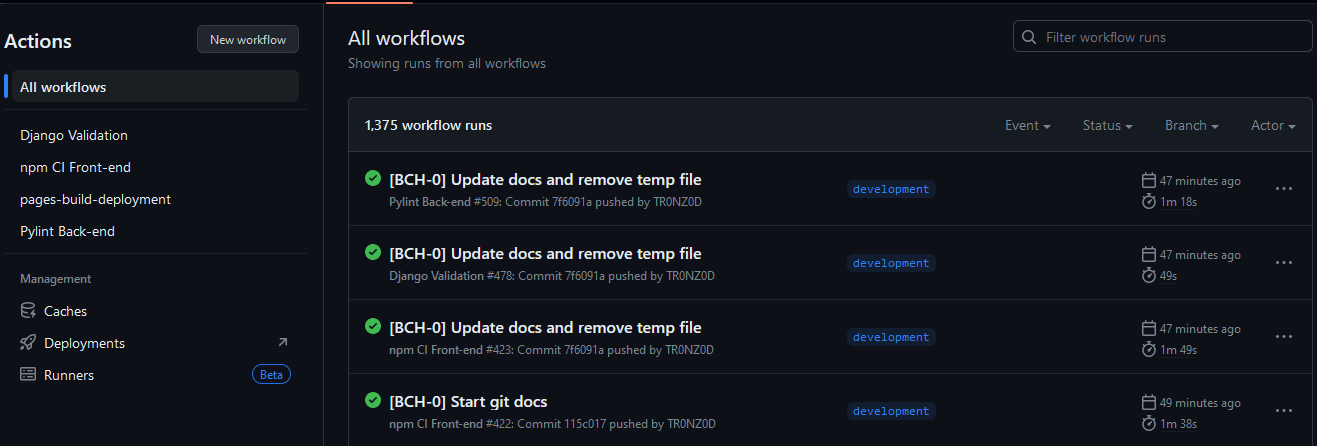
Para mantermos um controle de versão e suportar o desenvolvimento simultâneo do projeto utilizamos o GitHub como o principal local para armazenar e organizar o código. O GitHub é um serviço baseado em nuvem que hospeda um sistema de controle de versão (VCS) chamado Git, o qual é utilizado para controlar o histórico de alterações de arquivos e principalmente de projetos, permitindo mais flexibilidade no fluxo de trabalho, segurança e desempenho. De todas as funcionalidades que o serviço disponibiliza, as principais utilizadas foram: workflows, branches, issues, pull requests, projects e controle de dependência automatizada.

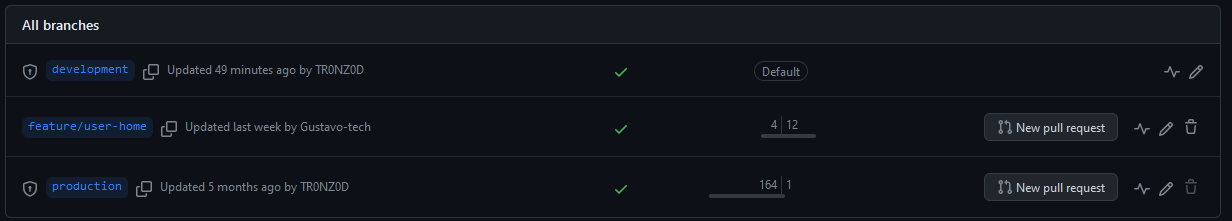
 

Um workflow é um conjunto de processos e ferramentas automatizadas que auxiliam no controle de qualidade do código, o projeto utiliza três principais workflows. Um para a avaliação do ambiente de frontend, outro para a avaliação do código Python escrito e outro para a avaliação do ambiente de backend.

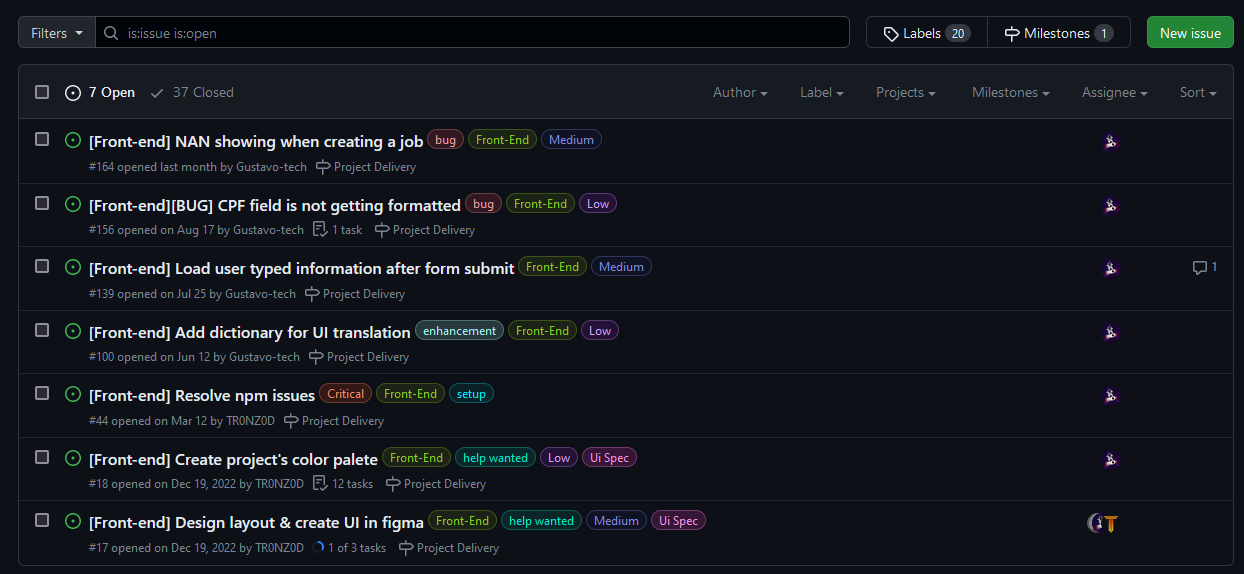
O workflow para avaliação do ambiente de frontend consiste em compilar o código utilizando a própria biblioteca do Node. Já o fluxo para avaliação do código Python consiste na utilização de uma biblioteca de padrão de código chamado Pylint, o qual irá validar diversos pontos e informar uma “nota” para o código, caso essa nota seja menor que 9, o workflow irá falhar. Por fim, o workflow para validação do ambiente de backend consiste na atualização da estrutura do banco de dados, coleta de arquivos estáticos, avaliação do código e execução de casos de testes.



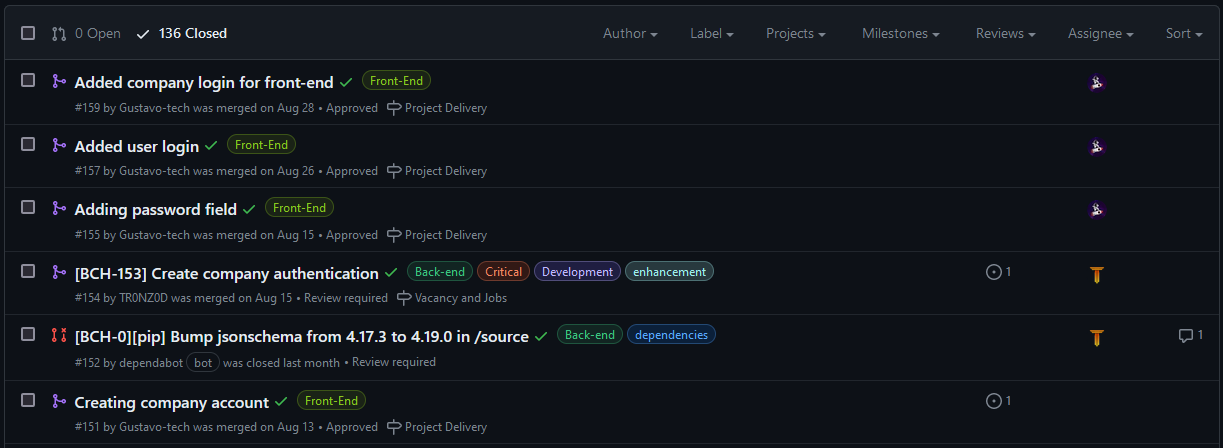
As branches são ramificações do projeto que possuem o objetivo de isolar desenvolvimentos e alterações para impedir modificações, quebras e problemas em outros ambientes. O nosso projeto possui duas principais ramificações, uma para desenvolvimento e outra para produção. A ramificação de desenvolvimento é destinada para o uso cotidiano, onde serão enviados os códigos finalizados, todas as ramificações pessoais (utilizado para o desenvolvimento ativo) são baseadas nessa branch, permitindo um controle ainda maior nas alterações. Já a ramificação de produção tem o objetivo de centralizar o ambiente e o código que estará publicamente disponível para o usuário final, isolando o ambiente e evitando que modificações de desenvolvimento quebrem o ambiente.



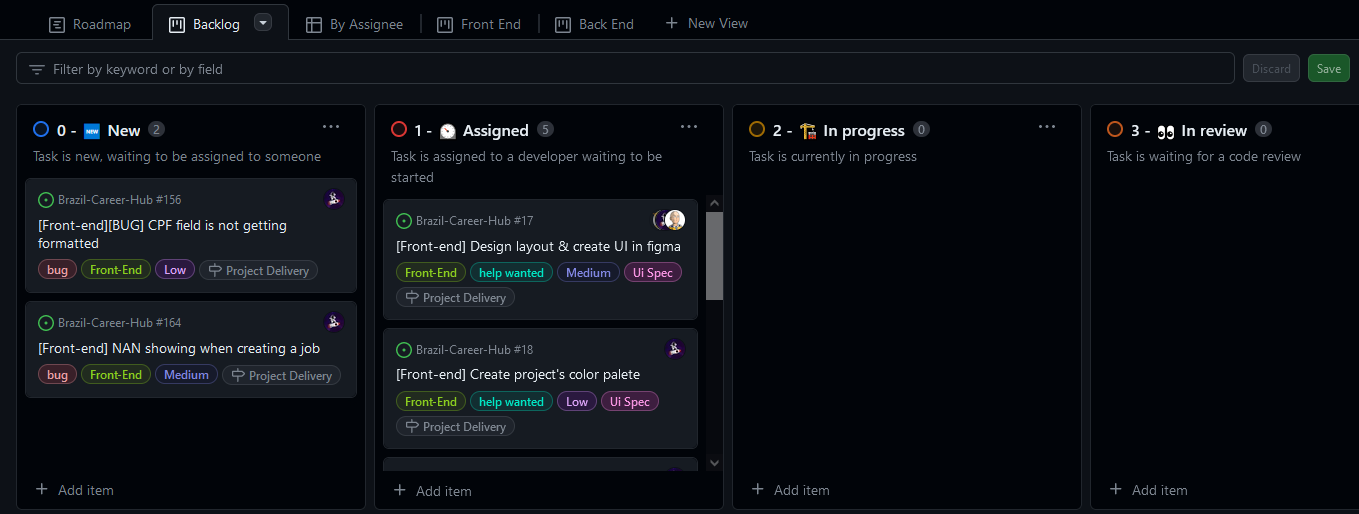
Os issues (do Inglês, problemas) são “tickets” abertos para controle de tarefas e problemas que devem ser corrigidos, neles são descritos os passos para reprodução, qual ambiente é afetado por esse problema, qual a criticidade do problema, quem será o responsável da resolução desse problema entre outros. Esses issues são os principais agentes para o controle de tarefas e pendências do projeto, permitindo que novas implementações e correções possam ser realizada futuramente sem correr o risco de “cair no esquecimento”.

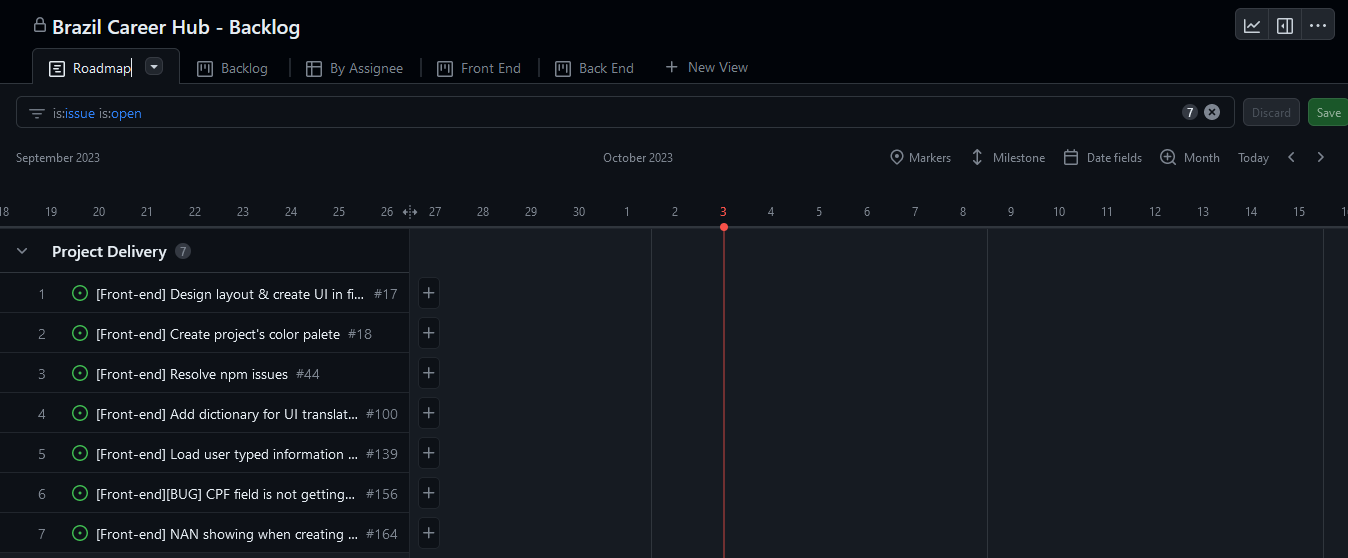


Conforme comentado anteriormente, projeto possui algumas ramificações para permitir o isolamento do desenvolvimento. Quando o desenvolvimento é finalizado, as alterações dessas ramificações devem ser mescladas com as alterações existentes da ramificação pai (no caso, a de desenvolvimento), para realizar essa mesclagem é necessária a abertura de um pull request, essa requisição é um método para permitir que outros desenvolvedores revisem o novo código antes de ser oficializado numa branch geral, no pull request também são executados os pipelines, os quais validam a qualidade do código de forma automatizada para que a revisão por outros desenvolvedores possa ser mais rápida e fácil.



A funcionalidade de projects é algo que foi recentemente adicionado ao GitHub, permitindo uma exibição, ordenação e gerenciamento melhor, mais otimizado e persistente dos issues, conforme comentado anteriormente, os issues são constantemente utilizados para manter um controle referente aos problemas e pontos pendentes do projeto, o projects pode ser utilizado para criar boards, roadmaps entre outros recursos visuais e automatizados para facilitar esse controle. O nosso projeto utiliza um board geral, dois específicos e um roadmap, permitindo que o desenvolvedor tenha uma melhor visão de suas tarefas em andamento, e pendentes.





Por fim, mas não menos importante, temos o controle de dependência automatizada. Comumente conhecido como Dependabot, o autômato criar pull requests ao detectar uma nova versão de alguma biblioteca ou dependência que o projeto utiliza, qualquer que seja o ambiente. Além disso, também aponta falhas de segurança e outros problemas relacionados a dependências e pacotes externos utilizados, mantendo uma maior qualidade, atualidade e segurança no projeto.

