Alunos: Isaac Santiago,João Gabriel Alves, Marcus Vinícius Ribeiro A, Paloma Bárbara Pinto, Riquelme Damião Silva.

Curso: Sistemas de Informação - 3° período

Disciplina: Engenharia de Software I

Projeto CodeHub

Atualmente existe uma constante preocupação em relação a otimização de tempo e recurso na execução de projetos, necessitando de um gerenciamento estratégico e utilização de metodologias que auxiliem na dinâmica da equipe e relacionamento com o cliente.

Diante desse cenário, a documentação retrata uma descrição sucinta do projeto GitHub, ao qual, estará fornecendo uma oportunidade de interação e desenvolvimento de habilidades de alunos e/ou entusiastas na área de programação. Por fim, o presente projeto se desenvolve com a metodologia Scrum, utilizando de metodologias ágeis, que serão de auxílio no gerenciamento dos processos e etapas do projeto.

1. Product Backlog

CODEHUB BACKLOG

ID	Nível hierárquico	Funcionalidade	Descrição	Prioridade	Sprint	Status	Legenda
1	Visitante	Criar uma conta	Cria uma conta a partir dos dados fornecidos no processo, salvando os mesmos no banco de dados	Alta	1	Riquelme	Concluído
2	Usuário	Fazer Login	Entra no sistema do CodeHub, salvando as informações da sessão utilizando Cookies	Alta	1	João	Aguardando aprovação
3	Usuário	Fazer Logout	O usuário desconecta-se do sistema, removendo dados da sessão; dados inseridos pelo método login.	Alta	1	Marcus	Bloqueada
4	Usuário	Iniciar novo repositório	Cria o repositório .CodeHub dentro do projeto, permitindo assim que o usuário utilize os demais comandos do CH.	Alta	1	Isaac	Pendente
5	Visitante	Exibir comando de ajuda	Orienta usuários quanto as funcionalidades e exibe os comandos disponíveis na aplicação.	Alta	1	Paloma	Em Desenvolvimento
6	Usuário	Adicionar ao Container	Salva os paths dos arquivos apontados pelou usuário em um compartimento denominado de container	Alta	2	Riquelme	Em teste
7	Usuário	Remover do Container	Modifica o container removendo o path do arquivo que o usuário apontar e reescreve o container.	Alta	2	Paloma	
8	Usuário	Exibir histórico de versões	Lista todo o histórico de versões do projeto ordenado por datas (partindo da mais antiga para a mais recente)	Alta	2	Isaac	
9	Usuário	Criar nova versão	Permite salvar o código em um diretório de versão, associando um hash a este versionamento. Após isso, o container deverá estar vazio e as mudanças efetivadas	Alta	2	Marcus	
10	Usuário	Voltar para versão	Retorna a uma versão determinada: deletando a atual versão do diretório do projeto e trazendo a escolhida de volta	Alta	2	João	
11	Usuário	Remover versão	Deleta a versão informada pelo usuário	Alta	3	João	
12	Usuário	Exibir Container	Lista todos os itens contidos no container	Média	3	Paloma	
13	Usuário	Backup versão removida	Após determinada versão ser removida, ela deve ser salva para permitir sua restauração.	Média	3	Isaac	
14	Usuário	Restaurar versão removida	Recupera a última versão apagada, do backup feito.	Média	3	Riquelme	
15	Usuário	Enviar para nuvem	Permite que seja adicionado, a um repositorio remoto, os conteúdos do projeto, além de, é claro, funcionar como backup	Baixa	3	Marcus	

2. Sprint Backlog

Durante o planejamento, definiu-se uma sequência de ações a serem consideradas, já cientes da existência de possibilidade de serem alteradas durante o desenvolvimento do projeto.

ANÁLISE DE REQUISITOS SPRINT 3	
Funcionais	**Não funcionais**
Remover versão	Os arquivos da versão são excluidos sem interferir nas demais versões
Exibir Container	O sistema exibe uma lista com todos os itens a versionar diferenciando por diretórios e arquivos
Backup versão removida	O sistema armazena como backup somente uma única versão por vez, sendo ela a última versão removida
Restaurar versão removida	O sistema restaura a versão removida disponível no backup sem nenhuma perda de dados
Enviar para nuvem	O sistema garante o envio dos dados para o github, com isso, esses arquivos estarão, a qualquer momento, disponíveis.

Sprint 2:

Tempo estimado: 15 dias

Tempo decorrido: 10 dias

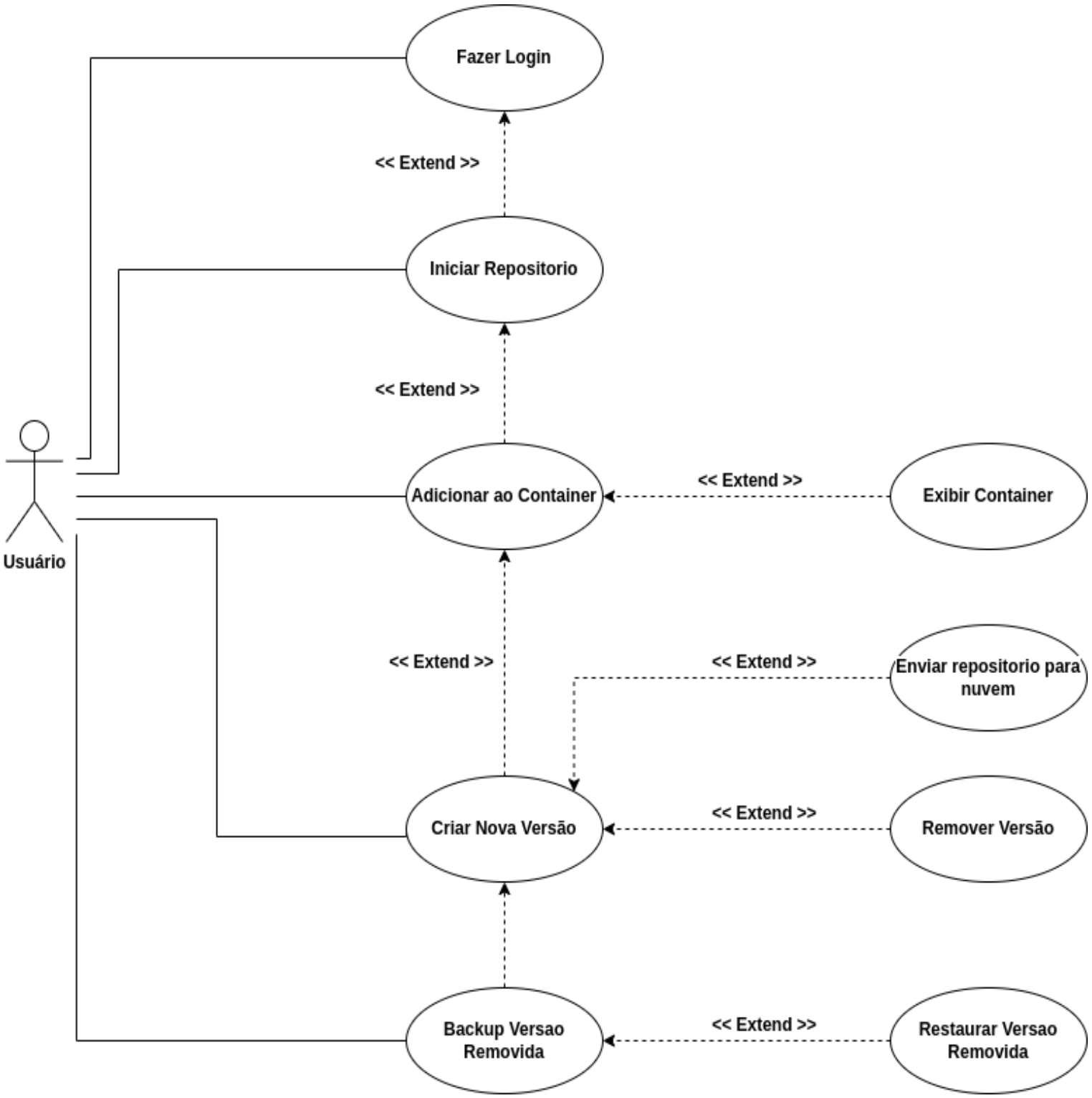
Entrega: 22/08/22

Relator: Professora Luciana

Funcionalidade	Prioridade	Responsável
Remover versão	**Alta**	João
Exibir Container	**Média**	Paloma
Backup versão removida	**Média**	Isaac
Restaurar versão removida	**Média**	Riquelme
Enviar para nuvem	**Baixa**	Marcus

3. Caso de uso e cenário de caso de uso

3.1. Diagrama de casos de uso:



3.2. Cenários de caso de uso:

Cenário 11:

Nome do Cenário: Remover versão

Ator: Usuário

Pré-condição: 09. Criar nova versão.

Fluxo normal:

1. Usuário digita o comando: “--apagar -f”, passando como parâmetro a *hash* da versão que ele deseja apagar;
2. Sistema verifica se a *hash* existe;
3. O caminho dos arquivos da *hash*, versão selecionada, são obtidos no banco de dados;
4. Os arquivos da hash são deletados;
5. Exibe a mensagem : “Versão deletada com sucesso”;

Fluxos alternativos:

Hash da versão não existe no banco:

1. Exibe a mensagem: “A hash inserida não existe no banco.”

Pós-condição: A versão selecionada é deletada.

Cenário 12:

Nome do Cenário: Exibir Container

Ator: Usuário

Pré-condição: 06. Adicionar ao Container

Fluxo normal:

1. Usuário insere o comando “--container”;
2. O sistema verifica a existência de itens no container;
3. Sistema percorre a listagem de *paths* dos arquivos e/ou diretórios que estão disponíveis no container (arquivo Json presente no repositório ‘.CodeHub’);
4. Exibe as seguintes mensagens:
   "O container atual contém os seguintes itens:"
   "[Arquivos]"

5. Sistema exibe a lista de *paths* do tipo arquivo armazenados no container;

6. Exibe a seguinte mensagem:
   "[Diretorios]"

7. Sistema exibe a lista de *paths* do tipo diretório armazenados no container.

Fluxos alternativos:

Não existem *paths* disponíveis dentro do repositório container:

1. O sistema exibe a seguinte mensagem:
   "O container atual não possui itens!"

Pós-condição: A lista de *paths* disponíveis no container são exibidas para o usuário.

<u>Cenário 13:</u>

Nome do Cenário: Backup versão removida

Ator: Usuário

Pré-condição: 09. Criar nova versão.

Fluxo normal:

1. Usuário digita o comando: “--apagar <*hashVersao*>”, no qual a <*hashVersao*> indica a versão específica a ser apagada;
2. Sistema verifica se a <*hashVersao*> existe;
3. Sistema apaga o backup existente no repositório;
4. Sistema cria um novo backup para versão removida;

Fluxos alternativos:

A versão informada não existe

1. O Sistema exibe: “A versao informada nao existe”

Pós-condição: A pasta da versão específica é removida do diretório de versões e é movida para a lixeira, onde fica armazenada como forma de backup.

Cenário 14:

Nome do Cenário: Restaurar versão removida

Ator: Usuário

Pré-condição: 13. Backup versão removida

Fluxo normal:

1. Usuário digita o comando: “--restaurar”;
2. Sistema verifica se existe um backup de uma versão removida;
3. Sistema restaura o backup da versão removida, recompondo o histórico das versões do repositório;
4. Sistema exibe a mensagem para o usuário: “Exibe a mensagem para o usuário: “Versao de hash " + hash da versão no backup + " restaurada.”

Fluxos alternativos:

Não Existe um backup de versão removida:

1. Sistema exibe a mensagem para o usuário:
   “Não existe um backup na base de dados.”

Pós-condição: Último Backup de uma versão removida é restaurada no repositório.

Cenário 15:

Nome do Cenário: Enviar para a nuvem

Ator: Usuário

Pré-condição: 09. Criar versão

Fluxo normal:

1. Usuário digita “--enviar”
2. É enviado os arquivos que foram versionados para a nuvem.

Fluxos alternativos:

1. Não existe versão.
2. Sistema exibe mensagem para o usuário
   “ERRO não versionado”
3. Pode-se também, ter os seguinte erro:
   “error: failed to push some refs to '”

Pós-condição: Os arquivos são enviados para o repositório em nuvem.