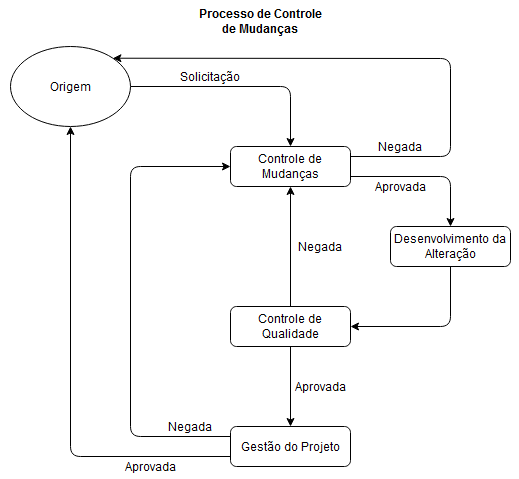
**Gerenciamento de Requisitos**

**4.1 Descrição do Processo de Gestão**

****O processo de gestão de mudanças e gestão de configuração de software é realizado no processo de desenvolvimento de software seguindo as seguintes especificações:  
Ao se detectar a necessidade de realização de uma mudança, o requerente deverá preencher um formulário de solicitação de alterações com as informações referentes à mesma, como a data, o motivo da alteração, as seções alteradas e o detalhamento das alterações. A equipe de controle de mudanças então irá realizar a análise da solicitação, podendo essa ser negada ou aprovada, e no caso de aprovação esse formulário será encaminhado, acrescido das considerações necessárias, ao time de desenvolvimento. Após o desenvolvimento da alteração, a mesma é avaliada pelo nosso controle de qualidade, caso negado retorna ao controle de mudanças, caso aprovada avança para a próxima etapa. Nesta ultima etapa a mudança então é analisada em conjunto com a gestão para garantir a sua integração harmoniosa ao projeto final. A equipe encarregada de realizar o controle de mudanças consiste no time de programadores, para o desenvolvimento da mesma, nos analistas e testadores, para garantir a qualidade do produto e, por fim, os gerentes do projeto que irão certificar a integridade do trabalho.

**4.1.1 Formulário de Solicitação de Alterações**

|  |  |  |
| --- | --- | --- |
| **Cliente** | **Projeto** | **Versão do documento** |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Histórico de Alterações** | | | |
| **Data** | **Autor** | **Seção alterada** | **Motivo da alteração** |
|  |  |  |  |

**ESPECIFICAÇÕES**

|  |  |
| --- | --- |
| **Alteração** | **Descrição** |
|  |  |

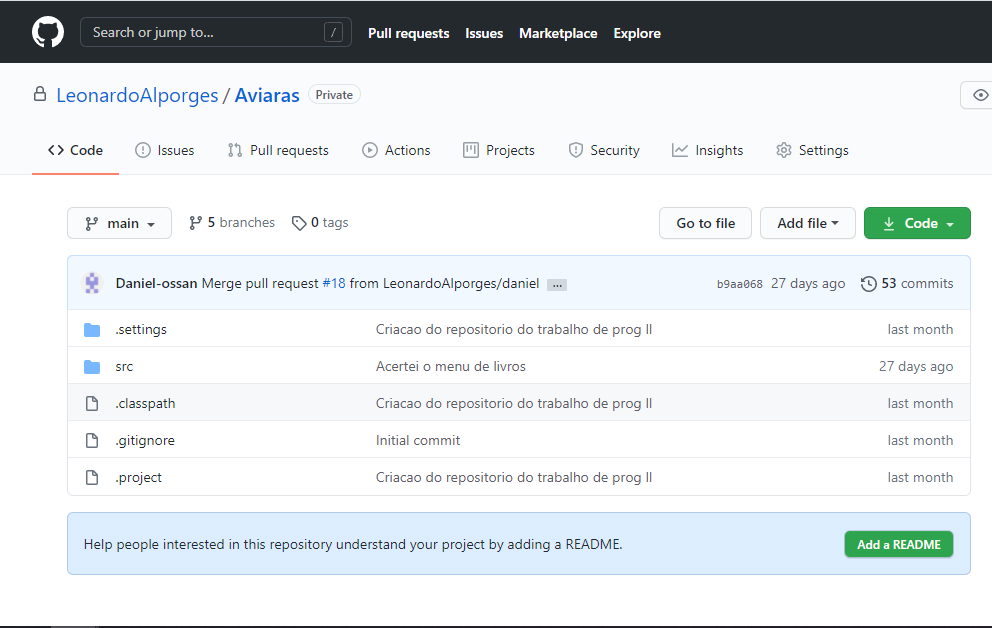
|  |  |  |
| --- | --- | --- |
| APROVAÇÕES | | |
|  |  | Data |
| Patrocinador |  |

|  |  |  |
| --- | --- | --- |
|  |  | Data |
| Diretor Operacional |  |

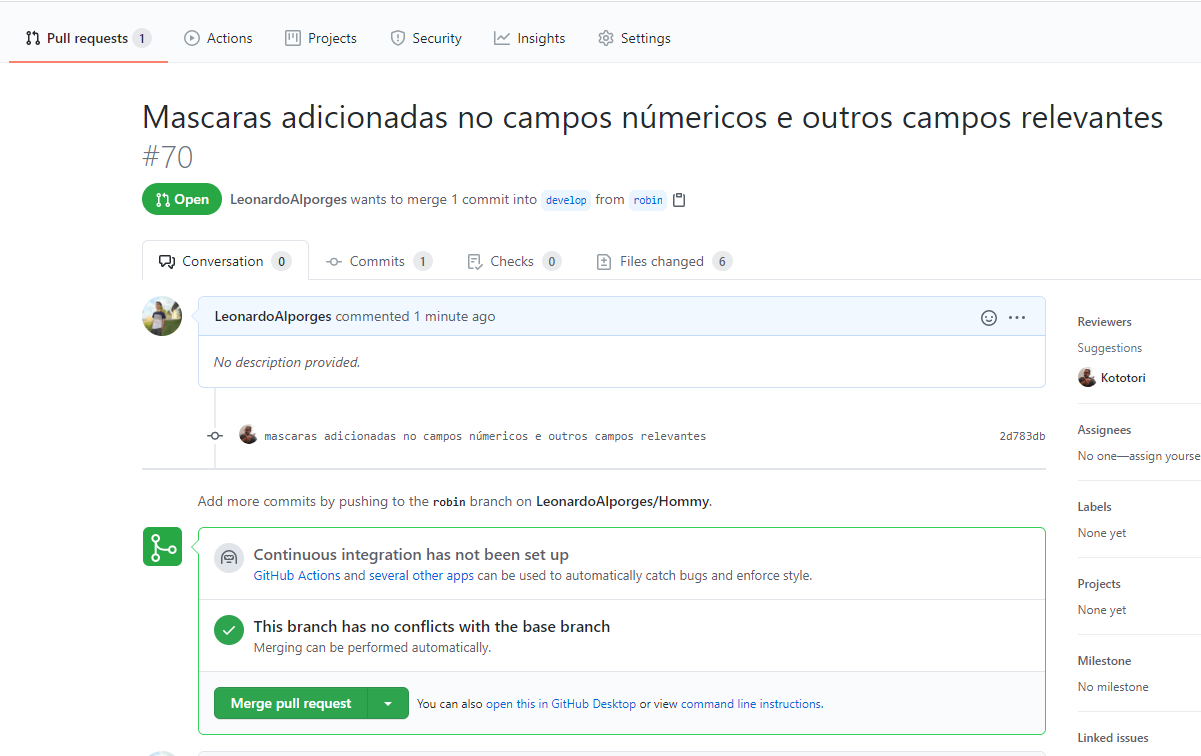
|  |  |  |
| --- | --- | --- |
|  |  | Data |
| Gerente de Projetos |  |

**4.2 Informações Adicionais**

Para controle de alteração e mudança será utilizado a ferramenta de controle de versão GitHub, que é uma ferramenta é um sistema de gerenciamento de projetos e versões de códigos, com ela podemos controlar as versões dos sistemas com maior facilidade. Essa ferramenta nos dará a possibilidade de controlar o que entra no projeto e facilidade na alteração de algum modulo caso seja necessário e sempre que é alterado algo no projeto cria-se uma versão do projeto fazendo assim com que não seja perdida nenhuma informação.

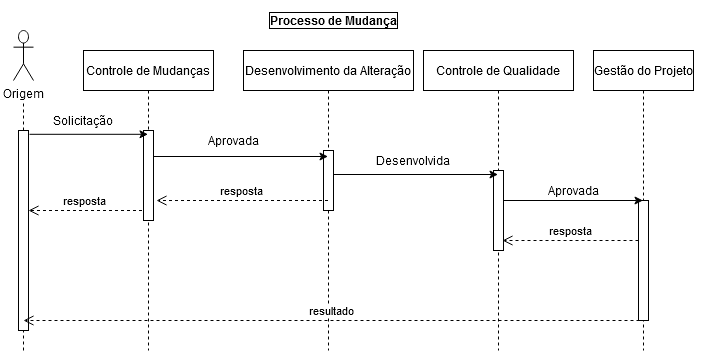


O GitHub usa um sistema de branch que é onde ficam as versões, cada branch recebe um nome específico definido pela equipe de desenvolvimento, que deixa claro o que está em cada branch. Por padrão é usam-se as branchs Master/Productions, Develop e Homologação. Essas branch são as principais de um projeto pois nelas estão os códigos e se baseiam outra branch de desenvolvimento.

Além do mais o GitHub também conta com um sistema de pull request que é por onde as alterações passam por uma avaliação antes de serem inseridas no branch de destino.

Após passar pelo processo de pull request as alterações são merged (unificadas) com as da branch de destino, isto é, caso não tenha nenhum conflito entre as branch, e então uma nova versão estará disponível na branch de destino.

**4.3 Modelagem de Interações**

****

**Verificação e Validação de Requisitos**

**5.1 Descrição do Processo de Verificação e Validação**

O processo de verificação e validação de requisitos (V&V) será aplicado utilizando das seguintes técnicas:

* Revisão e Inspeção de Requisitos: Uma equipe de revisores analisa sistematicamente a especificação produzida, realizando inspeções constantes para remoção de defeitos, utilizando como principal método as revisões informais e reuniões com a equipe (clientes, usuários e analistas), visando a fácil comunicação entre os atores, tornando tudo mais simples e de fácil compreensão.

Realizam também o uso de checklists para prosseguir com o desenvolvimento das correções e demais afazeres, seguidos de um follow-up frequente dos usuários.

* Prototipação: A equipe de Analistas segue com o PencilProject, onde foi criado o primeiro protótipo-base do software. Nas etapas seguintes será mantido o uso do mesmo para uma resposta rápida e barata aos feedbacks constantes dos usuários, auxiliando também na detecção preventiva de futuros problemas devido a fácil compreensão.
* Geração de Casos de Teste: Baseando-se nos requisitos adquiridos desde o início do projeto, os testadores elaboram casos de teste a partir de cenários de casos de uso, utilizam da ferramenta TestRail que lhes permite executar testes, criar projetos e simular bugs, com foco no feedback se mantendo sempre atentos as novas atualizações de acordo com o avanço do restante da equipe e as necessidades dos usuários.

**5.2 Registro de Revisões**

O sumário estava desatualizado e foi atualizado conforme as edições feitas no documento. Os títulos das sessões 3.2 e 3.3 estavam incorretos e as sessões 4.2 e 5.1 continham erros ortográficos e de formatação.

A Modelagem de Domínio na página 7 apontava algumas classes que se relacionavam sem serem devidamente representadas com as setas preenchidas entre as classes Operador e Cliente para a classe Livro e os nomes das atribuições não foram informados. Também havia relacionamentos inapropriados entre o Fornecedor e Cliente na parte das subclasses de Operador. Foi criada uma classe abstrata “Pessoa”, para definir as demais classes que representam uma pessoa, sendo ela a classe mãe e as outras as suas subclasses.

O Diagrama de Casos de Uso da página 8 possuía erros na representação do diagrama, alguns casos se referiam a processos internos ou ações que ocorrem dentro de casos de uso, não sendo assim adequadamente representados como um caso de uso a parte. Sendo assim, o diagrama foi refeito seguindo os princípios adequados baseando-se nos casos de uso recém redigidos. A narrativa estava com redação incorreta, não objetiva e coloquial. Foi redigida uma nova narrativa com a linguagem apropriada. Os processos tiveram suas interações melhor detalhadas, expandidas e abrangendo casos de exceção. Os casos de uso “Solicita atendimento” e “Faz doação” não estavam adequados. Havia interações entre dois atores, sendo que um caso de uso representa uma interação entre um sistema e um ator. Além disso, alguns estados, como o “Voltar para a tela de início” não estavam corretos e foram corrigidos na reformulação.

A Modelagem de Estados na página 13 estava com notação incorreta. Foi usada notação de diagrama de atividades. Foi feito um novo modelo usando os princípios da representação da modelagem de estados e que respeitasse as características do projeto.

**5.3 Casos de Teste**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se usuário consegue reservar um livro que está alugado. | | | | |
| Resultado esperado | Usuário não deve conseguir realizar a reserva do livro. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se existe disponibilidade para livros que inexistentes no sistema. | | | | |
| Resultado esperado | Sistema não deve trazer dados. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se um operador consegue realizar cadastro duplicado. | | | | |
| Resultado esperado | Sistema deve retornar erro. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se um operador consegue realizar um cadastro. | | | | |
| Resultado esperado | Cadastro deve ser bem sucedido. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se é possível registrar mais de um cliente com mesmo CPF. | | | | |
| Resultado esperado | Não deve ser possível realizar esse cadastro. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se um operador consegue realizar login no sistema. | | | | |
| Resultado esperado | Operador deve conseguir realizar login. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se o sucesso do registro de reservar de livro é informado para o cliente. | | | | |
| Resultado esperado | O cliente deve receber uma mensagem informando que sua solicitação foi concluída com sucesso. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se livro doado está disponível para locação imediatamente após cadastro. | | | | |
| Resultado esperado | A locação deve estar disponível. | | | | |
|  |  |  |  |  |  |
| Tipo do teste | Funcional | | | | |
| Objetivo do teste | Verificar se livro locado está disponível para locação de outro cliente. | | | | |
| Resultado esperado | O livro locado não deve estar disponível. | | | | |

**APÊNDICE E -   
Log de Commit do GitHub**

|  |
| --- |
| \*\*Log de commit bibliotech\*\*  ID-Nome autor-Dia da alteração-Descriçao commit-  01 Leonardo Alporges Martins - 05/12/2020 - Adição do arquivo de log de commit  02 Leonardo Alporges Martins - 07/12/2020 - Adição dos arquivos de modelagem e os prototipos do sistema  03 Juliano Orlandi Lopes - 09/12/2020 - Adição do PDF corrigido do diagramas de classes, e adição do documento de Controle de versões e V&V.  04 Lucas dos Santos Carvalho - 09/12/2020 - Adição do documento parte 3 de Gerenciamento de requisitos  05 Leonardo Alporges Martins - 09/12/2020 - Adicição do topico 4.2 no Documento de Gerenciamento  06 Lucas dos Santos Carvalho - 09/12/2020 - Atualização de Documento e inserções da correções do tópico anterior  07 Alef Biano dos Santos - 09/12/2020 - Adição dos casos de testes do sistema  08 Daniel Oliveira Souza - 09/12/2020 - Adição de registro de reviso e correção diagrama de estado |