Back to Documentation Menu

♡ Les Mainteneurs de Blitz

Topics

Jump to a Topic

Hormis l'équipe principale, il existe deux niveaux de mainteneurs décris ci-dessous.

Devenir un Mainteneur

Nous avons toujours besoin de mainteneurs de Niveau 1 ! La principale condition à remplir est que vous puissiez montrer de l'empathie lors de vos communications en ligne. Nous vous exerceronsau besoin aux autres spécificités. **C'est le rôle qui vous convient si vous avez peu de temps à consacrer, car vous pouvez dédier autant de temps que vous le souhaitez sans aucune responsabilités permanentes (contrairement aux mainteneurs de Niveau 2)**.

Les mainteneurs de niveau 2 ont de plus grandes responsabilités. Généralement, vous passerez plus de temps en tant que mainteneur de Niveau 1 avant de pouvoir passer au Niveau 2.

Veuillez envoyer un Message Privé à un membre de l'équipe principale (Brandon Bayer) sur Discord si vous êtes intéressé à devenir un mainteneur officiel !

Les Mainteneurs de Niveau 1

Les mainteneurs de niveau 1 sont essentiels à une communauté saine ainsi qu'au au bon déroulement du projet. Ils enlèvent un poids considérable à l'équipe principale et aux mainteneurs de niveau 2 pour que ces derniers puissent se concentrer sur des sujets à plus haut niveau ayant un impact à plus long terme.

Les responsabilités élémentaires des mainteneurs de niveau 1 sont :

- Être une voix amical et accueillante pour la communauté Blitz.
- Trier les issues

- Trier les issues.
- Trier les pull request.
- Surveiller et répondre dans les canaux d'entraide sur le serveur Discord.
- Suivre les issues assignées afin d'assurer leur bon fonctionnement.
- Suivre les PRs en cours pour garantir leur bonne continuité.
- Encourager la communauté.
- Gérer la communauté.

Les Mainteneurs de Niveau 2

Les mainteneurs de niveau 2 sont la charpente du projet. Ils sont les gardiens du code, assurant sa qualité, sa justesse et sa sécurité. Aussi, ils facilitent un rythme de progression rapide.

Les responsabilités élémentaires des mainteneurs de niveau 2 sont :

- S'approprier certaines parties du code du projet.
- Maintenir et améliorer la structure de leurs parties.
- Effectuer les dernières revues des pull requests.
- Fusionner les pull requests.
- Réaliser le suivi et assurer la progression des pull requests ouvertes.

Prendre congé

Les mainteneurs ont la possibilité de se retirer de leur rôle à n'importe quel moment sans aucune honte ou culpabilité. Il suffit juste d'en informer l'équipe principale !

⚠ Les Fondamentaux

Les mainteneurs sont le visage du projet ainsi que le point de contact de première ligne pour la communauté. Par conséquent, les mainteneurs ont le devoir de mettre à l'aise les personnes afin qu'ils se sentent bien accueillis, estimés, compris et appréciés.

Veuillez prendre le temps de lire et de comprendre tout ce qui est écrit dans ce guide sur comment créer une communauté accueillante

Quelques points importants :

- **Gratitude :** Exprimer immédiatement votre gratitude lorsque quelqu'un ouvre une issue ou une PR. Cela demande un peu de temps et d'effort et nous apprécions ce geste.

- **Réactivité :** durant le tri d'une issue/PR, même si nous ne pouvons réaliser une revue complète de suite, n'hésitez pas à un commentaire en les remerciant et en disant que nous effectuerons une revue prochainement.
  - Notre but est de répondre à toutes les issues et les PRs dans les 2 jours, ou plutôt idéalement dans la journée.
- **Compréhension :** Il est essentiel de vous assurer d'avoir bien compris ce que quelqu'un dit avant de lui répondre. Demandez toutes les questions possibles pour vous aidez à mieux comprendre si nécessaire. Ce serait malheureux si quelqu'un devrait répondre à votre retour en disant "en fait, j'étais en train de demander ceci".
  - En effet, au moins une question est bien souvent requise avant de pouvoir répondre adéquatement — que ce soit sur GitHub ou sur Discord.

Les Sources

- **Comment Être Utile en Ligne**
- **Comment Débuter une Revue sur un Code**

Surveiller le Serveur Discord

- Nous voulons que chaque question obtienne sa réponse, idéalement dans la journée, ou en deux jours maximum. Cependant, cela ne veut pas dire que vous devez impérativement résoudre leur problème. Par exemple, s'ils demandent après une librairie quelconque dont vous ne connaissez absolument rien, vous pouvez répondre que ce n'est pas votre domaine, mais qu'ils peuvent essayer de regarder certains exemples/documentations leur permettant de l'utiliser dans nextjs, comme le ferait l'integration.
- Assurez-vous que les fils de discussion sont bien utilisés afin de conserver une bonne organisation des canaux. Même s'il n'y a que peu de messages à propos d'un sujet dans le canal principal et que l'issue/la question n'a pas été résolue, n'hésitez pas à créer un fil de discussion et à y répondre.
- Si quelqu'un reporte quelque chose qui s'apparente à un bogue, demandez-leur gentiment d'ouvrir une issue sur Github.
- Si quelqu'un requiert une fonctionnalité ou un changement, demandez-leur gentiment d'ouvrir une issue pour une requête de fonctionnalité sur Github.
- Si vous remarquez un problème de type DX ou une opportunité d'amélioration dans une conversation au hasard, veuillez ouvrir une issue sur Github à ce sujet. Réaliser un grand nombre de petites modifications sur les plus longues permet d'avoir des retours conséquents.
- Si quelqu'un pose une question à propos de quelque chose qui ne possède aucune documentation ou qui n'est pas renseignée correctement, prenez donc un instant pour ajouter la documentation à ce sujet. Puis, copiez collez le lien à la nouvelle documentation que vous venez d'ajouter.

sujet. Puis, copiez-collez le lien à la nouvelle documentation que vous venez d'ajouter.

Le Tri des Issues

Si un bogue a été reporté :

- Y a-t-il suffisamment d'informations à son sujet ? Quelles versions ? Quels Logs? Y a-t-il un moyen de le reproduire ?
- A-t-il déjà été résolu dans une précédente publication ?
- Y a-t-il une issue déjà existante à ce propos ?

Si une requête de fonctionnalité/changement :

- La requête demandée est-elle clairement expliquée et quels en seraient les bénéfices ?
- Est-ce une victoire évidente pour Blitz ? Si oui, acceptez-la.
- Si ce n'est pas le cas, veuillez en référer un membre de l'équipe principale ou un mainteneur de niveau 2 pour une revue en profondeur.

Les Actions

1. Ajouter des tags:
   - Ajoutez un tag `status/*`.
   - Ajouter un tag `kind/*`.
   - Ajouter un tag `scope/*`.
   - Ajoutez un tag à l'issue si applicable.

Tri des Pull Requests

- Les changements sont-ils couverts par les tests ?
- Les changements semblent-ils corrects ? Assurez-vous qu'il n'y ait pas d'issues évidentes.
- Si applicable, une PR a-t-elle été ouverte pour mettre à jour les documents ?

Les Actions

1. Demandez gentiment une requête pour tout changement nécessaire.
2. Autrement, ajoutez une approbation GitHub afin que les mainteneurs de niveau 2 sachent que la requête possède déjà une revue initiale.

Finaliser & Fusionner la revue de PR (Mainteneurs de Niveau 2)

En tant que mainteneur de niveau 2, il est de votre devoir de vous assurer que le code fonctionne et qu'aucune régression ne parvienne à la branche canary.

1. Assurez-vous que le code PR'ed fonctionne pleinement et correctement comme prévu et qu'il n'y ait aucune régressions dans le code associé.
    1. S'il n'est pas totalement couvert par des test automatiques, vous devez abaisser localement le code et manuellement tout vérifier (le GitHub CLI aide à ce propos !).
2. Durant l'écrasement & la fusion :
    1. Changez le titre de la commission afin qu'il soit publiquement compréhensible - ce texte exact ira dans les notes de version.
    2. Ajoutez un type de commission dans la description, entre parenthèses comme il suit `(patch)`. types de commission :
        - `major` - changement majeur de rupture
        - `minor` - ajout de fonctionnalités mineures
        - `patch` - patches, résolution de bogues, améliorations de performance, etc
        - `newapp` - changements au nouveau schéma d'application
        - `recipe` - changements à une recette Blitz
        - `example` - changement à un exemple d'application

Idea for improving this page? Edit it on GitHub.

← How to Contribute

Code of Conduct →

Docs

All Docs

Get Started

How To Contribute

Community
Discord

Forum Discussions

Twitter

Showcase

Other

Deploy with Flightcontrol

GitHub

Wiki

Swag

Hosted on ▲ Vercel