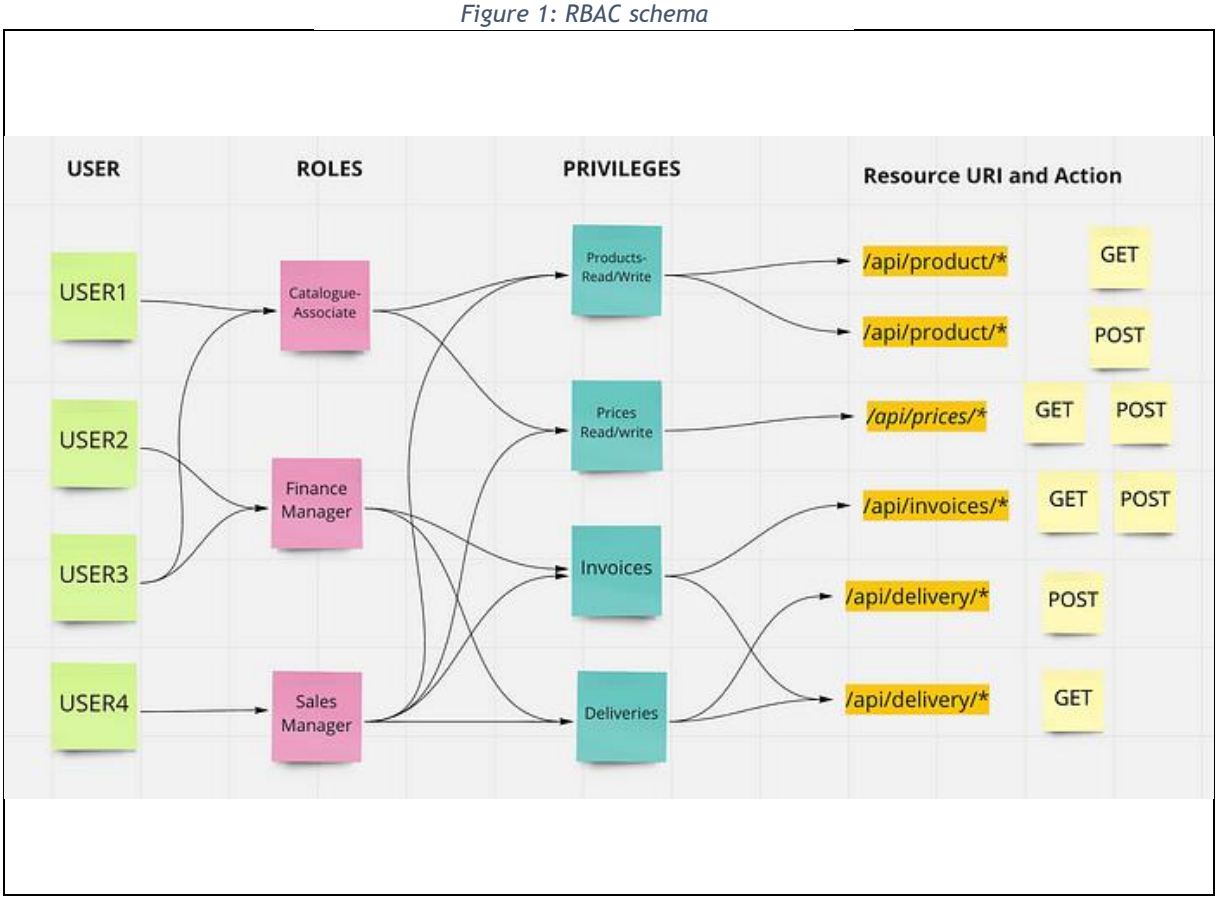
Team Academy - Article réflexif

Comment implémenter un contrôle d'accès basé sur le rôle dans un projet d'application web ?

Figure 1: RBAC schema



Etudiant : Dasek Joiakim

Professeur : Jean-Luc Beuchat

Table des matières

Introduction

Cette année, l'un des projets les plus marquants a été le développement de Koloka, une plateforme innovante conçue pour mettre en relation de futurs colocataires en fonction de leurs affinités personnelles. Ce projet collaboratif entre la Digital Team Academy et la Business Team Academy a été lancé il y a trois semestres. Dès le début, j'étais conscient de l'ampleur technique de ce projet et de son importance stratégique. Toutefois, une problématique particulière s'est rapidement dévoilée : comment implémenter un contrôle d'accès basé sur les rôles de manière efficace et sécurisée ?

Alors que nous progressions dans le développement de Koloka, il est devenu évident que la sécurité et la gestion des accès étaient des aspects critiques à maîtriser. Les tests utilisateurs ont révélé des vulnérabilités potentiellement graves, notamment des accès non autorisés à des conversations privées et à des informations de profil. Confronté à ces défis, j'ai ressenti une frustration initiale, suivie d'une motivation croissante pour résoudre ces problèmes et renforcer mes compétences en sécurité logicielle.

Pour structurer cette réflexion, j'ai appliqué le modèle d'apprentissage expérientiel de Kolb. Ce modèle, en quatre phases – expérience concrète, réflexion sur l'expérience, conceptualisation abstraite et expérimentation active – m'a servi de guide pour analyser cette problématique complexe. En partant de mon expérience vécue lors du projet Koloka, j'ai réfléchi aux défis rencontrés, exploré des solutions théoriques et élaboré un plan d'action pour améliorer la gestion des accès dans des projets futurs.

Cet article s'articule autour de la problématique suivante : comment implémenter un contrôle d’accès basé sur le rôle dans un projet d’application web ? À travers cette réflexion, je vise à développer une compréhension plus approfondie des mécanismes de gestion des

accès et de la sécurité des données, tout en proposant des stratégies concrètes pour renforcer ces aspects essentiels dans le cadre de projets de développement web.

Expérience Concrète

Pour contextualiser la problématique, il est essentiel de présenter le projet Koloka afin de permettre une meilleure compréhension de la situation et des solutions recherchées. Koloka est un projet collaboratif entre la Digital Team Academy et la Business Team Academy, inauguré il y a trois semestres. Cette plateforme innovante vise à mettre en relation de futurs colocataires en fonction de leurs personnalités, grâce à un système sophistiqué de matching. Les utilisateurs peuvent, après une mise en relation réussie, engager des conversations directement sur la plateforme, favorisant ainsi des connexions basées sur des affinités réelles.

Dès le départ, j'étais conscient de l'ampleur technique et de l'importance stratégique du projet. Cependant, en tant qu'étudiant en informatique de gestion, j'ai dû composer avec mes connaissances initiales limitées. Je n'avais pas encore une vision claire des nuances et des exigences complexes inhérentes à la gestion des accès et des permissions granulaires, essentielles pour garantir la sécurité et la confidentialité des données sur une plateforme aussi interactive.

La problématique du contrôle d'accès s'est manifestée au cours du deuxième tiers du projet. Avant cela, ma priorité était de développer les fonctionnalités de base et d'assurer un niveau de performance optimal. Cependant, en réalisant des tests utilisateurs et en procédant à des revues de code, j'ai rapidement identifié que certaines ressources, telles que les conversations privées et les informations de profil, étaient vulnérables. Certaines pages et données étaient potentiellement accessibles par des utilisateurs non autorisés, posant ainsi des risques majeurs de confidentialité et de sécurité.

Pour ajouter davantage de contexte technique, j'ai choisi d'utiliser Strapi comme technologie backend. Strapi est un CMS tête découplée (headless CMS) populaire, qui génère automatiquement une API RESTful ou GraphQL, facilitant ainsi la communication avec le frontend. Ce CMS offre une interface utilisateur conviviale et de puissantes fonctionnalités d'administration de contenu, incluant la gestion des utilisateurs et des rôles.

Bien que Strapi promette une gestion simplifiée des accès basée sur les rôles, j'ai estimé essentiel de comprendre en profondeur son fonctionnement afin de l'adapter à nos besoins spécifiques. J'ai constaté que le système par défaut de Strapi était certes efficace, mais pour assurer une flexibilité et une personnalisation maximale, il était nécessaire d'aller au-delà des configurations basiques. Ces connaissances me seraient également précieuses pour de futurs projets requérant un contrôle d'accès plus personnalisé et granulaire.

Ainsi, mon objectif est d'explorer davantage des solutions théoriques et pratiques autour du contrôle d'accès basé sur les rôles, non seulement pour améliorer Koloka, mais aussi pour renforcer mes compétences en sécurité logicielle et gestion des autorisations dans un environnement d'applications web.

Observation Réfléchie

En réfléchissant à l'expérience vécue au cours du projet Koloka, une combinaison de sentiments et de réflexions m'a traversé l'esprit.

Premièrement, j'ai ressenti une certaine frustration en découvrant les vulnérabilités potentielles des ressources critiques de la plateforme, telles que les conversations privées et les informations de profil. Même si nous n'avons pas encore de réels utilisateurs, il était évident que si ces problèmes persistaient, ils pourraient poser des risques majeurs de confidentialité et de sécurité une fois la plateforme lancée. Cette prise de conscience a été un choc, car elle révélait des lacunes dans un système que je pensais relativement sécurisé.

D'un autre côté, cette situation a également été une source de motivation. Détecter les faiblesses dans le contrôle d'accès m'a poussé à aller au-delà de ce que j'avais initialement prévu d'apprendre et de maîtriser. Cette prise de conscience a éveillé en moi une envie de comprendre les mécanismes plus avancés de la gestion des accès afin de résoudre nos problèmes actuels mais aussi prévenir de futures vulnérabilités dans mes projets à venir.

En analysant cette expérience, je reconnais que le manque initial de vision claire sur les exigences complexes de la gestion des accès était en partie dû à une focalisation excessive sur les fonctionnalités de base et la performance. Cette erreur d'appréciation est courante lorsqu'on est sous pression pour livrer un produit fonctionnel rapidement. Cependant, elle m'a enseigné l'importance de prévoir une architecture de sécurité dès les premières phases de développement !

Il est également devenu apparent que, bien que Strapi offre une solution intégrée pour la gestion des rôles, son utilisation méritait une étude plus approfondie pour l'adapter correctement à nos besoins spécifiques de sécurité. Cela m'a amené à réfléchir plus largement sur les outils et les pratiques que je pourrais utiliser ou développer à l'avenir pour renforcer la sécurité des applications web.

Finalement, cette expérience m'a mis face à mes propres limites et zones d'amélioration. Elle m'a aussi offert une opportunité précieuse de croissance en approfondissant mes connaissances et mes compétences techniques tout en m'inculquant une rigueur nouvelle en matière de sécurité et de gestion d'autorisation dans le développement web. C'est avec un sentiment enrichi de responsabilité et de vigilance que je me suis résolu à aborder, à l'avenir, toute implémentation de contrôle d'accès et de sécurité des données.

Conceptualisation Abstraite

Pour résoudre les défis liés au contrôle d'accès identifiés dans le projet Koloka, plusieurs approches théoriques et pratiques peuvent être envisagées. Ces approches visent non seulement à assurer la sécurité et la confidentialité des données, mais aussi à renforcer l'efficacité et la flexibilité du système de gestion d'accès. Parmi ces solutions, le contrôle d'accès basé sur les rôles (RBAC) s'est révélé être particulièrement prometteur et pertinent pour notre contexte. Mais pour une analyse exhaustive, explorons d'abord les différentes options de gestion d'accès disponibles :

1. Contrôle d'Accès Discrétionnaire (DAC)

Le contrôle d'accès discrétionnaire, ou DAC pour « Discretionary Access Control », est une méthode où les propriétaires de ressources ont la possibilité de déterminer les permissions d'accès. Ils peuvent accorder ou révoquer les droits d'accès à d'autres utilisateurs. Cette méthode offre une grande flexibilité, mais elle peut devenir difficile à gérer dans des environnements à grande échelle ou hautement sécurisés. En outre, DAC est souvent plus sujet aux erreurs humaines, car il repose sur des décisions manuelles pour chaque ressource.

2. Contrôle d'Accès Obligatoire (MAC)

Le contrôle d'accès obligatoire, ou MAC pour « Mandatory Access Control », est une approche de sécurité plus rigide où les accès sont définis par des politiques strictes et centralisées, souvent basées sur des niveaux de sécurité. Ce système est principalement utilisé dans des environnements où la sécurité est primordiale, comme les institutions militaires et gouvernementales. Bien que très sécurisé, le MAC perd en flexibilité et en facilité de gestion, ce qui peut être inadapté pour des projets collaboratifs comme Koloka.

3. Contrôle d'Accès Basé sur les Rôles (RBAC)

Après une analyse approfondie, le Contrôle d'Accès Basé sur les Rôles (RBAC) s'est avéré être la solution la plus adaptée pour Koloka. Le RBAC permet de simplifier la gestion des permissions et de renforcer la sécurité tout en offrant une flexibilité raisonnable.

Concepts clés de RBAC

1. **Rôles** : Les rôles sont des ensembles de permissions attribués aux utilisateurs en fonction de leur fonction ou tâche spécifique au sein du système. Par exemple, dans Koloka, nous pourrions avoir des rôles tels que « Utilisateur », « Modérateur », et « Administrateur ».
2. **Permissions** : Une permission est une autorisation d'effectuer une opération spécifique sur une ressource. Par exemple, consulter des conversations, modifier des profils, etc.
3. **Utilisateurs** : Les utilisateurs finaux accèdent aux ressources via les rôles qui leur sont attribués. Chaque utilisateur peut avoir un ou plusieurs rôles.
4. **Sessions** : Les sessions permettent de gérer les rôles actifs d'un utilisateur au cours d'une interaction spécifique avec le système.
5. **Hiérarchies de Rôle** : Les rôles peuvent être structurés de manière hiérarchique, où des rôles « enfants » héritent des permissions des rôles « parents ». Cela permet une administration plus simple et une mise à jour centralisée des permissions.

Principes de RBAC

1. **Principle of least privilege (principe de moindre privilège)** : Chaque utilisateur doit recevoir les permissions minimales nécessaires pour accomplir ses tâches. Ceci réduit les risques de violations de sécurité.
2. **Separation of duties (séparation des tâches)** : Cette stratégie divise les responsabilités de façon à réduire les risques de fraude ou d'abus. Par exemple, une même personne ne devrait pas initier et approuver un changement critique.

3. **Role-based assignment** : Les utilisateurs acquièrent des permissions en se voyant attribuer des rôles, et non directement des permissions. Cela aide à une gestion plus structurée et moins erronée.
4. **Role-based review** : Les rôles et permissions attribués doivent être régulièrement revus pour s'assurer qu'ils sont toujours en adéquation avec les besoins actuels du système et des utilisateurs.

Mise œuvre théorique

La mise en œuvre du RBAC dans un système tel que Koloka nécessiterait les étapes suivantes :

1. **Définir les rôles** : Identifier et documenter tous les rôles possibles au sein du système en collaboration avec les divers parties prenantes. Dans notre cas, des rôles comme « Utilisateur », « Modérateur », et « Administrateur » pourraient être créés.

2. **Attribution des permissions** : Décrire précisément les permissions associées à chaque rôle. Par exemple, un "Utilisateur" peut visualiser des profils et envoyer des messages, tandis qu'un "Administrateur" peut créer, modifier et supprimer des profils.

3. **Association des utilisateurs aux rôles** : Mappez chaque utilisateur à un ou plusieurs rôles en fonction de leurs responsabilités.

4. **Implémentation technique** : Utiliser les fonctionnalités offertes par Strapi pour intégrer RBAC, en étendant les configurations basiques si nécessaire. La gestion des rôles et des permissions pourrait nécessiter l’ajout de middleware pour vérifier les autorisations de l'utilisateur sur chaque demande HTTP.

5. **Audit et surveillance** : Mettre en place des audits réguliers des rôles et permissions afin de s'assurer qu'aucune permission superflue ou dangereuse ne reste active inutilisée.

6. **Formation** : Informer les développeurs et administrateurs sur le modèle RBAC mis en place, incluant comment inspecter et modifier les autorisations des utilisateurs en cas de nécessité.

L’utilisation d’un middleware

1. **Séparation des responsabilités** : En séparant la logique de vérification des permissions dans un middleware, on maintient une architecture propre et modulaire. Cela permet de décorréler la logique de gestion des accès de la logique fonctionnelle de l'application, facilitant ainsi la maintenance et l'évolutivité du code.

2. **Centralisation des Contrôles d'Accès** : Un middleware centralise toutes les vérifications d'accès en un seul endroit, rendant plus aisées les mises à jour et les audits de sécurité. Par exemple, si une nouvelle règle de permission doit être ajoutée, elle peut l'être dans le middleware sans avoir à modifier chaque route individuellement.

3. **Réutilisabilité** : Le middleware peut être réutilisé à travers différentes routes et parties de l'application, ce qui réduit la redondance du code et assure une application cohérente des politiques de sécurité. Cette réutilisabilité est particulièrement précieuse dans des projets de grande envergure où les mêmes logiques de contrôle d'accès sont nécessaires dans divers contextes.

Processus d'implémentation

1. **Planification et design** :

    - **Cartographier les rôles et permissions** : Collaborer avec les équipes de conception et les développeurs pour identifier les rôles essentiels et les associer aux permissions nécessaires.

- **Configurer la base de données** : Structurer les tables ou collections pour stocker les rôles, permissions, et associations utilisateur-rôle.

2. **Développement du Middleware** :

   - **Vérification des permissions** : Développer des fonctions de middleware qui inspectent les rôles d'un utilisateur et comparent les permissions requises pour accéder aux ressources demandées.

   - **Gestion des erreurs** : Inclure une logique pour gérer les accès non autorisés de manière sécurisée, en fournissant des messages d'erreur clairs et sécurisés sans divulguer d'informations sensibles.

3. **Intégration et tests** :

   - **Intégration avec les routes** : Appliquer le middleware aux routes appropriées de l'application pour contrôler l'accès aux ressources sensibles comme les profils utilisateur et les conversations privées.

   - **Tests unitaire et d'intégration** : Effectuer des tests rigoureux pour s'assurer que le middleware fonctionne correctement et que les politiques de sécurité sont appliquées comme prévu.

4. **Surveillance et audits** :

   - **Surveillance en continu** : Utiliser des outils de surveillance pour suivre les accès et détecter toute activité suspecte ou non autorisée.

   - **Audits périodiques** : Effectuer des audits réguliers des rôles et permissions pour s'assurer que les configurations reflètent toujours les besoins actuels de l'application.

Expérimentation Active

Pour aborder les futures implémentations de contrôle d'accès avec une approche plus structurée et sécurisée, j'ai formulé plusieurs hypothèses et engagements basés sur les apprentissages et les réflexions issus de ma participation au projet Koloka. Ces propositions visent à améliorer non seulement les aspects techniques, mais également les processus de développement et de gestion de la sécurité.

Hypothèse 1 : Définir une architecture de sécurité dès les premières phases du projet

Je m'engage à intégrer la sécurité comme une composante essentielle de la phase de conception de tout projet futur. Au lieu d'aborder la sécurité et la gestion des accès de manière réactive, je proposerai et participerai activement à l'élaboration d'une architecture de sécurité avant même le début du développement. Cette stratégie comprend la définition des rôles et des permissions dès le départ et l'évaluation des risques potentiels.

Actions spécifiques :

- Créer des diagrammes de flux de données pour identifier les points de vulnérabilité potentiels.
- Documenter les rôles et les permissions détaillés lors de la phase de planification.
- Collaborer avec les parties prenantes pour établir des politiques de gestion des accès dès le début du projet.

Hypothèse 2 : Appliquer le principe de moindre privilège de manière rigoureuse

Une des leçons clés que j'ai tirées de Koloka est l'importance du principe de moindre privilège. Je vais non seulement appliquer ce principe rigoureusement dans mes projets

futurs, mais aussi m'assurer que l'équipe de développement le comprenne et le mette en œuvre de façon cohérente.

Actions spécifiques :

- Configurer des rôles avec des permissions minimales strictement nécessaires pour accomplir leurs tâches.
- Effectuer des revues régulières des permissions attribuées aux utilisateurs pour supprimer les droits d'accès non nécessaires.
- Éduquer les membres de l'équipe sur l'importance de limiter les permissions.

Hypothèse 3 : Implémenter un middleware personnalisable pour la gestion des accès

Pour une gestion des accès optimisée, j'ai décidé de m'engager dans le développement d'un middleware personnalisable qui pourra être adapté à divers projets, y compris ceux utilisant Strapi ou toute autre technologie backend.

Actions spécifiques :

- Développer et tester un middleware modulaire qui vérifie les permissions des utilisateurs.
- Intégrer le middleware dès les premières phases de développement et le documenter pour en faciliter l'utilisation par d'autres développeurs.
- Mettre en place une stratégie de test approfondie, incluant des tests unitaires et d'intégration pour s'assurer qu'il fonctionne comme prévu.

Hypothèse 4 : Mettre en place une surveillance et des audits réguliers

Une autre leçon apprise concerne la nécessité d'une surveillance continue et d'audits réguliers pour maintenir la sécurité du système. La mise en place de ces pratiques aidera à détecter les failles de sécurité et les accès non autorisés de manière proactive.

Actions spécifiques :

- Utiliser des outils de suivi et de surveillance des accès pour garder un œil sur les activités suspectes ou non autorisées.
- Planifier des audits réguliers des rôles et permissions pour s'assurer qu'ils sont toujours pertinents et sécurisés.
- Maintenir des journaux d'accès détaillés et les analyser régulièrement pour détecter des anomalies.

Engagement Final : Poursuivre la formation en sécurité et gestion des accès

Pour maintenir un haut niveau de compétence en sécurité et gestion des accès, je m'engage à continuer ma formation et à suivre les évolutions du domaine. Cela inclut la participation à des ateliers, des webinaires et la lecture de publications spécialisées.

Actions spécifiques :

- Suivre des cours en ligne sur les dernières pratiques de sécurité et gestion des accès.
- Participer à des conférences et séminaires pour rester à jour avec les nouvelles tendances et technologies.
- Connexion avec des communautés professionnelles pour échanger des idées et des expériences.

Ces hypothèses et engagements constituent une feuille de route pour améliorer la manière dont je gère la sécurité dans le développement d'applications web. En les mettant en œuvre, je suis convaincu que je pourrai non seulement renforcer la sécurité de mes projets futurs, mais aussi contribuer de manière significative à l'excellence technique et stratégique des équipes avec lesquelles je travaillerai.

Conclusion

En conclusion, cette expérience au sein du projet Koloka m’a offert une occasion précieuse de croissance personnelle et professionnelle. En confrontant les défis liés à la gestion des accès et à la sécurité des données, j'ai non seulement développé des compétences techniques essentielles, mais j’ai également affiné ma capacité à anticiper et à résoudre des problématiques complexes.

Les vulnérabilités initiales que j'ai découvertes ont été des signaux d'alarme clairs sur l'importance d'intégrer une stratégie de sécurité dès les premières étapes du développement d'un projet. Elles m'ont également démontré la nécessité d'une approche proactive et réfléchie pour assurer la protection des données des utilisateurs.

En approfondissant mes connaissances sur les différentes techniques de contrôle d'accès, en particulier le contrôle d'accès basé sur les rôles (RBAC), j'ai compris que la sécurité passe non seulement par une bonne application technique mais aussi par des pratiques de gestion rigoureuses et une formation continue. Cette compréhension multifacette va dorénavant guider mes actions dans tous mes projets futurs.

J’ai formulé des hypothèses et engagements clairs, tels que la définition d’une architecture de sécurité dès les premières phases de projet, l'application rigoureuse du principe de moindre privilège, le développement et l’intégration d’un middleware personnalisable pour la gestion des accès, ainsi que la mise en place d’une surveillance et d’audits réguliers. Je suis convaincu que ces engagements contribueront à la création de systèmes plus sécurisés et plus robustes, tout en m’aidant à devenir un meilleur développeur et un meilleur gestionnaire de projet.

Par ailleurs, cette expérience m'a également inculqué une rigueur nouvelle en matière de sécurité et une vigilance accrue face aux potentiels risques. Elle m’a motivé à continuer à me former et à rester à jour sur les évolutions du domaine. En maintenant cette démarche

d'amélioration continue, je suis persuadé que je pourrai non seulement renforcer la sécurité de mes projets futurs mais aussi apporter une valeur ajoutée significative aux équipes avec lesquelles je collaborerai.

Ce parcours n’a été que le début d’une exploration plus vaste du domaine de la sécurité des applications web, et je suis impatient d'appliquer ces apprentissages et d'approfondir encore mes connaissances dans les projets à venir.

Bibliographie

Choudhary, J. (s.d.). *building-role-based-access-control-rbac-in-node-js-and-express-js*. Récupéré sur https://medium.com: https://medium.com/@jayantchoudhary271/building-role-based-access-control-rbac-in-node-js-and-express-js-bc870ec32bdb

Dewangan, A. (s.d.). *role-based-access-control-rbac*. Récupéré sur https://atuldewangan.medium.com/: https://atuldewangan.medium.com/role-based-access-control-rbac-476452806518

ENTRUST. (s.d.). *QU'EST-CE QUE LE CONTRÔLE D'ACCÈS BASÉ SUR LE RÔLE (RBAC) ?ur%20le%20r%C3%B4le%20(RBAC),les%20t%C3%A2ches%20d'un%20utilisateur.* Récupéré sur https://www.entrust.com/: https://www.entrust.com/fr/resources/learn/what-is-role-based-access-control#:~:text=Le%20contr%C3%B4le%20d'acc%C3%A8s%20bas%C3%A9%20sur%20le%20r%C3%B4le%20(RBAC),les%20t%C3%A2ches%20d'un%20utilisateur.

getambassador.io. (s.d.). *master-rbac-in-kubernetes*. Récupéré sur https://dev.to/: https://dev.to/getambassador2024/master-rbac-in-kubernetes-288h

https://dev.to/egeaytin/rbac-vs-rebac-when-to-use-them-47c4. (s.d.). *rbac-vs-rebac-when-to-use-them*. Récupéré sur https://dev.to: https://dev.to/egeaytin/rbac-vs-rebac-when-to-use-them-47c4

Strapi. (s.d.). *Create new Role-Based Access Control (RBAC) conditions*. Récupéré sur https://strapi.io/: https://docs.strapi.io/dev-docs/configurations/rbac