode AHP et comment l'appliquer.

4.1 LA MÉTHODE AHP

La méthode multicritère Processus d'analyse hiérarchique (AHP) a été développée par le mathématicien Thomas L. Saaty dans les années 1970 pour l'analyse décisionnelle et la planification multicritères. La méthode AHP a été développée pour modéliser des problèmes non structurés dans la vie quotidienne des gens, alors qu'ils prennent des décisions sans nécessairement avoir la notion exacte de l'importance des paramètres utilisés [63, 64].

La méthode AHP est largement utilisée et se caractérise par la décomposition d'un problème discret dans une structure hiérarchique descendante, au sommet de la hiérarchie on a l'objectif global, au deuxième niveau on a les critères et enfin les alternatives [65]. La figure 4.1 résume la structure hiérarchique standard de la méthode AHP.

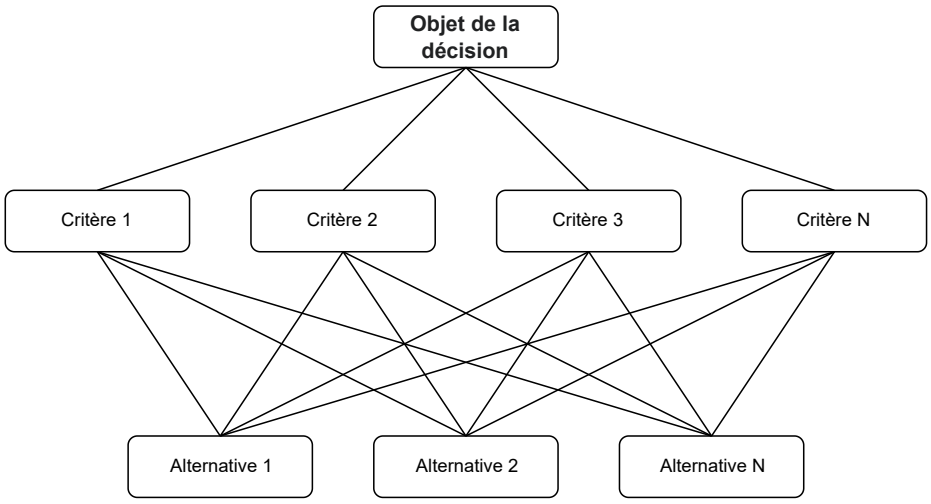

FIGURE 4.1 : La structuration hiérarchique par défaut de la méthode AHP
© Mário Danilo Alves da Silva, 2022

La méthode fournit au décideur une meilleure évaluation et une meilleure compréhension du problème de décision [66]. La méthode tient compte de l'importance relative des facteurs analysés [67, 68], car les préférences des décideurs sont élucidées à travers les jugements subjectifs des participants, ce qui leur permet de quantifier numériquement leurs jugements. Selon Hartman et Goltz [69], l'utilisation de la méthode AHP convient aux situations dans lesquelles les critères analysés par les utilisateurs de la méthode sont contradictoires.

Par conséquent, cette méthode AHP fournit une procédure parfaitement compréhensible et rationnelle pour modéliser un problème de décision, car elle mesure des critères tangibles avec des intangibles pondérés par les préférences. Cette méthode de décision devient très utile pour réduire les échecs, car le résultat est un modèle qui vous permet d'analyser plusieurs alternatives et de les comparer rapidement [68].

4.2 SÉLECTION DES CRITÈRES DU MODÈLE

Un aspect important de la méthode AHP est que le nombre d'éléments dans chaque niveau hiérarchique doit être compris entre 2 et 7 [18]. Selon Miller [70], il s'agit de la limite supérieure de la capacité humaine à traiter l'information et à comparer les éléments avec précision. De plus, Saaty et Ozdemir [71] rapportent que grâce à l'indice de cohérence d'un grand nombre d'éléments comparés, l'incohérence diminue si lentement qu'elle est insuffisante pour améliorer la cohérence. Ce sont deux raisons que nous avons prises en compte pour sélectionner les critères du modèle.

À travers les résultats de nos questions de recherche dans le chapitre 3, nous sélectionnons les sept critères qui avaient le plus de citations. Les aspects suivants sont : « communauté », « le but », « style », « performance », « utilisabilité », « évolutivité » et « documentation ». Après avoir sélectionné les sept premiers critères, nous décidons de supprimer le critère « le but » du modèle, car nous devons d'abord établir l'objectif du projet [46, 50, 55, 56, 14]. Ensuite, nous déterminons si le framework convient aux besoins de l'application [53, 56, 11, 49]. Ainsi, établir le framework comme une alternative au modèle. Ainsi, notre modèle établit 6 critères qui appartiennent au même niveau d'importance.

4.2.1 COMMUNAUTÉ

La communauté existante autour d'un framework est important [50], comme on peut le voir à partir de l'analyse du chapitre 3, que 16 articles soulignent l'importance de la communauté. La popularité de la communauté est un paramètre de préférence parmi les développeurs [46, 19, 47, 48, 11, 49]. La communauté est un endroit où nous pouvons trouver de l'aide lorsque nous avons un problème avec le framework [46, 50, 48]. Une communauté

active qui aide les nouveaux utilisateurs est l'une des ressources les plus précieuses pour les développeurs [15].

4.2.2 STYLE

Nous sélectionnons le critère « style » pour notre modèle, car il s'agit du troisième critère le plus cité parmi les articles, avec 14 citations. Ce critère est important, car il fait référence à des frameworks qui offrent des fonctionnalités intégrées [59, 53, 15] et qui structurent le code de manière spécifique afin que vous puissiez profiter des fonctionnalités qu'ils offrent [50, 10, 11], les rendant ainsi « opiniâtres » ou « non opiniâtres » par rapport au code [50, 15].

4.2.3 PERFORMANCE

Nous sélectionnons le critère « performance », car il obtient 8 citations parmi les articles de l'analyse du chapitre 3. Lorsque l'ossature est soumise aux conditions d'utilisation prévues, un comportement correct est essentiel. De cette façon, nous nous attendons à ce que le framework charge les ressources système [55, 56, 52], soit rapide [15] et réponde aux besoins d'affaires [19, 11]. Exécuter un simple test de performance avant de choisir un framework s'avère être une tâche intéressante [52].

4.2.4 UTILISABILITÉ

Une adaptation rapide au framework est indispensable [47, 15]. La courbe d'apprentissage représente la facilité d'interaction de l'utilisateur avec un framework [54, 52, 47, 11, 15], la facilité d'utilisation [11], la simplicité de création [55, 54, 52] et d'exécution des applications [54]. Sur la base de ces facteurs et de l'analyse dans le chapitre 3 qu'il obtient 7 citations, nous sélectionnons ce critère pour le modèle.

4.2.5 ÉVOLUTIVITÉ

Le critère « évolutivité » a été retenu, puisqu'il obtient 7 citations dans l'analyse du chapitre 3. Ce critère porte sur la capacité du framework à permettre à l'application d'évoluer vers le haut ou vers le bas, en termes de performances ou de coût [15]. Le framework ne doit pas limiter les possibilités d'exécution de l'application [53]. L'évolutivité du framework est essentielle pour assurer la personnalisation de nouveaux services et fonctionnalités [72].

4.2.6 DOCUMENTATION

La documentation est sélectionnée pour notre modèle, car elle obtient 6 citations entre les articles du chapitre 3. Selon les articles [46, 50], il faut éviter d'utiliser le framework s'il ne fournit pas de documentation, ou si la documentation est mal écrite et ne vous aide pas. Lorsque nous utilisons un framework pour la première fois, nous pouvons avoir du mal à utiliser ses ressources [19]. Une fois que le framework fournit de la documentation, nous pouvons l'utiliser pour résoudre nos doutes [50].

4.3 DÉFINITION DES PARAMÈTRES DES CRITÈRES DU MODÈLE

Dans cette section nous définissons les paramètres d'évaluation et de sélection de notre modèle pour les critères que nous avons sélectionnés dans la section 4.2, les critères suivants sont : « communauté », « style », « performance », « utilisabilité », « évolutivité » et « documentation » . Les paramètres établis dans cette section sont destinés à soutenir la comparaison par paires d'alternatives, par conséquent, ils ne sont pas établis à un niveau hiérarchique de notre modèle.

4.3.1 COMMUNAUTÉ

Pour définir la popularité d'un framework dans la communauté, nous devons vérifier si le framework est utilisé par d'autres développeurs ou s'il est largement adopté [50, 53, 54]. Nous utiliserons les paramètres suivants pour mesurer la taille et la popularité de la communauté :

- Nombre d'étoiles dans le dépôt du framework sur GitHub [50, 51, 52]
- Nombre de projets dépendants du framework sur GitHub [51]
- Nombre de suiveurs sur Stack Overflow [46, 50]
- Nombre de téléchargements hebdomadaires du framework sur la page NPM [50, 51, 52]

Pour obtenir les informations sur le nombre d'étoiles et le nombre de personnes à charge des frameworks, nous devons visiter les référentiels de frameworks sur GitHub. La page du framework sur GitHub fournit un bouton avec le nombre d'étoiles en temps réel du référentiel. Il fournit également un outil d'analyse dans le menu Insights [16], qui fournit des informations sur le projet, les équipes et le nombre de personnes à charge.

Concernant le nombre de suiveurs de la plateforme Stack Overflow, nous utilisons le nombre de suiveurs du « Tag » de la plateforme faisant référence aux frameworks. Pour obtenir le nombre de téléchargements hebdomadaires, nous utilisons NPM Trends, nous extrayons donc le nombre de téléchargements hebdomadaires pour chaque framework.

4.3.2 STYLE

Pour définir le « style » du framework, il faut vérifier les fonctionnalités incluses que le framework propose [50, 15]. Ces fonctionnalités sont des outils : d'aide à la validation des

16. Les GitHub Insights d'une organisation sont disponibles avec GitHub Enterprise Cloud, disponible sur https://docs.github.com.

données, de gestion de la journalisation, d'aide à l'authentification, de réalisation d'abstractions de base de données, de documentation de l'application, de test, de dépendance et d'injection CLI. Déterminez comment le framework diffère de son style, « opiniâtre » ou « non opiniâtre » [50, 15]. Toutes ces définitions peuvent être vérifiées dans la documentation du framework. Nous évaluons l'agilité du développement de code [42] avec un framework « opiniâtre » [50, 15] et avec des fonctionnalités intégrées [50, 10, 15].

4.3.3 PERFORMANCE

Le test de charge est une évaluation des performances effectuée par un système externe qui envoie des demandes répétées à l'application [73]. Cela vous permet d'identifier rapidement les goulots d'étranglement [55, 56, 52] et la vitesse [15] d'application.

Nous déterminons les performances en mesurant le nombre de requêtes par seconde prises en charge par le framework [19, 52, 11]. Nous pouvons utiliser des outils pour aider à comprendre les caractéristiques de performance [19]. Des outils, comme Autocannon [17] , nous aident à comprendre et à améliorer les caractéristiques de performance des applications et du framework. Le framework Fastify fournit un tutoriel pour effectuer des tests de performances dans sa documentation [74]. Une autre façon d'obtenir les résultats de performance des frameworks est d'utiliser des données de référence provenant d'entreprises spécialisées.

4.3.4 UTILISABILITÉ

Pour avoir une idée approximative de la courbe d'apprentissage des frameworks, nous devons implémenter une simple de *Getting Started* dans la documentation de chaque framework. Bien que ces exemples d'implémentation disponibles dans la documentation ne soient pas des

17. Disponible sur : https://github.com/mcollina/autocannon

exemples super complexes, ils contribuent à donner une idée de ce que ce serait d'implémenter une application à partir de zéro en utilisant le framework. Notre intention est de voir à quelle vitesse nous pourrions démarrer une application simple. Nous devons également mesurer la courbe d'apprentissage en vérifiant les conditions minimales pour exécuter l'application en production [54, 11], établissant ainsi la vitesse pour rendre l'application disponible en production.

4.3.5 ÉVOLUTIVITÉ

Il existe deux types d'évolutivité de base dans l'informatique : la mise à l'échelle et la mise à l'échelle. L'évolutivité verticale est limitée au matériel serveur ou de la machine, tel que le stockage, la mémoire et le processeur [75]. La mise à l'échelle ajoute plus de ressources telles que des serveurs à votre système pour répartir la charge de travail sur les machines, ce qui à son tour augmente les performances et la capacité de stockage [76].

Pour évaluer le critère « évolutivité » du modèle, il faut rechercher les caractéristiques des frameworks favorisant la création d'applications évolutives. Nous devons vérifier que le framework fournit des méthodes, des concepts ou des outils qui permettent l'évolutivité des applications [15, 77].

4.3.6 DOCUMENTATION

Nous devons identifier si le framework fournit de la documentation, si le framework ne fournit aucune documentation, ou si la documentation est mal écrite et n'aide pas, vous devriez probablement éviter de l'utiliser [46, 50]. Ce premier paramètre consiste à déterminer si la documentation existe. Nous devrions rechercher la documentation sur le site officiel ou le référentiel GitHub du framework.

Pour définir la qualité de la documentation d'un framework, nous utiliserons les paramètres suivants : disponibilité de tutoriels [15], disponibilité d'exemples pratiques [50] et disponibilité d'un appui facile avec des moteurs de recherche [50] et une bonne structure de menu [19].

4.4 MÉTHODOLOGIE DE LA MÉTHODE AHP

La méthodologie de la méthode AHP comprend cinq étapes [64]. Nous énumérons ci-dessous les étapes pour résoudre le problème de sélection d'un framework Node.js qui convient le mieux à l'application, puis nous vous expliquerons comment postuler.

1. Identifier un ensemble de critères et d'alternatives pour construire la hiérarchie
2. Évaluation par paires des critères
3. Estimation de l'importance relative (poids) des critères
4. Déterminer le niveau de préférence des alternatives
5. Valorisation globale des alternatives

4.4.1 IDENTIFIER UN ENSEMBLE DE CRITÈRES ET D'ALTERNATIVES POUR CONSTRUIRE LA HIÉRARCHIE

Dans la section 4.2, nous identifions l'ensemble de critères de notre modèle de prise de décision multicritère et il est organisé en 6 critères. Nous montrons sur la figure 4.2 la structure hiérarchique comme objectif au premier niveau, les critères au deuxième niveau et au troisième niveau nous avons les cadres comme alternatives du modèle basé sur la méthode AHP, afin de prendre en charge le processus de sélection des frameworks Node.js.

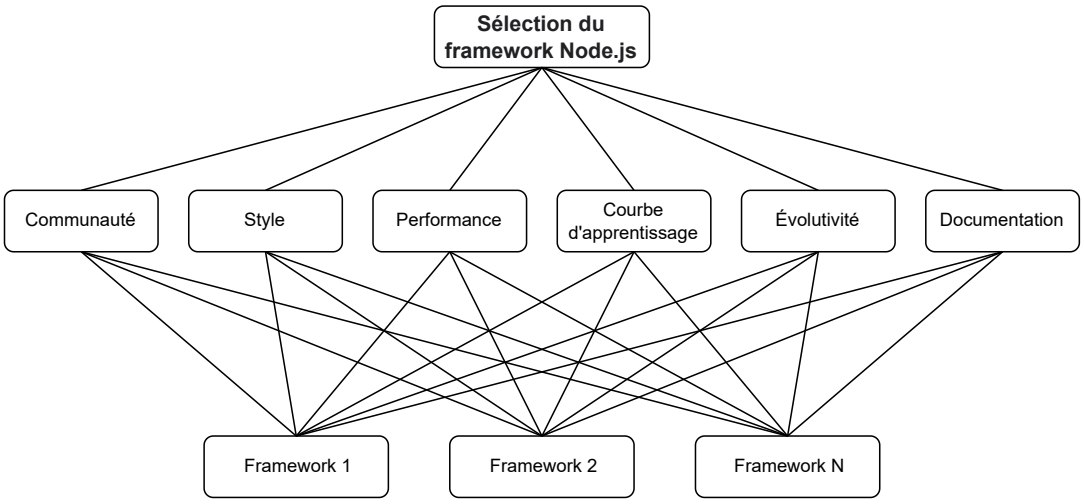

FIGURE 4.2 : La structuration hiérarchique de notre modèle décisionnel multicritère
© Mário Danilo Alves da Silva, 2022

4.4.2 ÉVALUATION PAR PAIRES DES CRITÈRES

Après une hiérarchie établie, le décideur doit effectuer une comparaison par paires de chaque élément au niveau hiérarchique, créant une matrice de décision carrée [64]. En comparant les éléments E_n, où n est le nombre d'éléments de même niveau hiérarchique, en ligne i par rapport aux éléments de colonne j, avec $i, j = 1, 2, ..., n$, les poids fournis par le décideur, sera w_{ij}, et la préférence de l'élément E_i sur l'élément E_j par rapport à l'élément du niveau supérieur est égale à w_{ij}. Par conséquent, dans la comparaison de E_j par rapport à E_i, il insère la valeur égale à $\frac{1}{w_{ij}}$ dans la matrice, puisque les valeurs sont réciproques.

La diagonale principale de la matrice représente la comparaison d'un élément avec lui-même, donc les valeurs de la diagonale principale sont toujours 1 (un), pour tout w_{ii}. Le nombre de comparaisons requises est défini par la formule $\frac{n \times (n-1)}{2}$ [78].

Cette matrice est la représentation du résultat de l'étape de comparaison des éléments, comme on le voit dans le tableau 4.1. Cette même comparaison est faite entre critères, et entre

alternatives A_n pour chaque critère C_n du modèle, c'est-à-dire que chaque critère génère une matrice de comparaison entre alternatives.

TABLEAU 4.1 : Modèle de tableau de comparaison binaire entre les éléments
© Mário Danilo Alves da Silva, 2022

Élément	E_1	E_2	E_3
E_1	1	w_{12}	w_{13}
E_2	$\frac{1}{w_{12}}$	1	w_{23}
E_3	$\frac{1}{w_{13}}$	$\frac{1}{w_{23}}$	1

Dans cette matrice, le décideur représentera, à partir d'une échelle prédéfinie, sa préférence parmi les éléments comparés, sous le focus d'un élément du niveau supérieur [66]. Par exemple, nous devons demander : *« Comment le critère C_i satisfait-il mieux le **but** que le critère C_j ? »* ou *« Comment l'alternative A_i satisfait-elle le **critère** C_1 meilleur que l'alternative A_j ? »*. Pour formuler ses priorités, le décideur répond à la question par un nombre qui reflète l'appréciation verbale de l'importance relative de chacun des éléments dans l'environnement décisionnel [64]. Dans cette méthode, l'échelle de 1 à 9 de comparaison binaire présentée dans le tableau 4.2 est utilisée.

TABLEAU 4.2 : Échelle de comparaison binaire fondamentale, par Saaty [65]
Reproduit avec l'autorisation de l'éditeur

Degré d'importance	Appréciation	Explication
1	Importance égale de deux éléments	Les deux activités contribuent à parts égales à l'objectif.
3	Faible importance d'un élément par rapport à un autre	L'expérience et le jugement favorisent une activité plutôt qu'une autre.
5	Importance forte d'un élément par rapport à un autre	L'expérience et le jugement favorisent fortement une activité plutôt qu'une autre.
7	Importance très grande ou démontrée d'un élément par rapport à un autre	Une activité est très fortement favorisée par rapport à l'autre. Cela peut être démontré dans la pratique.
9	Importance absolue d'un élément par rapport à un autre	Les preuves favorisent une activité plutôt qu'une autre, avec le plus haut degré de certitude.
2, 4, 6, 8	Valeurs intermédiaires	Lorsque l'on recherche une condition de compromis entre deux définitions.

4.4.3 ESTIMATION DE L'IMPORTANCE RELATIVE (POIDS) DES CRITÈRES

Après la comparaison de parité, les valeurs attribuées sont normalisées où les vecteurs propres maximaux V_i et le ratio de cohérence RC sont calculés.

Nous aborderons l'algorithme appelé la méthode de moyenne des valeurs normalisées pour obtenir le vecteur de priorité. Cet algorithme est illustré à l'aide de la matrice du tableau 4.1. La moyenne des valeurs normalisées comprend les étapes suivantes : normalisation des valeurs de chaque colonne, calcul du vecteur propre maximal et calcul du ratio de cohérence.

Le calcul de la normalisation des valeurs se fait d'abord en calculant les sommes des poids de l'évaluation de chaque colonne SC_j, exemplifiées dans l'équation 4.1.

$$SC_j = \sum_{i=1}^{n} w_{ij} \tag{4.1}$$

Ensuite, nous calculons la matrice de comparaison par paires normalisée W_{ij}, en divisant chaque élément w_{ij} de la matrice d'origine 4.1 par la somme des poids d'évaluation SC_j, exemplifiés dans l'équation 4.2.

$$W_{ij} = \frac{w_{ij}}{SC_j} \tag{4.2}$$

Pour calculer le vecteur propre maximum V_i, calculez la moyenne des lignes de la nouvelle matrice normalisée, en ajoutant les valeurs de ligne et en divisant par le nombre d'éléments, illustrés dans l'équation 4.3. De cette manière, on retrouve le poids de chaque critère lors du choix d'un framework Node.js. Nous donnons un aperçu des calculs dans le tableau 4.3.

$$V_i = \frac{\sum_{i=1}^{n} W_{ij}}{n} \tag{4.3}$$

TABLEAU 4.3 : Modèle de tableau de calcul du vecteur propre maximal

Élément	E_1	E_2	E_3	Vecteurs propres
E_1	W_{12}	W_{12}	W_{13}	V_1
E_2	W_{22}	W_{22}	W_{23}	V_2
E_3	W_{32}	W_{32}	W_{33}	V_3
Somme C	SC_1	SC_2	SC_3	

RATIO DE COHÉRENCE DE LA MATRICE DE DÉCISION

Après avoir trouvé le poids de chaque critère, nous devons vérifier que nous avons fait une comparaison par paire cohérente. Selon Saaty [64], le concept de cohérence est basé sur l'idée que lorsqu'une quantité de base de jugements de une matrice est faite, toutes les autres données peuvent en être logiquement déduites. En supposant que $c1 = 2 \times c2$, $c2 = 2 \times c3$ et, par conséquent, $c1 = 4 \times c3$.

L'écart des jugements à la cohérence est mesuré par l'indice de cohérence IC, exemplifié dans l'équation 4.4, plus l'indice est proche de zéro, meilleure est la cohérence globale de la matrice de comparaison des jugements [63].

$$IC = \frac{\lambda_{max} - n}{n - 1} \tag{4.4}$$

L'étape suivante consiste à calculer la valeur propre maximale λ_{max}, qui est la moyenne de la valeur de chaque élément du vecteur lambda λ_n, exemplifié dans l'équation 4.4.

$$\lambda_{max} = \frac{\sum_{j=1}^{n} \lambda_{ij}}{n} \tag{4.5}$$

Enfin, le calcul de λ_n se fait par le produit matriciel de la matrice d'origine 4.1 par le vecteur propre maximum V_n, qui est défini dans le tableau 4.3, divisé par le poids du vecteur propre maximum V_n, le calcul est exemplifié dans l'équation 4.6. Le résultat est une matrice avec le même nombre de lignes que la matrice d'origine et avec le même nombre de colonnes que le vecteur propre maximum.

$$\lambda_n = \frac{\sum_{k=1}^{n} w_{kj} \times V_{ik}}{V_{kk}} \tag{4.6}$$

Le ratio de cohérence *RC* est mesuré par l'indice de cohérence aléatoire *IR*, qui est déterminé par des expériences et après tabulation. Le tableau 4.4 montre l'ordre des matrices avec leurs RI correspondants. Le *IR* utilisé a la même dimension que *n* comme *IC*.

TABLEAU 4.4 : Cohérence aléatoire, par Saaty [65]
Reproduit avec l'autorisation de l'éditeur

Nombre de critères *n*	1	2	3	4	5	6	7	8	9	10
Cohérence aléatoire *RI*	0	0	0,52	0,89	1,11	1,25	1,35	1,40	1,45	1,49

Le calcul du ratio de cohérence *RC* est illustré dans l'équation 4.7. Si le ratio de cohérence est inférieur à $0,1$, les jugements dans la matrice de décision sont cohérents, sinon il y a une certaine insistance sur les jugements, par conséquent, le juge doit revoir ses jugements.

$$RC = \frac{IC}{IR} \tag{4.7}$$

4.4.4 DÉTERMINATION DU NIVEAU DE PRÉFÉRENCE DES ALTERNATIVES

Dans cette étape, l'évaluation par paires des alternatives est effectuée, les frameworks Node.js, au sein de chaque critère, pour tous les critères préférés sélectionnés dans la section 4.2, écrivent une matrice de comparaison. Nous devons suivre les étapes similaires à la sous-section 4.4.2. De plus, nous devons obtenir le vecteur propre maximum des alternatives pour chaque critère, ainsi que, nous devons évaluer la constance des jugements de cette étape, similaire à la sous-section 4.4.3.

4.4.5 VALORISATION GLOBALE DES ALTERNATIVES

Après avoir obtenu les vecteurs propres de chaque alternative $V_{alternative}$ sous chaque critère C_n, et du vecteur propre des critères $V_{critere}$, nous pouvons établir une valorisation globale des priorités des alternatives. Pour mesurer la priorité des alternatives P_n, il faut calculer la somme pondérée des critères et des alternatives. C'est-à-dire une somme des poids des alternatives multipliée par les poids des critères, illustrés dans l'équation 4.8. Ce processus de comparaison fournit une échelle relative des mesures prioritaires des alternatives, la meilleure alternative étant celle qui obtient la note globale la plus élevée. Un aperçu de ces données est organisé dans le tableau 4.5.

$$P_n = \sum_{i=1}^{n} V_{critere_j} \times V_{alternative_{ij}} \tag{4.8}$$

TABLEAU 4.5 : Modèle de sélection de framework Node.js
© Mário Danilo Alves da Silva, 2022

Sélection du framework Node.js

Critères	$V_{critere_1}$	$V_{critere_2}$	$V_{critere_3}$	$V_{critere_n}$	Valorisation globale
Alternative 1	$V_{alternative_1}$	$V_{alternative_1}$	$V_{alternative_1}$	$V_{alternative_1}$	P_1
Alternative 2	$V_{alternative_2}$	$V_{alternative_2}$	$V_{alternative_2}$	$V_{alternative_2}$	P_2
Alternative n	$V_{alternative_n}$	$V_{alternative_n}$	$V_{alternative_n}$	$V_{alternative_n}$	P_n

CHAPITRE V
APPLICATION DU MODÈLE

Pour l'application de notre modèle décisionnel multicritère, nous développons une étude de cas pour sélectionner un framework Node.js. Elle consiste à appliquer notre modèle aux alternatives de frameworks Node.js établies dans le scénario de l'étude de cas et à sélectionner le framework qui correspond le mieux au scénario.

Ce chapitre est divisé en quatre parties, qui consiste à décrire le scénario de l'étude de cas, évaluer les critères, évaluer les alternatives de framework pour chaque critère et les résultats de l'application du modèle pour soutenir la prise de décision de l'étude de cas.

5.1 SCÉNARIO D'ÉTUDE DE CAS

Le scénario de cette étude de cas est basé sur une situation réelle d'une petite entreprise québécoise, afin de développer une application. L'objectif est de développer une application back-end pour répondre aux demandes formulées par l'application mobile et le site web de l'entreprise. Dans ce scénario, le directeur de la technologie établit que la technologie utilisée dans le back-end est Node.js, mais il ne définit pas de framework de développement. Parmi les frameworks disponibles sur le marché, le directeur de la technologie prédéfinit trois possibilités de frameworks d'utilisation, à savoir les frameworks Nest.js, Fastify et Hapi.

Parmi les exigences fonctionnelles et non fonctionnelles de l'application de scénario, nous énumérons quelques éléments pertinents pour déterminer quel framework est le mieux adapté au développement d'applications.

- Déterminer un framework qui a une adoption élevée et que vous pouvez trouver un support pour les problèmes et les bibliothèques
- Le temps de projet est court, un framework qui accélère le développement du code est d'une grande valeur
- Établir un framework qui fournit des fonctionnalités pour documenter l'application, développer des tests, faire des abstractions avec la base de données et la validation des données
- Répondre et prendre en charge le nombre de demandes d'application mobile et de site web
- Déterminer un framework qui n'a pas besoin de beaucoup de configurations pour démarrer le développement et qui a également peu d'exigences pour rendre l'application disponible aux clients
- Déterminer un framework qui favorise la division de l'application en parties plus petites, ou selon la nécessité
- Déterminer un framework qui fournit de la documentation et fournit des exemples de mise en œuvre de fonctionnalités
- L'équipe de développement n'a aucune connaissance et expérience dans ces frameworks

5.2 ÉVALUATION DES CRITÈRES

Dans cette partie, le décideur effectue une comparaison par paires des critères du modèle, puis fixe le poids de chaque critère. Pour les 6 critères établis dans notre modèle, le décideur doit faire 15 comparaisons. Tous les critères sont comparés entre eux et tous suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons avec l'objectif de sélectionner un framework Node.js.

1. Le critère « communauté » a une **faible importance** que le critère « style »
2. Le critère « communauté » a une **importance très grande** que le critère « performance »
3. Le critère « communauté » a une **faible importance** que le critère « utilisabilité »
4. Le critère « communauté » a une **importance forte** que le critère « évolutivité »
5. Le critère « communauté » a une **importance égale** que le critère « documentation »
6. Le critère « style » a une **importance forte** que le critère « performance »
7. Le critère « style » a une **importance égale** comme critère « utilisabilité »
8. Le critère « style » a une importance intermédiaire entre une **faible importance** et une **importance forte** que le critère « évolutivité »
9. Le critère « documentation » a une **faible importance** que le critère « style »
10. Le critère « utilisabilité » a une **importance forte** que le critère « performance »
11. Le critère « évolutivité » a une **importance forte** que le critère « performance »
12. Le critère « documentation » a une **importance très grande** que le critère « performance »
13. Le critère « utilisabilité » a une importance intermédiaire entre une **faible importance** et une **importance forte** que le critère « évolutivité »
14. Le critère « documentation » a une **faible importance** que le critère « utilisabilité »
15. Le critère « documentation » a une **importance forte** que le critère « évolutivité »

Ces résultats de comparaisons parmi les critères sont représentés dans un tableau, on peut le voir dans le tableau 5.1.

TABLEAU 5.1 : Matrice de comparaison par paires entre critères (C_1 = « communauté », C_2 = « style », C_3 = « performance », C_4 = « utilisabilité », C_5 = « évolutivité », C_6 = « documentation »)

Critères	C_1	C_2	C_3	C_4	C_5	C_6
C_1	1	3	7	3	5	1
C_2	$\frac{1}{3}$	1	5	1	4	$\frac{1}{3}$
C_3	$\frac{1}{7}$	$\frac{1}{5}$	1	$\frac{1}{5}$	$\frac{1}{5}$	$\frac{1}{7}$
C_4	$\frac{1}{3}$	1	5	1	4	$\frac{1}{3}$
C_5	$\frac{1}{5}$	$\frac{1}{4}$	5	$\frac{1}{4}$	1	$\frac{1}{7}$
C_6	1	3	7	3	7	1

Après la comparaison par paires des critères, on calcule le vecteur propre maximum et puis le ratio de cohérence de la matrice 5.1. On commence par la normalisation des valeurs de la matrice de comparaison par paires entre critères, en suivant les étapes définies dans la sous-section 4.4.3. Le calcul de la matrice de comparaison par paires normalisée est exemplifié dans l'équation 5.1.

$$W_{11} = \frac{1}{(1+0,33+0,14+0,33+0,20+1)} = 0,33 \tag{5.1}$$

Nous exemplifions le calcul du vecteur propre maximum de la nouvelle matrice normalisée dans l'équation 5.2. De cette manière, nous trouvons le poids de chaque critère lors du choix d'un framework Node.js. Un aperçu des résultats est présenté dans le tableau 5.2.

$$V_1 = \frac{(0,33+0,36+0,23+0,36+0,24+0,34)}{6} = 0,31 \tag{5.2}$$

TABLEAU 5.2 : Matrice de comparaison par paires normalisée entre les critères et du vecteur propre maximum(C_1 = « communauté », C_2 = « style », C_3 = « performance », C_4 = « utilisabilité », C_5 = « évolutivité », C_6 = « documentation »)

Critères	C_1	C_2	C_3	C_4	C_5	C_6	Vecteurs propres
C_1	0,33	0,36	0,23	0,36	0,24	0,34	0,31
C_2	0,11	0,12	0,17	0,12	0,19	0,11	0,14
C_3	0,05	0,02	0,03	0,02	0,01	0,05	0,03
C_4	0,11	0,12	0,17	0,12	0,19	0,11	0,14
C_5	0,07	0,03	0,17	0,03	0,05	0,05	0,06
C_6	0,33	0,36	0,23	0,36	0,33	0,34	0,32

De manière générale, on peut dire que le critère le plus important est la « documentation » et le moins important est la « performance ». Les critères « style » et « utilisabilité » ont la même importance, la séquence complète est présentée ci-dessous.

- DOCUMENTATION > COMMUNAUTÉ > STYLE = UTILISABILITÉ > ÉVOLUTIVITÉ > PERFORMANCE

Après avoir trouvé le poids de chaque critère, on applique les équations mathématiques 5.3, 5.4 et 5.5, et on détermine l'indice de cohérence.

$$\lambda_1 = \frac{(1 \times 0,31) + (3 \times 0,14) + (7 \times 0,03) + (3 \times 0,14) + (5 \times 0,06) + (1 \times 0,32)}{0,31}$$
$$\lambda_1 = 6,45 \tag{5.3}$$

$$\lambda_{max} = \frac{(6,45 + 6,59 + 6,09 + 6,59 + 6,12 + 6,53)}{6}$$
$$\lambda_{max} = 6,40 \tag{5.4}$$

$$IC = \frac{6,40 - 6}{6 - 1}$$
$$IC = 0,08 \tag{5.5}$$

En utilisant la table d'indices de cohérence aléatoire 4.4 nous obtenons la cohérence aléatoire du nombre de critères de notre modèle, une valeur égale à 1.24. Le ratio de cohérence de l'évaluation est égal à $RC = 0,06$, puisqu'il doit être inférieur à $0,10$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des critères est cohérent. L'équation 5.6 illustre le calcul du ratio de cohérence.

$$RC = \frac{0,08}{1,24}$$
$$RC = 0,06 \tag{5.6}$$

5.3 ÉVALUATION DES ALTERNATIVES

Dans cette section, le décideur effectue une comparaison par paires des alternatives, les trois frameworks Node.js, Nest.js, Fastify et Hapi, au sein de chaque critère du modèle, puis établit le poids de chaque alternative au sein de chaque critère et vérifie le ratio de cohérence. Pour chaque critère établi dans notre modèle, le décideur doit faire 3 comparaisons entre les alternatives, sur un total de 18 comparaisons pour cette étape.

5.3.1 COMMUNAUTÉ

On obtient les valeurs des paramètres définis dans la sous-section 4.3.1, on présente donc les valeurs dans le tableau 5.3 ci-dessous. La taille de la communauté Nest.js se démarque avec le plus grand nombre d'étoiles, le nombre de dépendants et de suiveurs sur la « Tag » dans la plateforme Stack Overflow, il faut donc trouver un support pour les problèmes et les bibliothèques à utiliser avec le framework s'avère être une tâche facile. Le montant de téléchargement hebdomadaire de Nest.js est de plus de 1,2 million, soit trois fois plus que Fastify, en somme, le framework a une grande utilité. Fastify a la moitié des étoiles de Nest.js, et pour les autres paramètres, il présente des chiffres bien en dessous par rapport au framework Nest.js. Hapi ne fait que surmonter Fastify qu'en nombre de suiveurs sur le Stack Overflow.

TABLEAU 5.3 : L'évaluation les critères de Communauté

Communauté			
Paramètres	Nest.js	Fastify	Hapi
étoile	43310	22138	13642
dépendant	4576	1906	936
NPM téléchargement	1213755	410856	106860
stackoverflow	3545	176	558

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons devant le critère « communauté ».

1. Le framework Nest.js a une **importance forte** que le framework Fastify
2. Le framework Nest.js a une **importance très grande** que le framework Hapi
3. Le framework Fastify a une **faible importance** que le framework Hapi

Les résultats des comparaisons entre les frameworks sont représentés dans un tableau, on peut le voir dans le tableau 5.4.

TABLEAU 5.4 : Matrice de comparaison par paires entre frameworks pour le critère « communauté »

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	5	7
Fastify	$\frac{1}{5}$	1	3
Hapi	$\frac{1}{7}$	$\frac{1}{3}$	1

Comme le montre le tableau 5.4, le modèle de comparaison est utilisé pour évaluer les frameworks dans le critère « communauté » du modèle, avec leurs paramètres respectifs de popularité et de taille de la communauté. Ensuite, nous présentons le tableau 5.5 avec les résultats de normalisation des valeurs et conjointement avec le vecteur propre maximum.

TABLEAU 5.5 : Matrice de comparaison par paires normalisée entre les frameworks pour le critère de « communauté » et du vecteur propre maximum

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,74	0,79	0,64	0,72
Fastify	0,15	0,16	0,27	0,19
Hapi	0,11	0,05	0,09	0,08

L'indice de cohérence est égal à $IC = 0,03$, le ratio de cohérence de l'évaluation est égal à $RC = 0,06$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères de « communauté » est cohérent.

5.3.2 STYLE

Les frameworks diffèrent dans leur style, car les termes couramment utilisés sont « opiniâtre » ou « non opiniâtre » [50, 15]. Hapi et Fastify ont également une syntaxe plus propre pour écrire du code, ce qui en fait des frameworks « non opiniâtre ». Dans le cas de Nest.js, il s'agit d'un framework « opiniâtre » car il définit comment structurer le code d'application pour fournir le meilleur support pour les abstractions des fonctionnalités.

Tous les frameworks fournissent plusieurs fonctionnalités pour documenter l'application, développer des tests, faire des abstractions avec la base de données et la validation des données. Cependant, les frameworks Hapi et Fastify sont établis comme des frameworks d'aspect minimal, avec peu de fonctionnalités incluses par défaut [79, 74], les fonctionnalités doivent être installées et configurées.

Les frameworks Fastify [74] et Nest.js [80] fournissent un CLI. Le CLI de Fastify aide uniquement à la génération de projet pour démarrer le développement. Cependant, avec le CLI de Nest.js, nous pouvons générer du code d'un Créer, Lire, Mettre à jour et Supprimer (CRUD) [18] pour une ressource d'application particulière.

La commande *« nest g resource »* génère non seulement tous les blocs de construction Nest.js (module, service, classes de contrôleur), mais aussi une classe d'entité, comme les classes Objet de transfert de données (DTO) [19] ainsi que des fichiers de test.

Tous les frameworks fournissent les fonctionnalités nécessaires à l'application, mettant en évidence Nest.js via le CLI. Le Nest.js organise le modèle architectural, en générant du

18. CRUD sont les quatre opérations de base utilisées dans les bases de données relationnelles fournies aux utilisateurs du système.

19. DTO est un modèle de conception de logiciel utilisé pour transférer des données entre les sous-systèmes d'un logiciel.

code et en le modularisant, de sorte que les développeurs doivent créer des applications plus rapidement. Le framework Hapi pour ne pas fournir de CLI ralentit le développement, car tout le code doit être développé.

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons devant le critère « style ».

1. Le framework Nest.js a une **faible importance** que le framework Fastify
2. Le framework Nest.js a une **importance forte** que le framework Hapi
3. Le framework Fastify a une **faible importance** que le framework Hapi

Ces résultats de comparaisons entre les frameworks sont représentés dans un tableau, on peut le voir dans le tableau 5.6.

TABLEAU 5.6 : Matrice de comparaison par paires entre les frameworks pour le critère « style »

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	3	5
Fastify	$\frac{1}{3}$	1	3
Hapi	$\frac{1}{5}$	$\frac{1}{3}$	1

Comme le montre le tableau 5.6, le modèle de comparaison est utilisé pour évaluer les frameworks dans le critère « style » du modèle, avec leurs paramètres respectifs, si le framework fournit des fonctionnalités incluses et s'il détermine comment le code d'application doit être développé. Ensuite, nous présentons le tableau 5.7 avec les résultats de normalisation des valeurs et conjointement avec le vecteur propre maximum.

TABLEAU 5.7 : Matriz normalizada de comparação par a par entre frameworks para o critério « style » e resultado do autovetor máximo
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,65	0,69	0,56	0,63
Fastify	0,22	0,23	0,33	0,26
Hapi	0,13	0,08	0,11	0,11

L'indice de cohérence est égal à $IC = 0,02$, le ratio de cohérence de l'évaluation est égal à $RC = 0,04$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères de « style » est cohérent.

5.3.3 PERFORMANCE

Nous utilisons les données de référence de TechEmpower [20]. Le benchmark est réalisé dans des environnements similaires pour les frameworks avec 128 connexions simultanées utilisées en compétition. Les résultats sont présentés dans le tableau 5.8. Ce benchmark indique clairement que Fastify a un avantage de performance sur Hapi et Nest.js en nombre de requêtes par seconde. Nest.js peut être configuré pour utiliser Express ou Fastify, l'utilisation de Fastify donne un meilleur résultat. Hapi est sérieusement en retard sur Nesj.js et Restify.

20. Disponible sur : https://www.techempower.com/benchmarks

TABLEAU 5.8 : L'évaluation le paramètre de Performance
© Mário Danilo Alves da Silva, 2022

Performance				
Paramètres	Nest.js		Fastify	Hapi
	Express	Fastify		
réponses par seconde	13,0	37,9	41,4	7,5

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons devant le critère « performance ».

1. Le framework Fastify a une **faible importance** que le framework Nest.js
2. Le framework Fastify a une **importance absolue** que le framework Hapi
3. Le framework Nest.js est **importance très grande** que le framework Hapi

Ces résultats de comparaisons entre les frameworks sont représentés dans un tableau, on peut le voir dans le tableau 5.9.

TABLEAU 5.9 : Matriz de comparação par a par entre frameworks para o critério « performance »
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	$\frac{1}{3}$	7
Fastify	3	1	9
Hapi	$\frac{1}{7}$	$\frac{1}{9}$	1

Comme le montre le tableau 5.9, le modèle de comparaison est utilisé pour évaluer les frameworks dans le critère « performance » du modèle, avec son paramètre respectif

de quantité de réquisitions. Puis nous présentons le tableau 5.10 avec les résultats de la normalisation des valeurs et avec le vecteur propre maximum.

TABLEAU 5.10 : Matrice de comparaison par paires normalisée entre les frameworks pour le critère de « performance » et du vecteur propre maximum

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,24	0,23	0,41	0,29
Fastify	0,72	0,69	0,53	0,65
Hapi	0,03	0,08	0,06	0,06

L'indice de cohérence est égal à $IC = 0,04$, le ratio de cohérence de l'évaluation est égal à $RC = 0,08$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères de « performance » est cohérent.

5.3.4 UTILISABILITÉ

Installer et créer une application avec les frameworks Fastify [74], Hapi [79] et Nest.js [80] sont des tâches extrêmement facile à faire. Hapi et Restify, tous deux doivent installer le framework à l'aide d'un gestionnaire de paquets et créer un fichier JavaScript et importer les paquets du framework, les codes 1 et 2 d'exemples, respectivement pour Fastify et Hapi, sont présentés ci-dessous. Quant à Nest.js, il faut d'abord installer Nest CLI à l'aide d'un gestionnaire de paquets, puis il faut exécuter la commande pour créer l'application. En plus de la facilité de création, Nest.js introduit une abstraction et implémente certains fichiers dans le projet, ces exemples sont visibles dans notre dépôt [21].

21. Disponible sur : https://github.com/Mario-UQAC/dataset

Les trois frameworks prennent en charge différents environnements de déploiement, c'est-à-dire, de développement, de test et de production, avec Node.js comme seule exigence. En somme, on peut dire que rendre disponible en production une application développée par l'un des trois frameworks est rapide.

```
 // Require the framework and instantiate it

// ESM
import Fastify from 'fastify'
const fastify = Fastify({
  logger: true
})
// CommonJs
const fastify = require('fastify')({
  logger: true
})

// Declare a route
fastify.get('/', function (request, reply) {
  reply.send({ hello: 'world' })
})

// Run the server!
fastify.listen(3000, function (err, address) {
  if (err) {
    fastify.log.error(err)
    process.exit(1)
  }
  // Server is now listening on \${address}
})
```

Listing 1 – Code Fastify

```
'use strict';

const Hapi = require('@hapi/hapi');

const init = async () => {

    const server = Hapi.server({
        port: 3000,
        host: 'localhost'
    });

    server.route({
        method: 'GET',
        path: '/',
        handler: (request, h) => {

            return 'Hello World!';
        }
    });

    await server.start();
    console.log('Server running on %s', server.info.uri);
};

process.on('unhandledRejection', (err) => {

    console.log(err);
    process.exit(1);
});

init();
```

Listing 2 – Code Hapi

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons devant le critère « utilisabilité ». Dès lors, les comparaisons des frameworks face au critère « utilisabilité » sont d'égale importance les unes par rapport aux autres.

1. Le framework Fastify a une **importance égale** que le framework Nest.js
2. Le framework Fastify a une **importance égale** que le framework Hapi

3. Le framework Nest.js a une **importance égale** que le framework Hapi

TABLEAU 5.11 : Matriz de comparação par a par entre frameworks para o critério « utilisabilité »
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	1	1
Fastify	1	1	1
Hapi	1	1	1

Comme le montre le tableau 5.11, le modèle de comparaison est utilisé pour évaluer les frameworks dans le critère « utilisabilité » du modèle, avec leurs critères respectifs de facilité d'installation et de création d'application, et la rapidité d'avoir une application en production. Puis nous présentons le tableau 5.14 avec les résultats de la normalisation des valeurs et avec le vecteur propre maximum.

TABLEAU 5.12 : Matrice de comparaison normalisée par paires entre les frameworks pour le critère « utilisabilité » et du vecteur propre maximum
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,33	0,33	0,33	0,33
Fastify	0,33	0,33	0,33	0,33
Hapi	0,33	0,33	0,33	0,33

L'indice de cohérence est égal à $IC = 0,00$, le ratio de cohérence d'évaluation est égal à $RC = 0,00$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères « d'utilisabilité » est cohérent.

5.3.5 ÉVOLUTIVITÉ

Nous présentons les fonctionnalités des frameworks Nest.js [80], Fastify [74] et Hapi [79] pour la création d'applications évolutives. Fastify et Hapi adoptent un concept de plugins, tandis que Nest.js adopte un concept de modules. Les plugins et les modules permettent au framework de diviser facilement votre application en éléments isolés de logique métier et en utilitaires réutilisables. Nous pouvons ajouter un plugin ou un module existant à notre application ou créer le nôtre. En plus de passer facilement d'une structure monolithique à des microservices, car chaque plugin ou module peut être un service. Ainsi, pour les frameworks, l'architecture résultante emploiera plusieurs plugins ou modules, chacun encapsulant un ensemble de fonctionnalités étroitement liées.

Nest.js prend en charge nativement le style de développement d'architecture de micro-services, prend en charge plusieurs implémentations de couche de transport intégrées, qui sont responsables de la transmission des messages entre différentes instances de microservices [80], par conséquent, Nest.js a peu d'importance par rapport aux autres.

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons devant le critère « évolutivité ».

1. Le framework Nest.js a une **faible importance** que le framework Fastify
2. Le framework Nest.js a une **faible importance** que le framework Hapi
3. Le framework Fastify a une **importance égale** que le framework Hapi

Ces résultats de comparaisons entre les cadres sont représentés dans un tableau, on peut le voir dans le tableau 5.13.

TABLEAU 5.13 : Matriz de comparação par a par entre frameworks para o critério « évolutivité »
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	3	3
Fastify	$\frac{1}{3}$	1	1
Hapi	$\frac{1}{3}$	1	1

Comme le montre le tableau 5.13, le modèle de comparaison est utilisé pour évaluer les frameworks dans le critère « évolutivité » du modèle, avec son paramètre de disponibilité d'outil respectif pour faire évoluer l'application. Ensuite, nous présentons le tableau 5.14 avec les résultats de normalisation des valeurs et conjointement avec le vecteur propre maximum.

TABLEAU 5.14 : Matrice de comparaison par paires normalisée entre les frameworks pour le critère d'évolutivité et du vecteur propre maximum
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,60	0,60	0,60	0,60
Fastify	0,20	0,20	0,20	0,20
Hapi	0,20	0,20	0,20	0,20

L'indice de cohérence est égal à $IC = 0.00$, le ratio de cohérence de l'évaluation est égal à $RC = 0.00$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères d'évolutivité est cohérent.

5.3.6 DOCUMENTATION

Tous les frameworks fournissent de la documentation. La documentation a une bonne structure, elle propose également des tutoriels directement sur le site pour un guide de démarrage rapide, ainsi que plusieurs références qui permettent de visualiser le code exemple. Ces exemples d'application à titre de référence sont un gros plus, ce qui permet de compenser le manque d'expérience du framework. La documentation du framework Hapi est la seule à ne pas proposer de moteur de recherche, les autres frameworks ont donc peu d'importance par rapport à Hapi pour la qualité de la documentation.

Toutes les alternatives sont comparées entre elles et toutes suivant l'échelle définie dans le tableau 4.2, ci-dessous nous listons les résultats des comparaisons selon le critère « documentation ».

1. Le framework Nest.js a une **importance égale** que le framework Fastify
2. Le framework Nest.js a une **faible importance** que le framework Hapi
3. Le framework Fastify a une **faible importance** que le framework Hapi

Ces résultats des comparaisons entre les frameworks sont représentés sous forme de tableau, comme on le voit dans le tableau 5.15.

TABLEAU 5.15 : Matrice de comparaison par paires entre frameworks pour le critère « documentation »
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi
Nest.js	1	1	3
Fastify	1	1	3
Hapi	$\frac{1}{3}$	$\frac{1}{3}$	1

Comme montré dans le tableau 5.15, le modèle de comparaison est utilisé pour évaluer les frameworks, pour le critère « documentation » du modèle, avec leurs paramètres respectifs, si le framework est documenté, et le niveau de qualité de la documentation. Ensuite, nous présentons le tableau 5.16 avec les résultats de la normalisation des valeurs et avec le vecteur propre maximum.

TABLEAU 5.16 : Matrice de comparaison par paires normalisée entre les frameworks pour le critère de « documentation » et du vecteur propre maximal
© Mário Danilo Alves da Silva, 2022

Frameworks	Nest.js	Fastify	Hapi	Vecteurs propres
Nest.js	0,43	0,43	0,43	0,43
Fastify	0,43	0,43	0,43	0,43
Hapi	0,14	0,14	0,14	0,14

L'indice de cohérence est égal à $IC = 0,00$, le ratio de cohérence de l'évaluation est égal à $RC = 0,00$, donc la matrice de comparaison choisie arbitrairement a du sens et le jugement des frameworks pour les critères de « documentation » est cohérent.

5.4 RÉSULTATS DE L'APPLICATION DU MODÈLE

Dans cette section, nous présentons les résultats consolidés de la section 5.3 basée sur la structure hiérarchique de notre modèle décisionnel multicritère. Ces résultats ont été obtenus à l'aide de notre modèle défini dans le chapitre 4, qui utilise les critères « communauté », « style », « performance », « utilisabilité », « évolutivité » et « documentation ».

À partir de la consolidation des données des vecteurs propres obtenus par la comparaison par paire de chaque framework sous chaque critère et le vecteur propre des critères, nous créons le tableau 5.17, de cette façon nous avons une vue complète de l'évaluation du scénario.

TABLEAU 5.17 : Résultat des critères vecteurs propres et frameworks Node.js

Critères		Frameworks		
	poids	Nest.js	Fastify	Hapi
communauté	0,31	0,72	0,19	0,08
style	0,14	0,63	0,26	0,11
performance	0,03	0,29	0,65	0,06
Utilisabilité	0,14	0,33	0,33	0,33
évolutivité	0,06	0,60	0,20	0,20
documentation	0,32	0,43	0,43	0,14

Afin d'établir la valorisation globale des priorités des frameworks et d'avoir les priorités des alternatives, on calcule la somme des poids des alternatives multipliée par les poids des critères, exemplifiés dans l'équation 5.7. Ces résultats d'évaluation globale entre les frameworks sont représentés dans un tableau, comme on le voit dans le tableau 5.18.

$$
\begin{aligned}
P_1 &= (0,72 \times 0,31) + (0,63 \times 0,14) + (0,29 \times 0,03) + (0,33 \times 0,14) \\
&\quad + (0,60 \times 0,06) + (0,43 \times 0,32) \\
P_1 &= 0,54
\end{aligned}
\tag{5.7}
$$

TABLEAU 5.18 : Résultat de la valorisation globale de Framework Node.js
© Mário Danilo Alves da Silva, 2022

Frameworks	Valorisation globale
Nest.js	0,54
Fastify	0,31
Hapi	0,14

Comme montré dans le tableau 5.18, l'objectif de construction du modèle est atteint, nous pouvons sélectionner le framework qui convient le mieux à l'application, du point de vue du décideur, le framework Nest.js répond plus intensément à l'analyse des préférences. Le framework Nest.js se démarque non seulement dans le critère « communauté », car il est le plus populaire et a la plus grande communauté, mais aussi dans le critère « style » avec plus de fonctionnalités incluses et la rapidité de génération de code à partir d'un CRUD pour une ressource donnée, ainsi que le critère « évolutivité » car il supporte nativement le style de développement de l'architecture microservices.

Nous avons un critère, « utilisabilité », selon lequel les frameworks ont une importance identique. En somme, la facilité d'installation et de création des applications, ainsi que la rapidité de leur mise à disposition en production sont favorables à tous les frameworks. Fastify

excelle dans le critère « performance » par rapport à Hapi et Nest.js pour les requêtes par seconde, même si Nest.js est configuré pour utiliser Express ou Fastify.

En plus d'analyser l'objectif final de notre modèle multicritère, nous observons que parmi les critères retenus du modèle, dans ce cas d'usage, le critère « documentation » a reçu le score le plus élevé, suivi du critère « communauté », et le moins important dans ce cas d'utilisation est un critère de « performance ».

Le framework Nest.js obtient la valorisation globale la plus élevée dans l'évaluation de notre modèle. Nest.js a une valeur totale de 0,54, qui peut être présentée comme 54% d'importance, suivi de Fastify avec 31% et Hapi avec 14%. Ainsi, notre modèle décisionnel multicritère pour la sélection du framework Node.js montre sa capacité à aider le développeur ou l'équipe projet lorsqu'il s'agit de sélectionner un framework parmi les différents frameworks.

CONCLUSION

L'analyse de la littérature grise écrite sur les blogs technologiques a permis de collecter des données sur l'industrie et la communauté, ainsi que de comprendre les caractéristiques de qualité et de préférence des frameworks. De plus, cela nous a permis de comprendre les critères utilisés pour sélectionner un framework Node.js pour un nouveau projet d'application, afin de créer un modèle décisionnel multicritères qui aide l'équipe de développement à déterminer quel framework est le mieux adapté à l'application.

Nous avons identifié un ensemble de critères, ainsi que des paramètres liés à l'évaluation et à la sélection d'un framework. Ces critères sont les plus utilisés par les développeurs et les entreprises dans la sélection des frameworks dans les projets logiciels. De plus, nous avons créé une cartographie des frameworks les plus recommandés pour la technologie Node.js.

Notre modèle décisionnel multicritères pour la sélection des frameworks Node.js est organisé en critères de sélection et en paramètres d'évaluation. Notre modèle présente une application d'une méthodologie basée sur la technique de Prise de décision multicritères (MCDM), il s'agit de la méthode multicritère de Processus d'analyse hiérarchique (AHP). De cette façon, notre modèle montre sa capacité à aider le développeur ou l'équipe projet dans le choix d'un framework qui correspond le mieux aux besoins de l'application, parmi les différents frameworks.

Ce travail met en lumière les bases de recherches futures. Des études pourraient être menées pour utiliser ou développer des sous-critères pour chacun des critères sélectionnés dans la section 4.2. Les travaux futurs incluent la validation des critères et des paramètres dans les environnements d'entreprise, augmentant l'efficacité des décisions. De plus, nous devons envisager d'étendre l'application de notre modèle décisionnel multicritère aux frameworks qui utilisent une autre technologie que Node.js, afin de soutenir la sélection d'un framework.

BIBLIOGRAPHIE

[1] H. Strydom, "Web development statistics," 2021. [En ligne]. Repéré à : https://strydomwebdevelopment.co.za/web-development-statistics/

[2] D. Curie, J. Jaison, J. Yadav, et J. Fiona, "Analysis on web frameworks," *Journal of Physics : Conference Series*, vol. 1362, p. 012114, 2019.

[3] S. Bangare, S. Gupta, M. Dalal, et A. Inamdar, "Using node.js to build high speed and scalable backend database server," *International Journal of Research in Advent Technology (E-ISSN : 2321-9637)*, vol. 4, p. 19, 2016.

[4] K. Peguero, N. Zhang, et X. Cheng, "An empirical study of the framework impact on the security of javascript web applications," dans *Companion Proceedings of the The Web Conference 2018*, ser. WWW '18. Republic and Canton of Geneva, CHE : International World Wide Web Conferences Steering Committee, 2018, p. 753–758. [En ligne]. Repéré à : https://doi.org/10.1145/3184558.3188736

[5] D. Ritter, "Why use a framework," 2021. [En ligne]. Repéré à : https://symfony.com/why-use-a-framework

[6] D. Roberts et R. Johnson, *Evolving frameworks : A pattern language for developing object-oriented frameworks*. Addison-Wesley, 1996.

[7] S. Overflow, "Stack overflow annual developer survey 2021," 2021. [En ligne]. Repéré à : https://insights.stackoverflow.com/survey/2021

[8] ——, "About stars," 2020. [En ligne]. Repéré à : https://insights.stackoverflow.com/survey/2020#technology-other-frameworks-libraries-and-tools-all-respondents3

[9] N. Foundation, "2018 node.js user survey report," 2018. [En ligne]. Repéré à : https://nodejs.org/en/user-survey-report/

[10] J. Patel, "Which is the best nodejs framework to use in web development ?" 2021. [En ligne]. Repéré à : https://www.monocubed.com/best-nodejs-frameworks/

[11] L. Shipton, “The best nodejs frameworks for 2021,” 2021. [En ligne]. Repéré à : https://rapidapi.com/blog/best-nodejs-frameworks/?utm_source=google&utm_medium=cpc&utm_campaign=DSA&gclid=Cj0KCQiAw9qOBhC-ARIsAG-rdn7l34w-4Qj4XcPH3GhXNq6eh34j8175sJQPCYh5N0-L98ukPjnEz2saAwcB

[12] J. Verville et A. Halingten, “An investigation of the decision process for selecting an erp software : The case of esc,” *Management Decision - MANAGE DECISION*, vol. 40, pp. 206–216, 2002.

[13] A. Gizas, S. Christodoulou, et T. Papatheodorou, “Comparative evaluation of javascript frameworks,” *WWW’12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, 2012.

[14] SPECIndia, “Top 15 best node js frameworks soaring in popularity,” 2021. [En ligne]. Repéré à : https://www.spec-india.com/blog/node-js-framework

[15] MDN, “Server-side web frameworks,” 2021. [En ligne]. Repéré à : https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks

[16] V. Garousi, M. Felderer, et M. Mäntylä, “Guidelines for including the grey literature and conducting multivocal literature reviews in software engineering,” *Information and Software Technology*, 2017.

[17] V. Garousi, M. Felderer, M. Mäntylä, et A. Rainer, “Benefitting from the grey literature in software engineering research,” dans *Contemporary Empirical Methods in Software Engineering*, 2019.

[18] A. Da Silva, E. Guedes, J. R. Ribeiro, L. Nascimento, M. Belderrain, et A. Correia, “Multiple criteria methods applied to select suppliers of a capital goods company,” dans *9th International Symposium on the Analytic Hierarchy Process*, ser. ISAHP 2007. Viña Del Mar, Chile : Proceedings of the 9th ISAHP, 2007, p. 10.

[19] C. Tobolski, “Restify vs. sails vs. hapi : Node.js framework comparison,” 2019. [En ligne]. Repéré à : https://stackify.com/restify-vs-sails-vs-hapi-node-js-framework-comparison/

[20] M. Salas-Zárate, G. Alor-Hernández, R. Valencia-García, L. Rodríguez-Mazahua, A. Rodríguez-González, et J. López Cuadrado, “Analyzing best practices on web development frameworks : The lift approach,” *Science of Computer Programming*, 2015.

[21] R. Santelices et M. Nussbaum, “A framework for the development of videogames,” *Softw., Pract. Exper.*, vol. 31, pp. 1091–1107, 2001.

[22] O. Foundation, “Node.js,” 2022. [En ligne]. Repéré à : https://nodejs.org/

[23] V8, “V8 javascript engine,” 2022. [En ligne]. Repéré à : https://v8.dev/

[24] L. Liang, L. Zhu, W. Shang, D. Feng, et Z. Xiao, “Express supervision system based on nodejs and mongodb,” dans *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 2017, pp. 607–612.

[25] J. Wang, W. Dou, Y. Gao, C. Gao, F. Qin, K. Yin, et J. Wei, “A comprehensive study on real world concurrency bugs in node.js,” dans *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 520–531.

[26] S. Maharjan, “Markeplace web application with foglab,” *Helsinki Metropolia University of Applied Sciences. Bachelor Thesis*, 2017.

[27] C. Peters, “Building rich internet applications with node. js and express. js,” *Rich Internet Applications w/HTML and Javascript*, p. 15, 2017.

[28] StackShare, “Node.js - reviews, pros & cons companies using node.js,” 2022. [En ligne]. Repéré à : https://stackshare.io/nodejs

[29] P. Schoemaker et J. Russo, “A pyramid of decision approaches,” *California Management Review*, vol. 36, 1993.

[30] C. A. Bana e Costa, “Processo de apoio à decisão : problemáticas, actores e acções,” *Florianópolis : ENE (Escola de Novos Empreendedores da UFSC), Santa Catarina, Brazil*, 1995.

[31] M. Zeleny, “Compromise programming, multiple criteria decision making, edited by jl cochrane and m. zeleny,” 1973.

[32] J.-P. Brans et B. Mareschal, *Promethee Methods*. New York, NY : Springer New York, 2005, pp. 163–186. [En ligne]. Repéré à : https://doi.org/10.1007/0-387-23081-5_5

[33] M. Sirshar, I. Hanif, et S. Shahzad, “A review paper on decision making using ahp (analytical hierarchy process) techniques in different types of project management,” dans *Preprints*, n° 2019120068, 2019.

[34] M. Zeleny, *Multiple Criteria Decision Making*, ser. McGraw-Hill series in quantitative methods for management. McGraw-Hill, 1982. [En ligne]. Repéré à : https://books.google.ca/books?id=vTayAAAAIAAJ

[35] E. Triantaphyllou, *Multi-Criteria Decision Making Methods : A Comparative Study*. Springer, Boston, MA, 2000, vol. 44.

[36] A. Zaidan, B. Bahaa, M. Hussain, A. Al-Haiqi, M. L. Mat Kiah, et M. Abdulnabi, “Multi-criteria analysis for os-emr software selection problem : A comparative study,” *Decision Support Systems*, vol. 78, 2015.

[37] M. Hanine, O. Boutkhoum, A. Tikniouine, et T. Agouti, “Application of an integrated multi-criteria decision making ahp-topsis methodology for etl software selection,” *SpringerPlus*, vol. 5, 2016.

[38] S. Muthuraman, S. Kamesh, et G. Patra, “Comparison of ahp based and fuzzy based mechanisms for ranking cloud computing services,” dans *2015 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 2015, pp. 175–180.

[39] B. Abdullateef, N. F. Elias, H. Mohamed, A. Zaidan, et B. Bahaa, “An evaluation and selection problems of oss-lms packages,” *SpringerPlus*, vol. 5, 2016.

[40] K. W. Collier, D. Sautter, C. Marjaniemi, et B. Carey, “A methodology for evaluating and selecting data mining software,” *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, vol. Track6, pp. 11 pp.–, 1999.

[41] A. S. Jadhav et R. M. Sonar, “Evaluating and selecting software packages : A review,” *Inf. Softw. Technol.*, vol. 51, pp. 555–563, 2009.

[42] T. Kaneriya, “12 best nodejs frameworks for web apps in 2022,” 2021. [En ligne]. Repéré à : https://www.simform.com/blog/best-nodejs-frameworks/

[43] B. A. Kitchenham, T. Dyba, et M. Jorgensen, “Evidence-based software engineering,” dans *Proceedings of the 26th International Conference on Software Engineering*, ser. ICSE ’04. USA : IEEE Computer Society, 2004, p. 273–281.

[44] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, et S. Linkman, “Systematic literature reviews in software engineering-a systematic literature review,” *Information and Software Technology*, vol. 51, pp. 7–15, 2009.

[45] T. Shan et W. Hua, “Taxonomy of java web application frameworks,” dans *2006 IEEE International Conference on e-Business Engineering (ICEBE’06)*, 2006, pp. 378–385.

[46] N. Kharchenko, “How to choose a perfect framework for node.js,” 2017. [En ligne]. Repéré à : https://www.codementor.io/@natalia/how-to-choose-a-perfect-framework-for-node-js-crpk9jygr

[47] P. Wadhwani, “Top 10 node.js frameworks – for ctos to choose right node web framework,” 2021. [En ligne]. Repéré à : https://www.bacancytechnology.com/blog/top-nodejs-frameworks

[48] M. Demchenko, “The best nodejs frameworks,” 2020. [En ligne]. Repéré à : https://ncube.com/blog/the-best-nodejs-frameworks

[49] C. Gor, “Top 25 node.js api frameworks for the year 2022,” 2021. [En ligne]. Repéré à : https://www.esparkinfo.com/node-js-frameworks.html

[50] S. Plenderleith, “Guidelines for choosing a node.js framework,” 2021. [En ligne]. Repéré à : https://simonplend.com/guidelines-for-choosing-a-node-js-framework/

[51] B. Griggs, “Introduction to the node.js reference architecture, part 6 : Choosing web

frameworks," 2021. [En ligne]. Repéré à : https://developers.redhat.com/articles/2021/12/03/introduction-nodejs-reference-architecture-part-6-choosing-web-frameworks#

[52] L. Parody, "Choosing the right node.js framework : Express, koa, or hapi ?" 2019. [En ligne]. Repéré à : https://nodesource.com/blog/Express-Koa-Hapi

[53] A. Kucher, "5 best node.js frameworks," 2021. [En ligne]. Repéré à : https://dev.to/alex_kucher/5-best-node-js-frameworks-2i68

[54] Technostacks, "Top node.js frameworks for web apps in 2022," 2021. [En ligne]. Repéré à : https://technostacks.com/blog/nodejs-frameworks/

[55] S. Lypchenko, "How to choose the best node.js framework : Express.js, koa.js or sails.js," 2019. [En ligne]. Repéré à : https://ourcodeworld.com/articles/read/364/how-to-choose-the-best-nodejs-framework-expressjs-koajs-or-sailsjs

[56] Matellio, "How to choose the best node.js framework ?" 2020. [En ligne]. Repéré à : https://www.matellio.com/blog/how-to-choose-the-best-node-js-framework/

[57] K. Santra, "10 best node.js framework in 2021," 2021. [En ligne]. Repéré à : https://medium.com/dhiwise/10-best-node-js-framework-in-2021-76b6647cf

[58] TypeScript, "Typescript : Javascript with syntax for types." 2022. [En ligne]. Repéré à : https://www.typescriptlang.org/

[59] S. K. Arora, "10 best nodejs frameworks for developers," 2021. [En ligne]. Repéré à : https://hackr.io/blog/nodejs-frameworks

[60] N. Bevan, "Quality in use : Meeting user needs for quality," *Journal of Systems and Software*, vol. 49, pp. 89–96, 1999.

[61] M. Kiah, A. Al-Haiqi, B. Bahaa, et A. Zaidan, "Open source emr software : Profiling, insights and hands-on analysis," *Computer Methods and Programs in Biomedicine*, vol. 117, 2014.

[62] T. Arh et B. Jerman, “A multi-attribute decision support model for learning management systems evaluation,” dans *First International Conference on the Digital Society (ICDS’07)*, 2007, p. 11.

[63] T. L. Saaty, *The analytic hierarchy process*. New York, NY : McGraw-Hill, Makron, 1980.

[64] ——, *Método de Análise Hierárquica*. São Paulo, SP : McGraw-Hill, Makron, 1991.

[65] ——, “How to make a decision : The analytic hierarchy process,” *European Journal of Operational Research*, vol. 48, n° 1, pp. 9–26, 1990, desicion making by the analytic hierarchy process : Theory and applications. [En ligne]. Repéré à : https://www.sciencedirect.com/science/article/pii/037722179090057I

[66] L. Gomes, M. González, et C. Carignano, *Tomada de decisões em cenários complexos : introdução aos métodos discretos do apoio multicritério à decisão*. Thomson, 2004.

[67] J. Yang et H. Lee, “An ahp decision model for facility location selection,” *Facilities*, vol. 15, pp. 241–254, 1997.

[68] L. G. Vargas, “An overview of the analytic hierarchy process and its applications,” *European Journal of Operational Research*, vol. 48, pp. 2–8, 1990.

[69] D. H. Hartman et M. N. Goltz, “Application of the analytic hierarchy process to select characterization and risk-based decision-making and management methods for hazardous waste sites,” *Environmental Engineering and Policy*, vol. 3, pp. 1–7, 2001.

[70] G. A. Miller, “The magical number seven, plus or minus two : Some limits on our capacity for processing information.” *Psychological review*, vol. 63, n° 2, p. 81, 1956.

[71] T. L. Saaty et M. S. Ozdemir, “Why the magic number seven plus or minus two,” *Mathematical and computer modelling*, vol. 38, n° 3-4, pp. 233–244, 2003.

[72] M. Fayad et D. Schmidt, “Object-oriented application frameworks,” *Communications of the ACM*, vol. 40, 1997.

[73] S. Matam et J. Jain, *Pro Apache JMeter : Web Application Performance Testing*. Apress, Berkeley, CA, 2017.

[74] Fastify, “Fastify - documentation,” 2022. [En ligne]. Repéré à : https://www.fastify.io/docs/latest

[75] A. Gupta, R. Christie, et P. Manjula, “Scalability in internet of things : features, techniques and research challenges,” *Int. J. Comput. Intell. Res*, vol. 13, n° 7, pp. 1617–1627, 2017.

[76] M. F. Arlitt, D. Krishnamurthy, et J. A. Rolia, “Characterizing the scalability of a large web-based shopping system,” *ACM Trans. Internet Technol.*, vol. 1, n° 1, p. 44–69, 2001. [En ligne]. Repéré à : https://doi.org/10.1145/383034.383036

[77] W. Pree et H. Sikora, “Design patterns for object-oriented software development (tutorial),” dans *Proceedings of the 19th International Conference on Software Engineering*, ser. ICSE ’97. New York, NY, USA : Association for Computing Machinery, 1997, p. 663–664. [En ligne]. Repéré à : https://doi.org/10.1145/253228.253810

[78] L. Gomes, C. Gomes, et A. de Almeida, *Tomada de decisão gerencial : enfoque multicritério*. Atlas, 2009.

[79] Hapi, “Hapi - documentation,” 2022. [En ligne]. Repéré à : https://hapi.dev/tutorials

[80] NestJS, “Nest.js - documentation,” 2022. [En ligne]. Repéré à : https://docs.nestjs.com/

APPENDICE A

LISTE DES FRAMEWORKS NODE.JS

TABLEAU A.1 : Liste des frameworks Node.js plus cités dans les blogs avec les sources

Frameworks	Sources																	Total
	[46]	[55]	[57]	[42]	[59]	[53]	[19]	[51]	[56]	[10]	[54]	[52]	[47]	[48]	[11]	[14]	[49]	[15]	
Express	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	17
Sails	✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	15
Koa	✓	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓		✓	✓		14
Hapi				✓	✓	✓	✓	✓		✓	✓	✓	✓			✓	✓		12
Meteor			✓	✓	✓	✓			✓	✓	✓		✓		✓	✓	✓		12
LoopBack			✓	✓	✓			✓		✓			✓	✓	✓	✓	✓	✓	11
Nest.js			✓	✓	✓	✓		✓		✓		✓	✓	✓		✓	✓		11
Adonis			✓	✓	✓			✓		✓				✓		✓	✓		8
Total.js				✓	✓					✓	✓		✓			✓	✓		7
Feathers				✓				✓			✓		✓			✓	✓	✓	7
Socket.io			✓	✓							✓		✓			✓	✓		6
Derby.js				✓	✓					✓						✓	✓		5
Restify.js			✓				✓	✓									✓		4
Mean.js																✓	✓	✓	3
Keystone.js																✓	✓	✓	3
Kraken.js									✓									✓	2
Strapi																✓	✓		2
Next.js								✓				✓							2
Nuxt								✓				✓							2
Mocha.js			✓														✓		2
Fastify								✓											1
Mojito																	✓		1
Electron.js									✓										1
Diet.js																	✓		1
ActionHero.Js																	✓		1
Sequelize																	✓		1
Molecular																	✓		1
Flatiron.Js																	✓		1
FoalTS																	✓		1
VulcanJS																	✓		1