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	Objetivo	Prioridade	Sprint	Status	Legenda
1	Visitante	Criar uma conta	Usufruir do sistema	Alta	1	Riquelme	Concluído
2	Usuário	Fazer Login	Autenticação	Alta	1	João	Aguardando aprovação
3	Usuário	Fazer Logout	Sair do sistema	Alta	1	Marcus	Bloqueada
4	Usuário	Iniciar novo repositório	Iniciar área de trabalho	Alta	1	Isaac	Pendente
5	Visitante	Exibir comando de ajuda	Orientar usuários quanto as funcionalidades	Alta	1	Paloma	Em Desenvolvimento
6	Usuário	Adicionar ao Container	Permitir confirmações e versionamento	Alta	2	Riquelme	Em teste
7	Usuário	Remover do Container	Não efetuar versionamento	Alta	2	Paloma	
8	Usuário	Exibir histórico de versões	Lista histórico de comentários e versões	Alta	2	Isaac	
9	Usuário	Criar nova versão	Versionar o código, evidênciar mudanças	Alta	2	Marcus	
10	Usuário	Voltar para versão	Retornar a versão anterior, recuperar	Alta	2	João	
11	Usuário	Remover versão	Cancelar versionamento realizado	Alta	3	João	
12	Usuário	Exibir Container	Listar itens contidos no container	Média	3	Paloma	
13	Usuário	Backup versão removida	Permitir acesso a contextos apagados	Média	3	Isaac	
14	Usuário	Restaurar versão removida	Recupera a última versão disponível	Média	3	Riquelme	
15	Administrador	Adicionar outros usuários	Permitir trabalho colaborativo	Baixa	3	Marcus	
16	Administrador	Enviar para nuvem	Salvar conteúdo em um Repósitorio Remoto	Baixa			
17	Usuário	Baixar da nuvem	Get dos dados para o repositório local	Baixa			
18							
19							
20							

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 2	
Funcionais	**Não funcionais**
Adicionar ao Container	O sistema deve gerenciar tanto o diretório principal do projeto quanto as suas subpastas
Remover do Container	O sistema deve ter uma paleta de cores intuitiva
Criar nova versão	O arquivo que constitui o container deve ser em formato JSON
Exibir histórico de versões	O sistema deve indicar a versão atual no histórico de versões
Voltar para versão	O sistema deve exibir as informações das versões de maneira ordenada

Sprint 2:

Tempo estimado: 15 dias

Tempo decorrido: 11 dias

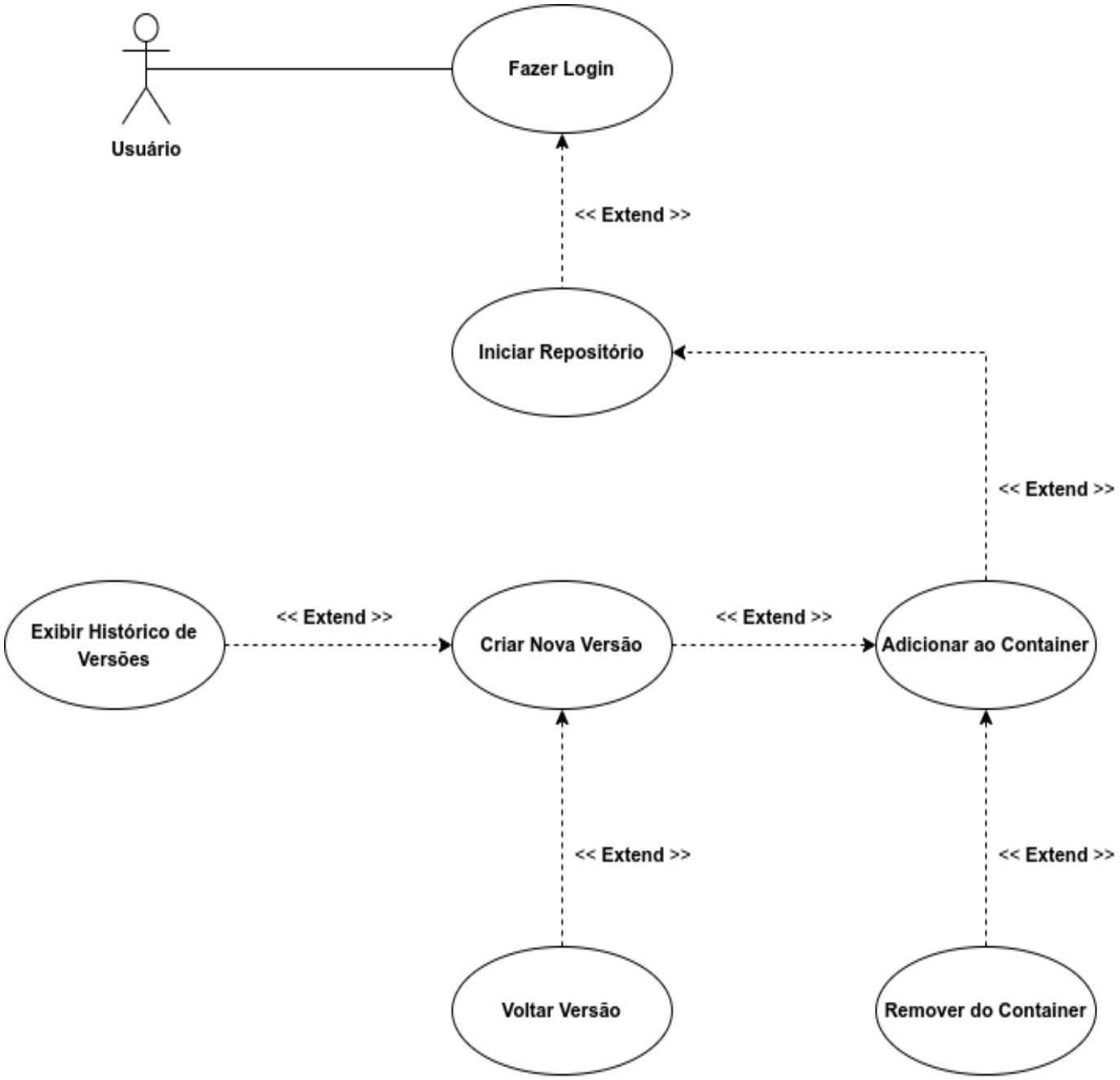
Entrega: 11/07/2022

Relator: Professora Luciana

Funcionalidade	Prioridade	Responsável
Adicionar ao Container	Alta	Riquelme
Remover do Container	Alta	Paloma
Exibir Histórico de Versões	Alta	Isaac
Criar Nova Versão	Alta	Marcus Vinícius
Voltar para versão	Alta	João Gabriel

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 06:

Nome do Cenário: Adicionar ao Container

Ator: Usuário

Pré-condição: 04. Iniciar repositório

Fluxo normal:

1. Usuário insere o comando “--adicionar”, seguido do nome de um arquivo que esteja presente na pasta do projeto a ser versionado, ou a flag “**.**”, que significa que **todos** os arquivos do projeto devem ser selecionados;
2. O sistema verifica se o arquivo informado pelo usuário existe na pasta do projeto a ser versionado, assim selecionando-o;
3. O sistema salva os *paths* dos arquivos selecionados no container (arquivo Json presente no repositório ‘.CodeHub’ na pasta do projeto a ser versionado);

Fluxos alternativos:

Arquivo informado pelo usuário não está presente na pasta do projeto a ser versionado:

1. O sistema não salva nenhum *path* de arquivo no container (arquivo Json presente no repositório ‘.CodeHub’ na pasta do projeto a ser versionado).

Pós-condição: Arquivos adicionados ao container prontos para serem versionados.

Cenário 07:

Nome do Cenário: Remover do Container

Ator: Usuário

Pré-condição: 06. Adicionar ao Container

Fluxo normal:

1. Usuário insere o comando “--remover” seguido do nome do arquivo que deseja remover do container. O usuário também tem a possibilidade de digitar a flag “**.**” para remover **todos** os arquivos;

2. Sistema percorre a listagem de *paths* dos arquivos que estão disponíveis no container (arquivo Json presente no repositório '.CodeHub' na pasta do projeto a ser versionado), removendo o que foi solicitado pelo usuário;
3. Sistema reescreve os *paths* armazenados no container com as devidas remoções.

Fluxos alternativos:

Arquivo informado pelo usuário não está disponível dentro do repositório container:

1. O sistema não remove nenhum *path* de arquivo no container.

Pós-condição: Arquivos indesejados foram retirados do container.

<u>Cenário 08:</u>

Nome do Cenário: Exibir histórico de versões

Ator: Usuário

Pré-condição: 09. Criar nova versão.

Fluxo normal:

1. Para exibir o histórico, o usuário deve inserir o comando "--historico". Assim, o sistema acessa o diretório 'versoes' do projeto atual, buscando lista todas as versões geradas;
2. O sistema ordena as versões por datas, partindo da mais antiga para a mais recente;

Fluxos alternativos:

Repositório não possui versões:

1. Exibe a mensagem para o usuário: "O repositorio atual nao possui versoes."

Pós-condição:

Para cada versão, o sistema exibe as seguintes informações:

"Versao: <hashVersão>"

"Comentario: <comentarioVersao>"

"Autor: <nomeAutor> <emailAutor>"

"Data: <dataVersão>"

Caso a versão seja a versão atual, o sistema exibe as informações da seguinte maneira:

"Versao: <hashVersão> [<versaoAtual>]"

"Comentario: <comentarioVersao>"

"Autor: <nomeAutor> <emailAutor>"

"Data: <dataVersão>".

Cenário 09:

Nome do Cenário: Criar nova versão

Ator: Usuário

Pré-condição: 06. Adicionar ao Container

Fluxo normal:

1. Usuário insere o comando "--versionar";
2. Com isso, é criado uma thread para garantir execução sem falhas e, se possível, otimizar a tarefa;
3. Além disso, é criado uma pasta de tabela de versões no banco de dados para conter a versão recém criada;
4. O conteúdo do repositório é lido, a fim de identificar os paths, pois, esses *paths* devem ser usados para mapear os arquivos e, posteriormente, salvar no banco de dados, como supracitado;
5. É criado um *hash numérico aleatório para evitar nomes iguais e, desse modo, evitar conflitos*;
6. Esse *hash* é utilizado como nome da pasta de destino;
7. É copiado, para o diretório de versões, os itens que estão contidos os *paths os* identificados, efetuando as mudanças;
8. É escrito, no banco de dados, a versão criada com informações como o nome do hash, o path, o usuário que adicionou, e a data.

Fluxos alternativos:

O repositório '.CodeHub' está vazio:

1. Não será efetuado o versionamento.

Pós-condição: É criado uma versão dos arquivos, salvando os dados essenciais da versão (data, autor, comentário) no banco de dados, e, se necessário, disponibiliza todo o amparo para retorno nesta versão.

<u>Cenário 10:</u>

Nome do Cenário: Voltar versão

Ator: Usuário

Pré-condição: 09. Criar nova versão

Fluxo normal:

1. Usuário digita o comando: “--voltar”, passando como parâmetro a *hash* da versão que ele deseja voltar;
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 atuais são deletados, com exceção de ‘.CodeHub’, e seus sub-diretórios;
5. Os arquivos da *hash*, versão selecionada, são copiados para a pasta do projeto;
6. O Sistema altera a *flag* “Versão atual” para true no objeto da *hash*;
7. O Sistema retira a *flag* “Versão atual” da versão anterior;
8. O Sistema salva os objetos no banco novamente com as alterações.

Fluxos alternativos:

Hash da versão não existe no banco:

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

Pós-condição: A versão anterior selecionada é usada como versão atual do projeto.