Zeste de savoir

# Première release candidate pour la 1.0 de RLS !

21 janvier 2019

# Table des matières


1. Introduction . . . . . 1
2. Support . . . . . 2
3. Fonctionnalités proposées . . . . . 2
4. Conclusion . . . . . 3
5. Source . . . . . 3


## 1. Introduction

RLS est une *chaîne d'outils*, indépendante de toute plateforme connue (e.g. *Eclipse*, *IntelliJ*), destinée à fournir le confort d'un IDE tout en offrant une interface aux éditeurs qui souhaitent la supporter. Dans les faits, RLS est un serveur communiquant en arrière-plan avec un client (qui peut être un éditeur ou un environnement complet), qui se chargera de fournir les diagnostics respectifs de chaque outil du backend. Par ailleurs, vous pouvez consulter une implémentation du frontend de RLS pour *Visual Studio Code* sur son dépôt .

Ce 19 août, la première release candidate à la version `1.0` a été annoncée et prévue pour le 3 septembre. Durant cette période, la communauté est vivement encouragée à chercher le moindre bug, le moindre problème et à le signaler  pour ceux qui en ont le temps et l'envie. Passons à la révision !

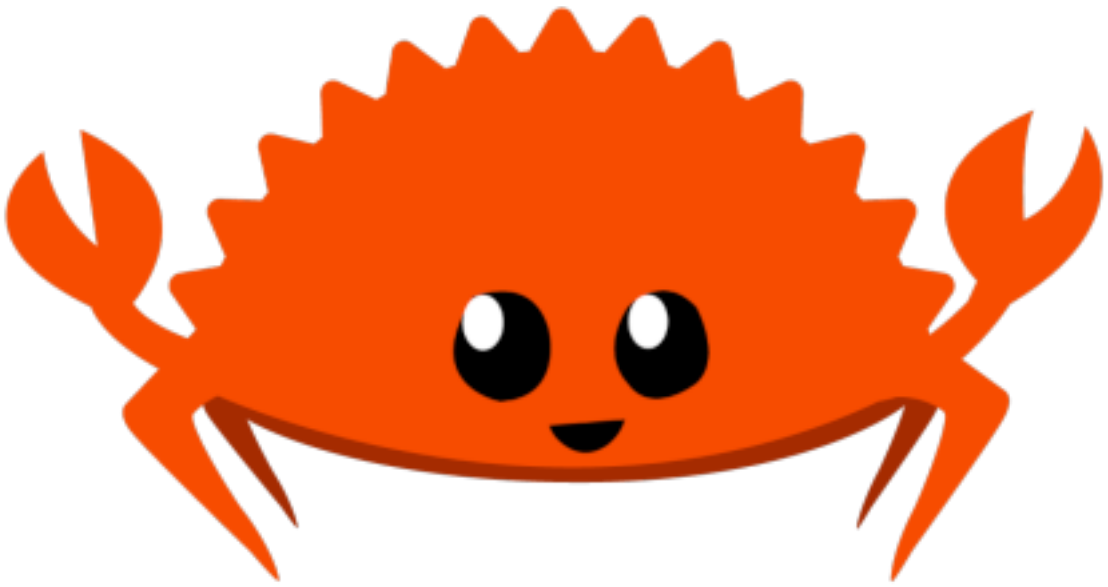

FIGURE 1. – Ferris le crabe (mascotte officieuse de Rust) !

## 2. Support

Bien que toute jeune, la suite est supportée par 4 outils (dont 1 IDE) présentés ci-dessous :

1. Visual Studio Code ;
2. Atom ;
3. Sublime Text ;
4. Eclipse (lien original ).

Rust disposant dès maintenant de son propre module pour *IntelliJ* , nous pourrions espérer que cette plateforme rejoigne bientôt la liste !

## 3. Fonctionnalités proposées

Au même titre qu'un environnement de développement intégré "classique", vous pourrez y retrouver tout le nécessaire pour ne plus avoir à jouer avec `ALT+TAB` toute la journée.

#### 3.0.1. Coloration syntaxique

Chaque éditeur (standalone ou non) supportera sa propre coloration pour la syntaxe du langage.

#### 3.0.2. Complétion du code

Comme précisé dans le billet, la complétion ne sera que syntaxique. Cela signifie que l'analyseur peut parfois vous transmettre un diagnostic incorrect dû à son champ d'action réduit. Comme contre exemple, je pourrais présenter l'analyseur statique des IDE *IntelliJ* qui, lui, analyse également la structure de votre code et peut pratiquement remplacer les retours du compilateur du langage cible (ce n'est toutefois pas encore le cas pour Rust).

Pour les curieux, cette fonctionnalité est fournie par la crate *Racer* .

#### 3.0.3. Erreurs et avertissements

Les erreurs et avertissements normalement renvoyés par le compilateur seront directement affichés à la ligne près, rendant la résolution de ces erreurs de compilation plus aisée.

#### 3.0.4. Formatage

Géré évidemment par *Rustfmt*, lui-même en phase de passer à sa première version stable (`0.99.3`) à l'heure où j'écris ces lignes.

#### 3.0.5. Analyse intelligente du code

Comme cité plus haut, cette fonctionnalité se rapproche de l'analyse statique. Entre autre, elle permettra de :

— détecter *la documentation* d'une signature en passant sa souris dessus au lieu d'aller directement chercher ce que nous voulons sur le site de la crate en question ;
— trouver toutes *les occurrences* d'une signature dans notre code ;
— détecter *les implémentations* des traits et des types concrets ;
— détecter et réajuster *les modèles d'imports* obsolètes ;
— ...

En revanche, il est précisé que toutes ses aides à l'analyse ne seront pas disponibles au sein de la logique écrite dans les macros, voire même lors de leur utilisation. Elles ne seront pas non plus disponibles pour des éléments parents d'un élément courant.

```
foo::bar::baz
// ^ Dans ce chemin, nous ne
// bénéficierons que d'une analyse
// pour `baz` et non `bar` ni `foo`.
```

## 4. Conclusion

Me concernant, j'ai, jusqu'ici, toujours utilisé chacun de ces outils séparément (sans RLS, donc) et offrent déjà, **selon moi**, une aide très précieuse sur beaucoup de points. N'étant "qu'une" amélioration de ce qui existe déjà, cette préversion sera un prétexte comme un autre pour tester la chaîne et certainement l'adopter.

Amusez-vous bien !

## 5. Source

— Billet du blog de Nick Cameron

# Liste des abréviations

**RLS** Rust Language Server. 1, 3