Aluno: Ariel Mileguir
Matricula: 1811928

Prova P2 de ProgramaÃ§Ã£o modular

QuestÃ£o 1:

Considere a seguinte situaÃ§Ã£o hipotÃ©tica:

Uma determinada equipe de desenvolvedores Ã© contradada para desenvolver uma aplicaÃ§Ã£o.

Detalhes da AplicaÃ§Ã£o (Escopo): Um sistema que seja capaz de gerenciar alunos em um a universidade, isto Ã©: Mantenha registro de todos os membros da universidade, alunos e pr ofessores (nome, sobrenome, CPF, e EndereÃ§o de moradia), e as disciplinas que os alunos es tÃ£o matriculadas (nome, turma, professor). Para ele foi criado um mÃ³dulo de criaÃ§Ã£o de membros da universidade, um modulo de criaÃ§Ã£o de disciplinas, e um mÃ³dulo com as funcion alidades necessÃ¡rias para persistencia e posteriormente recuperaÃ§Ã£o (retrieve / get) de dados guardados em um banco de dados. Vale notar tambÃ©m que apesar dos membros universitÃ¡ rios possuirem uma matricula universitaria Ãºnica, isso nÃ£o foi mencionada em momento nenh um durante as reuniÃµes entre o cliente e os desenvolvedores.

Os membros universitÃ¡rios apesar de serem identificados pela tupla descrita acima, sÃ£o recuperados sem ambiguidade do banco de dados pela tupla: (CPF, endereÃ§o).
ApÃ³s ser aprovado pelo cliente o documento contendo a especificaÃ§Ã£o da aplicaÃ§Ã £o detalhando as suas "regras de negÃ³cio" os desenvolvedores comeÃ§am a de fato programar tal aplicaÃ§Ã£o seguindo rigidamente uma documentaÃ§Ã£o de requisitos, sendo esta criada te ndo como base o entendemento destas regras de negÃ³cio.

Os trÃªs mÃ³dulos entÃ£o sÃ£o programados e testados com testes unitÃ¡rios com suce sso gerando a primeira versÃ£o da aplicaÃ§Ã£o.

ApÃ³s o cliente "C" testar a aplicaÃ§Ã£o e verificar que as disciplinas podem ser r egistradas fÃ¡cilmente, e sem falhas, que os membros universitÃ¡rios podem ser registrados normalmente, e que todas essas informaÃ§Ãµes podem ser recuperados do banco de dados, o cli ente C relata que:

Ele quer que seja implementada uma maneira mais simples de recuperar os dados refer entes aos membros universitÃ¡rios. Apesar disso C diz querer colocar essa versÃ£o em produÃ §Ã£o, e que enquanto esta versÃ£o estÃ¡ sendo usada normalmente, as evoluÃ§Ãµes devem ser d esenvolvidas.

Os desenvolvedores deverÃ£o realizar modificaÃ§Ãµes nela para que posteriormente qu ando as modificaÃ§Ãµes forem feitas, uma nova versÃ£o possa ser criada, e ela possa ser col ocada para uso.

Ã\211 decidido entÃ£o com base nisso que como serÃ¡ necessÃ¡rio usar um dado Ãºnico para cada membro universitÃ¡rio (como um ID) para que atravÃ©z dele sozinho, seja possÃ-ve l recuperar dados de qualquer membro, e que serÃ¡ possÃ-vel recuperar os dados dos membros universitÃ¡rios APENAS colocando no programa responsÃ¡vel por isso esta informaÃ§Ã£o (Prime ira manutenÃ§Ã£o evolutiva).

C informa entÃ£o que cada membro da universidade possuÃ- uma matricula Ãºnica. apÃ³ s os desenvolvedores perguntarem se existe algo semelhante a isso nesta universidade.

A matricula universitÃ¡ria Ã© escolhida para fazer este papel, e nota-se que serÃ¡ necessÃ¡rio a realizaÃ§Ã£o de algumas modificaÃ§Ãµes nos cÃ³digos da aplicaÃ§Ã£o para que s eja possÃ-vel registrar um aluno com a sua matricula (Segunda manutenÃ§Ã£o evolutiva).

Posteriormente, C recebe o relato de que certos nomes de membros universitarios est Ã£o sendo expostos incorretamente: Nomes com caracteres estrangeiros apesar de serem regist rados normalmente, aparecem "corrompidos" quando sÃ£o recuperados. Uma manutenÃ§Ã£o emergen cial terÃ¡ de ser realizada na aplicaÃ§Ã£o que jÃ¡ estÃ¡ em uso.

A equipe entÃ£o parte para realizar a manutenÃ§Ã£o de emergencia. Uma nova versÃ£o Ã© criada, e testada unitariamente dois dias depois contendo a correÃ§Ã£o do bug do nome, f oi mudada a codificaÃ§Ã£o de caracteres no banco de dados de ASCII para UTF-8. (ManutenÃ§Ã£ o corretiva)

C testa entÃ£o na nova versÃ£o o recurso de registrar membros com nomes que contenh am caracteres estrangeiros, e atesta que ela estÃ¡ funcionando conforme o desejado. Ela Ã© colocada em produÃ§Ã£o, isto Ã©: Ela substituÃ- a versÃ£o anterior com bugs que estava send o usada.

Ao longo de alguns dias, muitas modificaÃ§Ãµes sÃ£o feitas nos cÃ³digos da versÃ£o recem "congelada" da aplicaÃ§Ã£o afim de confortar as outras mudanÃ§as requisitadas por C. Uma nova versÃ£o Ã© concluÃ-da, e testada com sucesso.

Primeiro foi feita a segunda manutenÃ§Ã£o evolutiva para que o processo de realizar a primeira manutenÃ§Ã£o evolutiva fosse mais claro e rÃ¡pido, depois foi feita a primeira manutenÃ§Ã£o evolutiva.

A equipe realiza muitos testes na versÃ£o nova (como um todo) e concluÃ- que todas as queixas e pedidos que C relatou foram resolvidos e implementados com sucesso, e que a ap licaÃ§Ã£o estÃ¡ funcionando adequadamente em funÃ§Ã£o do que foi proposto.

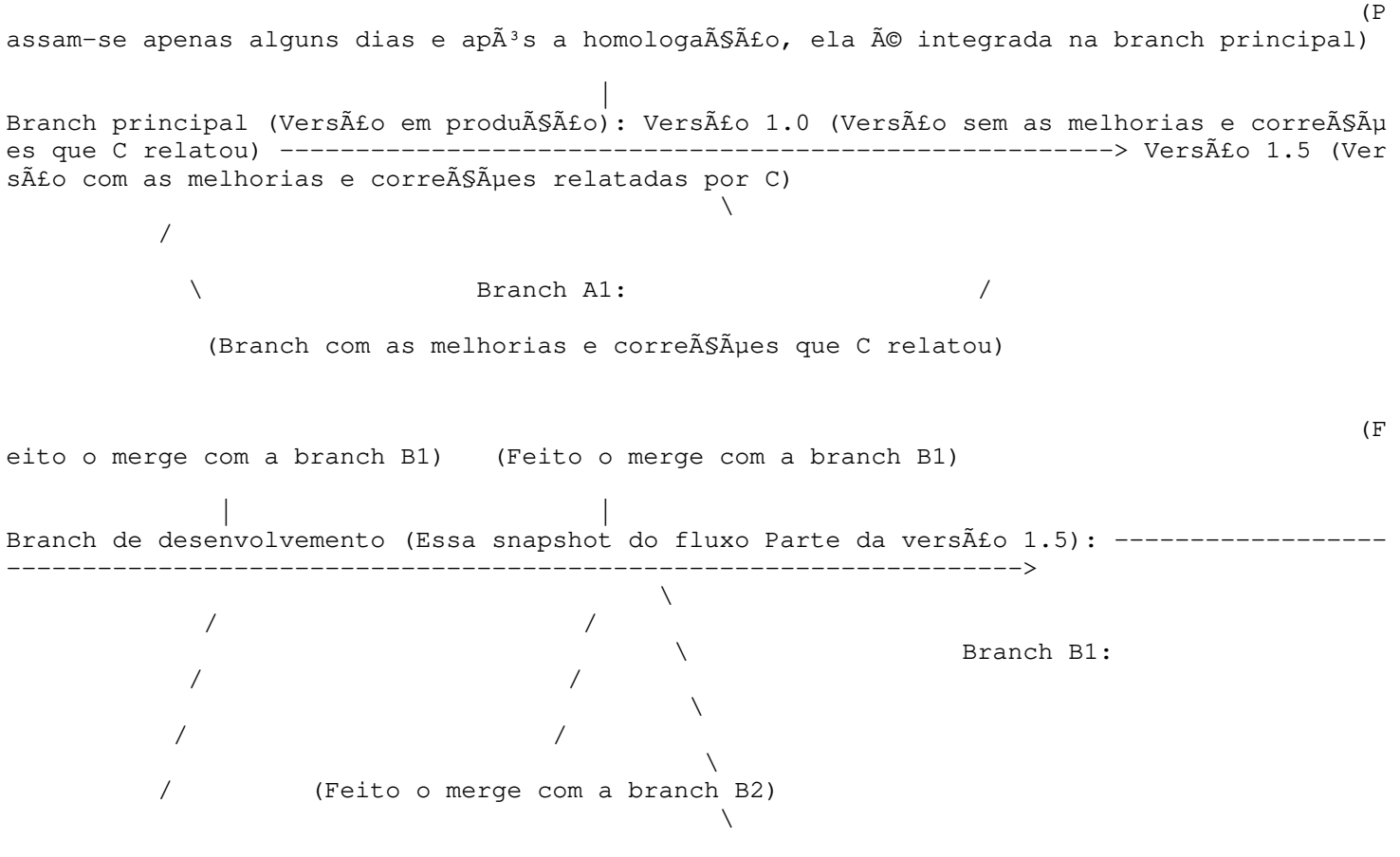
C entÃ£o testa novamente a aplicaÃ§Ã£o de forma semelhante como fez antes, chega na mesma conclusÃ£o, e aprova essa Ãºltima versÃ£o para ser colocada em produÃ§Ã£o.

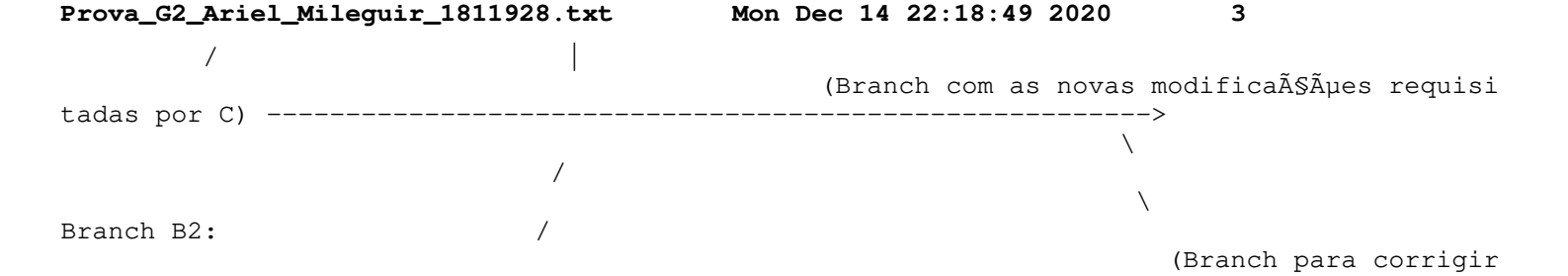
Com base nas modificaÃ§Ãµes feitas pela manutenÃ§Ã£o de correÃ§Ã£o, membros que ten ham nomes com caracteres estrangeiros podem agora utilizar essa aplicaÃ§Ã£o para se cadastr arem normalmente, e os dados de todos os membros universitÃ¡rios podem ser facilmente recup erados atravez de suas matriculas.

QuestÃ£o 2:

ApÃ³s a aplicaÃ§Ã£o descrita na questÃ£o acima ter sido colocada em uma versÃ£o est Ã¡vel, homologada e autenticada por C, C faz uma nova requisiÃ§Ã£o para a equipe durante o perÃ-odo de recesso universitÃ¡rio (fÃ©rias): Ele gostaria que fossem implementadas funcion alidades que impedissem membros os universitarios de estarem cadastrados simultaneamente em mais de sete disciplinas, e que apÃ³s o semestre (perÃ-odo de aulas) acabar, que os membro s universitÃ¡rios fossem automaticamente descadastrados das disciplinas.

Tem-se entÃ£o o seguinte fluxo no software de controle de versionamento:





QuestÃ£o 3:

Partindo do pressuposto de que trata-se de uma linguagem de programaÃ§Ã£o aonde a indentaÃ§Ã£o nÃ£o afeta a leitura do cÃ³digo pela maquina, modificar a indentaÃ§Ã£o pode se tratar de uma refatoraÃ§Ã£o.

Uma refatoraÃ§Ã£o Ã© definida como uma alteraÃ§Ã£o feita no cÃ³digo sem que se modifique de NENHUMA forma o seu funcionamento ou comportamento com a intenÃ§Ã£o de que a qualidade do cÃ³digo seja melhorada para futuras manutenÃ§Ãµes, ou leitura e entendimento humano do cÃ³digo.

Modificar as indentaÃ§Ãµes de um cÃ³digo escrito em uma linguagem destas pode se tratar de uma refatoraÃ§Ã£o, caso o objetivo seja de melhorar a visibilidade ou leitura do cÃ³digo para os fins definidos acima.

QuestÃ£o 4:

Partindo do pressuposto de que cada modificaÃ§Ã£o bem sucedida (testada) em um cÃ³digo deve ser imediatamente integrada com o flxuo principal de projeto (branch main) para que sejam evitados riscos de conflitos, ou merges de alto risco, o processo de refatoraÃ§Ã£o assim como qualquer outra mudanÃ§a substancial deve ser feita ANTES de um merge.

Caso o projeto em questÃ£o tenha um numero consideravel de desenvolvedores trabalhando nele, e que parte desses desenvolvedores trabalhem ou nÃ£o em um mesmo arquivo (que estÃ¡ sendo refatorado), Ã© mais indicado realizar um merge manual para minimizar o risco de conflitos ou merges "bem sucedidos", porÃ©m que causam consequencias indesejadas e que podem atÃ© quebrar a aplicaÃ§Ã£o, jÃ¡ que por estar sendo feito por um algoritimo (Isto Ã©: uma maquina) que nÃ£o tem a capacidade de pensar e julgar adequadamente que tipos de modificaÃ§Ãµes devem ser implementadas, e que tipos de modificaÃ§Ãµes devem ser descartadas em um merge, o risco de algum tipo de problema relacionado ao merge acontecer neste caso Ã© naturalmente maior do que se ele estivesse sendo feito por um desenvolvedor que teoricamente, teria um nÃ-vel de entendimento consÃ-derÃ¡vel da aplicaÃ§Ã£o ou do cÃ³digo.

Caso seja garantida a hipotese de que um Ãºnico desenvolvedor trabalhe em um projeto para uma aplicaÃ§Ã£o (por exemplo, o sistema backend de uma aplicaÃ§Ã£o), o merge poderia ser feito automaticamente pois o risco de conflitos, ou casos de merge descritos acima Ã© praticamente inexistente, pois a versÃ£o do(s) arquivo(s) em que este desenvolvedor estÃ¡ refatorando poderia teoricamente sempre ser considerada a mais atualizada.

QuestÃ£o 5:

Plugins utilizados de um acervo mantido por uma equipe de desenvolvemento sÃ£o necessariamente componentes reutilizÃ¡veis internos.

Componentes reutilizaveis internos podem ser definido como cÃ³digos feitos para um domÃ-nio ou aplicaÃ§Ã£o especifica, e que graÃ§as a isso podem aumentar a produtividade de desenvolvemento de um projeto, jÃ¡ que apesar de se tratar de um domÃ-nio especifico, muitas coisas que precisariam ser implementadas para desenvolver este projeto, podem jÃ¡ estar prontas, evitando a necessidade de ter que programar elas.

Plugins podem ser definido como imports dinÃ¢micos (o uso de componentes que complementam a execuÃ§Ã£o de um programa ou aplicaÃ§Ã£o e que fazem parte do domÃ-nio da mesma a tempo de execuÃ§Ã£o) porÃ©m que nÃ£o gerem uma versÃ£o nova da aplicaÃ§Ã£o.

Neste caso, como o uso de um plugin Ã© considerado um import dinÃ¢mico, e que bibliotecas de import dinamicos (por fazerem parte do domÃ-nio especÃ-fico da aplicaÃ§Ã£o) podem ser considerados componentes reutilizaveis internos, um plugin entÃ£o pode ser considerado necessÃ¡riamente um componente reutilizÃ¡vel interno (mesmo fazendo a utilizaÃ§Ã£o de wrappers que encapsulem chamadas de mÃ³dulos reutilizÃ¡veis externos, isto Ã©: abstraÃ§Ãµes que facilitem o uso dos mesmos em diversas partes dos cÃ³digos, e o acoplamento dos mesmos com diversos mÃ³dulos).