CENTRO UNIVERSITÁRIO UNIRUY WYDEN

FELIPE DAVIS TRAVASSOS DA SILVA

PROJETO INDEXADOR DE STASH PATH OF EXILE

SALVADOR
2020

FELIPE DAVIS TRAVASSOS DA SILVA

PROJETO INDEXADOR DE STASH PATH OF EXILE

Projeto de TÓPICOS EM DESENVOLVIMENTO DE SISTEMAS a fim de obter avaliação das características e necessidades do software.

Salvador
2020

SUMÁRIO

1. Introdução

O presente documento refere-se ao projeto de software que será apresentado como resultado para à disciplina Topicos em desenvolvimento de sistemas a fim de se obter aprovação na mesma. Tal projeto terá como seu conteúdo componentes fictícios para aplicar as técnicas aprendidas. Desta forma serão feitas pesquisas bibliográficas referentes tanto a livros, documentações de tecnologias, artigos etc. Ao concluir este artigo teremos passado por diversas pesquisas técnicas e teóricas acrescentando assim bagagem e conhecimento ao nosso currículo.

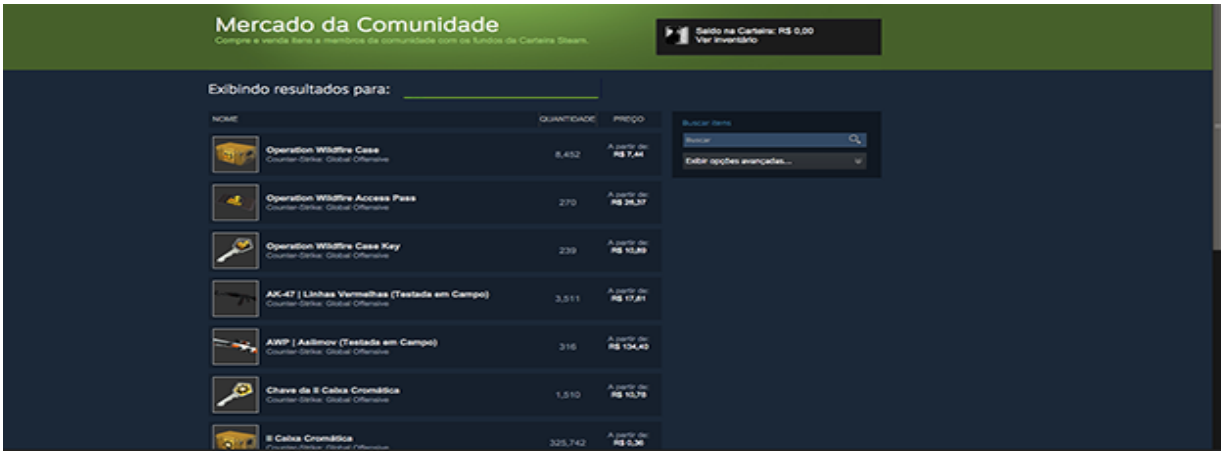
2. Apresentação do sistema

A ideia para desenvolvimento do projeto de software deste trabalho deriva-se de um outro software, no caso um *game* virtual, que possui um sistema busca por informações de outros jogadores disponibilizados por uma *Application Programing Interface* (API). O *game* em questão é o Path of Exile, ele possui um sistema de armazenamento de itens que pode ser exposto para indexadores disponibilizarem a busca por recursos a venda ou troca.

O software terá uma base de dados adquirida a partir da API proprietária do *game* e será disponibilizados itens a venda ou troca através de uma interface *web* que realizará requisições ao backend. Assim, como Minimum Viable Product (MVP) o sistema realizará apenas indexação e busca por itens e exibindo-os através de uma interface.

3. Modelo do processo de desenvolvimento

Com objetivo de colocar o software em operação rapidamente e com baixo custo, entregando um produto mínimo viável (MVP), o desenvolvimento do mesmo seguirá o modelo de prototipação, possibilitando o feedback rápido dos usuários, bem como o da comunidade de desenvolvedores, já que o software se trata de uma iniciativa open source.



4. Análise de requisitos

Neste tópico serão apresentados os requisitos necessários para o desenvolvimento do software. Serão listados os requisitos funcionais, organizacionais e não funcionais.

4.1 Requisitos funcionais

- O *software* deve identificar corretamente a conta do usuário no jogo.
- O *software* deve apresentar os itens que o usuário dispõe no jogo.
- O *software* deve listar os itens disponíveis para aquisição do usuário por categorias.
- O *software* deve conter um espaço de chat entre os usuários.
- O *software* deve conter um mecanismo de busca com ferramentas de filtragem

4.2 Requisitos não funcionais

- A base de dados deve atualizar todos os itens disponíveis para troca em no máximo 20 minutos, pois deverá ter uma resposta rápida de quais itens já foram trocados.

- O sistema deve está disponível todos os dias durante o dia inteiro, porém em caso de falhas deverá garantir que os dados estão atualizados de acordo com a última atualização da *API*.
- O *software* deve conter uma interface de usuário e responsiva, com elementos bem dispostos na tela.
- A seleção das especificidades do item deverá ser feita de maneira simples atraveś de sistema *drag and drop* (arrastar e soltar).
- O Aplicativo deve ser portável para dispositivos móveis, nos sistemas operacionais *Android* e *IOS*, além de *Web*.
- O software deve ser desenvolvido usando o *Design Patterns Gang of Four (GoF).*
- O software deverá ser implementado utilizando as técnicas de clean code de Robert Martin.
- Apenas usuários que forem listados pela API do fornecedor do jogo serão indexados pelo software a fim de não ir de contra as regras do jogo ou qualquer lei de privacidade.
- Para a consulta dos dados a partir do *front-end* será permitida apenas requisições do tipo GET.
- As inserções de dados serão permitidas apenas para um middleware responsável em atualizar as informações cujo mesmo deterá um token de autenticação.

4.3 Elicitação de requisitos

A elicitação de requisitos foi desenvolvida a partir de depoimentos mais comuns entre os jogadores, com isso foram destacados os principais pontos de insatisfação com as ferramentas já existentes.

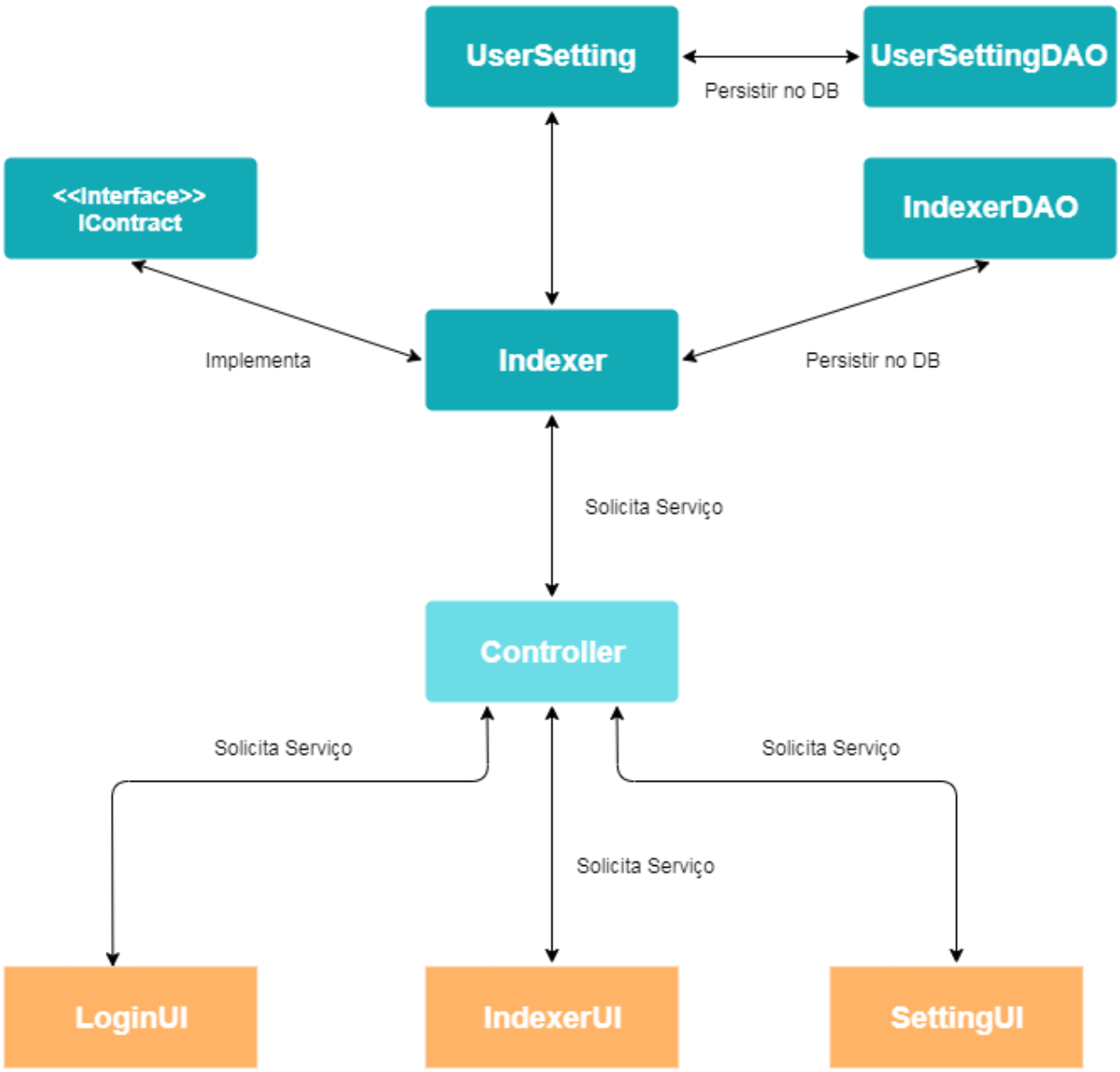
Já a coleta foi e continuará sendo através de questionários onlines para que os usuários finais possam dar *feedbacks* rápidos e facilitar a manutenção das partes do softwares comprometidas ou com bugs. Como se trata de um projeto *Open Source* também serão utilizados as *Issues* para criar novas *features* e corrigir bugs mais específicos detectados apenas por programadores.

Como já dito o levantamento de requisitos foi e continuará sendo feito atraveś de questionários online, pois a localização dos jogadores se encontra espalhada pelo mundo todo, impedindo assim de se ter reuniões

presenciais. Todavia, os questionários deverão ser refeitos constantemente para se adequar ao momento do desenvolvimento.

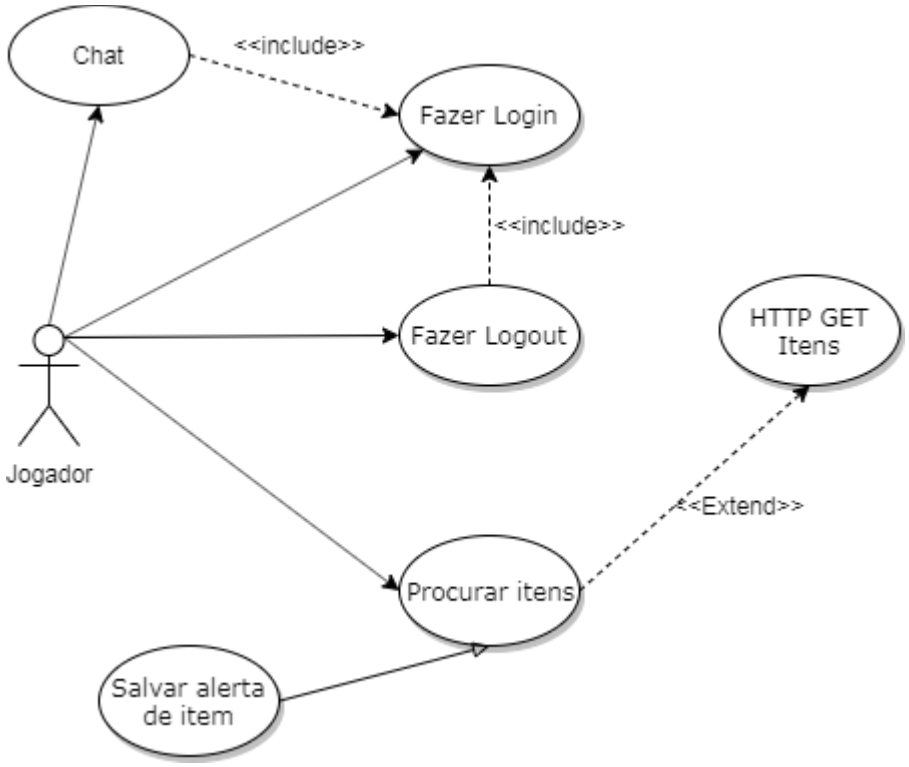
5. Projeto de software

5.1 Diagrama de Entidade Relacionamento



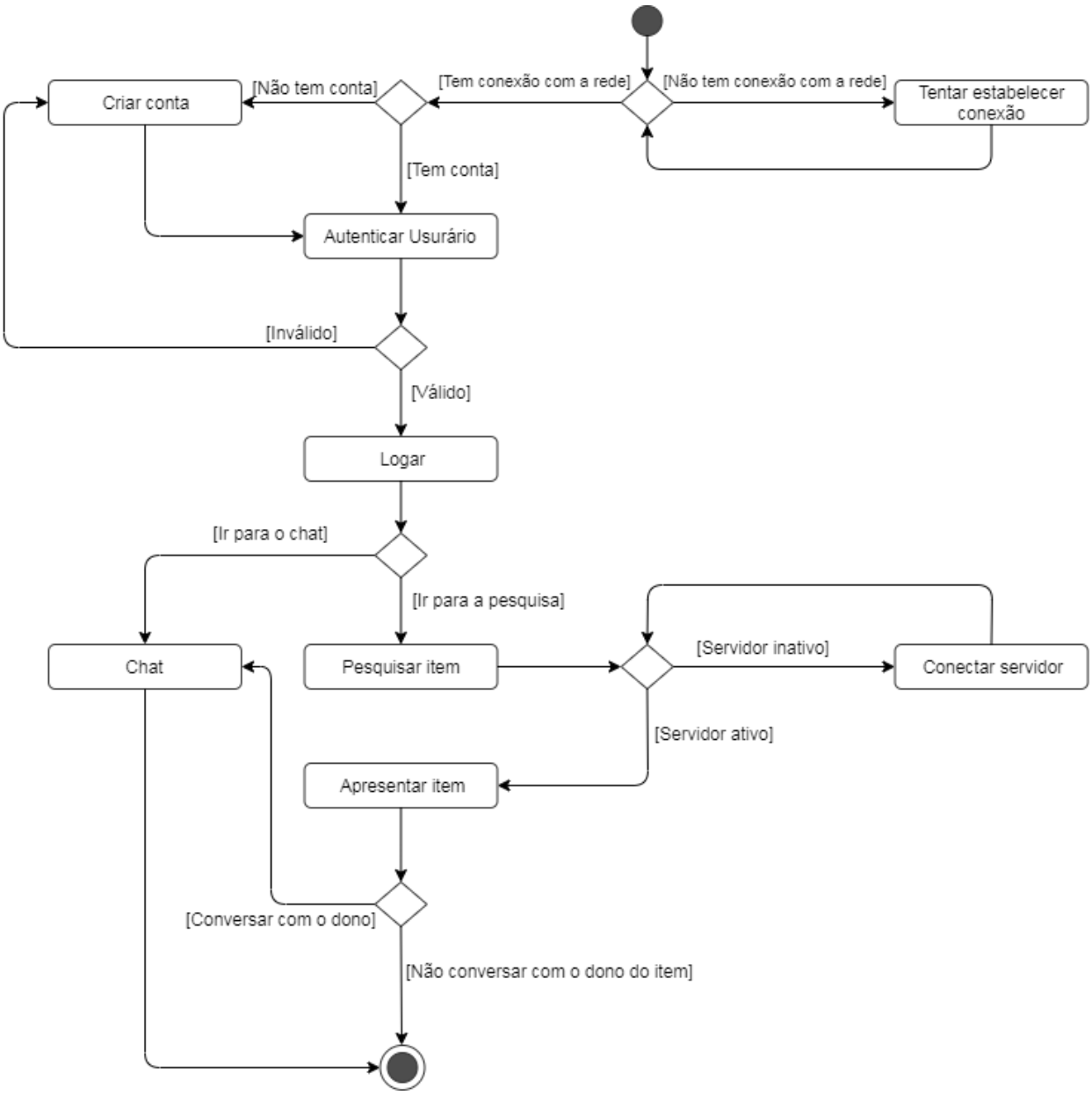
O *software* é dividido em três camadas: A de Visualização, Controladora, e os modelos de negócio. As classes de visualização que contém a interface com o usuário, solicitam o serviço de indexar a classe *Indexer* através do *Controller.* A classe *Indexer* implementa a *interface IContract*, onde estão os métodos que utilizam o serviço *HTTP. Indexer* irá tratar o *jason* recebido pelo serviço *HTTP* de acordo com as preferências do usuário da classe *UserSetting* que realiza persistência de dados através da *UserSettingDAO.* A classe Indexer salva as informações indexadas no banco de dados através da classe *IndexerDAO,* e retorna a informação para camada de visualização pelo *Controller.*

5.2 Diagrama de Casos de Uso



Com o indexador de Stash Path of Exile, o jogador poderá realizar as ações de: Fazer login e logout, pesquisar pelos itens que tem interesse, aonde o software usará o protocolo HTTP para buscar no servidor do jogo, além de ter a opção de salvar um alerta dos itens que procura, sendo notificado quando o tal item for encontrado no jogo, assim como terá acesso a um chat com os demais jogadores.

5.3 Diagrama de Atividades



Para concluir um processo de pesquisa no Indexador de Stash Path of Exile, são realizadas as seguintes atividades: testar conectividade com a rede, realizar o login do usuário ou criar conta, autenticar usuário, pesquisar item no servidor, se encontrar o item, apresenta lo, e mostrar o dono do baú em que se encontra o item, conversar com donos de itens e finalizar o processo.

6. Evolução de software

Baseado nas leis de Lehman, e sabendo que desde que se possui um projeto de desenvolvimento de software este estará suscetível a modificações e melhorias, será proposta mas não obrigatoriamente desenvolvidas algumas possíveis mudanças que poderão vir a ocorrer se houver necessidade para aperfeiçoamento das funcionalidades deste sistema.

A primeira mudança que poderá vir a ocorrer seria se as interfaces para acessar os itens de cada jogador pudessem ser realizadas via API, permitindo assim que o sistema além de expor e apresentar itens para que possam ser trocadas um a um pelos próprios jogadores, agora venha a ser realizado pelo sistema desde que houvesse a intenção mútua entre jogadores para trocar ou adquirir os itens.

A segunda mudança que obrigatoriamente deve ser feita é na implementação da interface gráfica para realizar as consultas. Isso se deve por que os usuários necessitam que sua experiência em utilizar tal sistema precisa ser agradável mas ao mesmo tempo não seja monótona sempre com as mesmas animações, cores, design etc.

A terceira mudança e mais ambiciosa será a de expandir a aplicação de comercialização de itens para outros jogos, permitindo assim a plataforma abranger diferentes estilos captando mais usuários.

7. Manutenção de software

Nesta seção serão tratadas as manutenções de software, pois partindo da premissa que foi desenvolvido por humanos ele conterá erros passíveis de serem corrigidos, prevenidos, adaptados ao meio ou melhorado.

7.1 Manutenção corretiva

O caso de uso para esta manutenção será descrito pela situação decorrida da mudança em uma dependência que realizava as consultas na API e as escreviam no banco de dados.

No dia 14 de outubro de 2019 a dependência “mongoose”, utilizada como DRM(Mapeamento de Documento Relacional) apresentou uma falha de segurança em seus procedimentos obrigando a ser modificada para sua versão *current* que apesar de ser beta possui a correção desta falha.

7.2 Manutenção preventiva

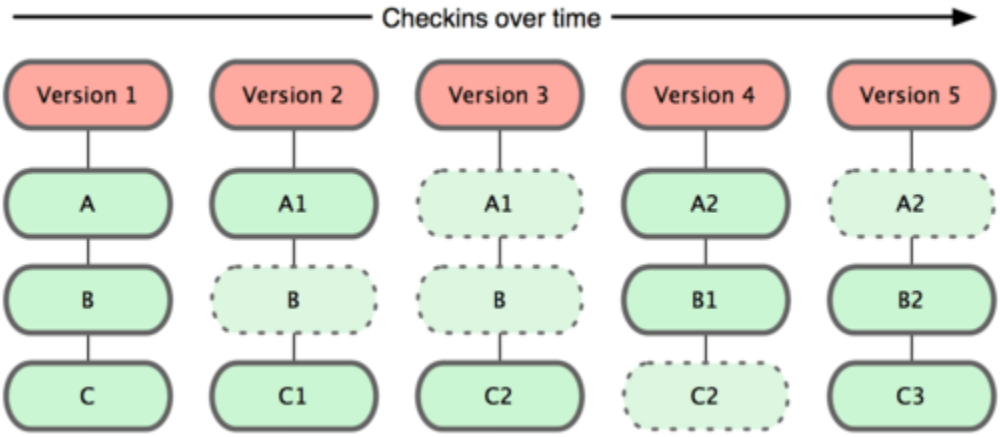
A fim de evitar falhas de segurança com ferramentas imaturas no mercado foi realizada a mudança tanto do ambiente de armazenamento quanto da interface para acessar os dados. Isto é, houve a mudança de um banco de dados não relacional compatível nativamente com os dados porém com interfaces de acesso inseguras para um tipo relacional que apesar de necessitar de alterações possui melhores políticas de segurança e processos para armazenamento de transações.

7.3 Manutenção adaptativa

Como foi necessário a mudança para um ambiente de banco de dados incompatível nativamente com os dados obtidos da API, então foi necessário que se desenvolve-se um sub-módulo para tratar os dados fornecidos em forma de documentos JSON (Java Script Object Notation) transformando estes dados em tabelas normalizadas relacionais.

8. Controle de versão

O Sistema de Controle de Versão distribuído escolhido foi o GIT, entre suas principais características estão, velocidade, design simples, suporte robusto a desenvolvimento e capacidade de lidar eficientemente com grandes projetos. Ao contrário de sistemas de controle de versão centralizados, a natureza distribuída do GIT permite ser muito mais flexível na forma para a comunidade de desenvolvedores, já que o software se trata de uma iniciativa open source, tendo o feedback mais rapido dos usuarios.
O GIT vai ser utilizado para registrar o histórico de edições de qualquer tipo de arquivo, ou seja, cada alteração que for realizado no arquivo pela comunidade de desenvolvedores ficará registrado de forma que se necessário, será possível identificar quem alterou o arquivo, assim como, recuperar a específica versão ou ajudar na manutenção da mesma mais facilmente.
Tratamento dos dados GIT:



Há três seções principais de um projeto do GIT: o diretório do GIT, o diretório de trabalho e a área de preparação.Desse modo, pode-se verificar que os principais comandos do GIT estão relacionados ao seu fluxo básico de trabalho que pode ser definido com as operações comuns realizadas por um usuário desenvolvedor.

9. Conclusão

Aprendemos com esse projeto que *o* campo de estudo muito abrangente e que para projetar um software deve se ter conhecimento não apenas restrito ao desenvolvimento, mas de diversos elementos que circundam o produto a ser entregue, temos que entender de legislação, de padrões empresariais, segurança, de usuário.

Aprendemos que existe modelos de desenvolvimento de *software* que atendem a cada necessidade e de acordo com os recursos disponíveis, sejam eles de tempo, de mão de obra, de qualidade e etc.

Um documento bem feito, implica em um processo de desenvolvimento de software mais eficiente, rápido, com custo menor, mais seguro, com facilidade de manutenção, e alinhado com a necessidade do cliente e seus usuários.

REFERÊNCIAS

PRESSMAN, Roger S. **Engenharia de Software: uma abordagem profissional**. 7. ed. São Paulo: Pearson Makron Books, 2011.