Universidade do Minho
Escola de Engenharia
Licenciatura em Engenharia Informática
Desenvolvimento de Sistemas de Software

Sistema de gestão para centros de reparação de equipamentos eletrónicos

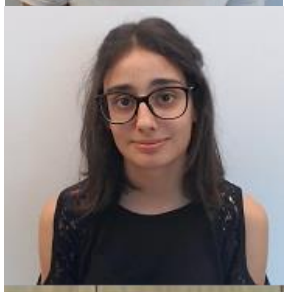
Especificação e Implementação da Aplicação

Grupo 10

A93253 - David Duarte

A93290 - Joana Alves

A93264 - Maria Cunha



A94166 - Samuel Lira

A93296 - Vicente Moreira

Índice

Índice de figuras

Introdução do projeto

Este projeto tem como objetivo a conceção, modelação, planeamento e implementação de uma aplicação de software capaz de gerir um centro de reparações de equipamentos eletrónicos, de forma a otimizar o tempo e recursos envolvidos no negócio e reduzir as dificuldades na gestão de empregados, inventário e clientes.

Para isso, foram-nos fornecidos vários cenários de utilização do sistema, assim como requerimentos e funcionalidades que este terá de disponibilizar.

Objetivos da fase

Nesta segunda fase do projeto, é necessário desenvolver a especificação da aplicação, com a utilização de diagramas, assim como desenvolver uma implementação de forma a garantir o cumprimento dos cenários e requisitos levantados da fase anterior. Para isso, optamos por desenvolver diagramas de subsistemas, diagrama de classes e diagramas de sequência.

Descrição do trabalho realizado

Para a realização desta fase, continuamos a evoluir o trabalho já alcançado na fase anterior.

Visto que, com o desenvolvimento dos fluxos, já possuímos todos os métodos necessários para obter uma lógica de negócio funcional, começamos por dividir estes métodos em vários subsistemas de forma a categorizá-los, e, assim, simplificar a implementação de cada subsistema.

Para unir estes subsistemas, recorremos a um diagrama de componentes com o objetivo de demonstrar a interação destes entre si e também as relações destes com a interface principal da lógica de negócio.

Depois de termos estes diagramas reunidos, passamos de seguida por realizar em conjunto o Diagrama de Classes da aplicação e os Diagramas de Sequência dos métodos da interface de negócio. Estes diagramas têm que ser capazes de responder a todas as necessidades de armazenamento e manipulação dos dados durante a execução da aplicação. Estes foram realizados em conjunto sendo esta a nossa estratégia para mitigar inconsistências no sistema tendo em vista que modificações num dos diagramas facilmente poderiam alterar outros.

Por último, para o desenvolvimento do código da aplicação, este foi iniciado quando a equipa acreditou ter um diagrama de classes perto da sua iteração “final”, sendo o código ligeiramente modificado devido a eventuais alterações nos diagramas, ou por problemas encontrados na fase de implementação.

Mudanças da primeira fase

Em relação ao planeamento da primeira fase, a nossa equipa não encontrou muita necessidade de recorrer a alterações nem nos modelos de *Use Case* nem nos fluxos desenvolvidos, apenas modificando alguns métodos para incluírem certos argumentos em falta.

Diagrama de Componentes

Depois de dividir os vários métodos obtidos nos fluxos, encontramos a necessidade de criar um diagrama de subsistemas, assim como um diagrama de classes exclusivo às interfaces, de forma a demonstrar as relações entre os vários subsistemas e os métodos de cada um destes.

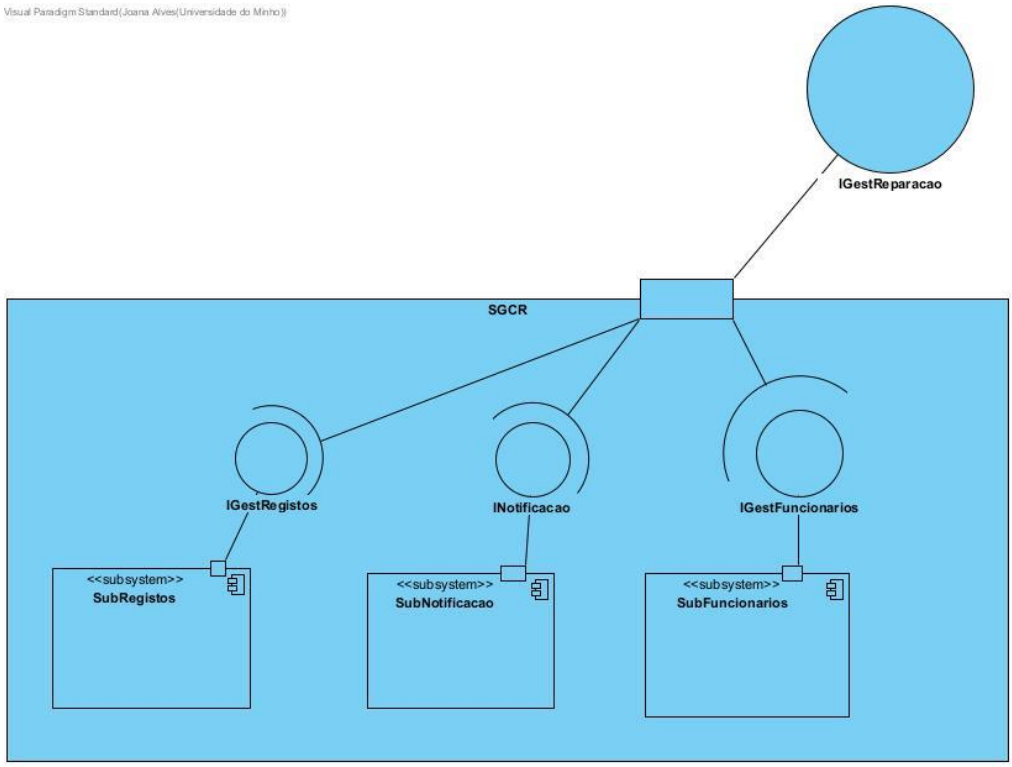


Figura 1 - Diagrama de Componentes

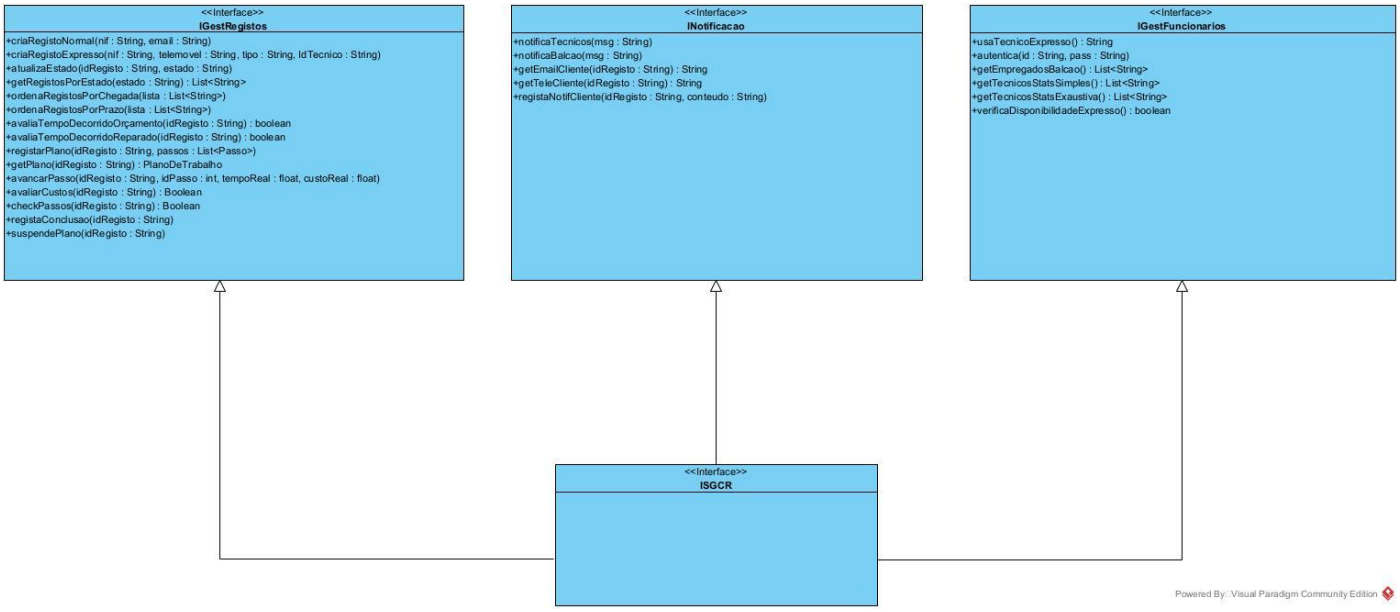


Figura 2 - Diagrama de Interfaces

Diagrama de Classes

Desenvolvemos de seguida, um protótipo inicial do nosso diagrama de classes. Este apenas continha a interface da lógica de negócio, assim como a estrutura das listas de equipamentos, funcionários, clientes e registos. Conforme o desenvolvimento dos diagramas de sequência, foi necessário evoluir o nosso diagrama de classes de forma que este tivesse todas as estruturas e classes requeridas.

Com o desenvolvimento de todos os diagramas completos, obtivemos este diagrama de classes final:

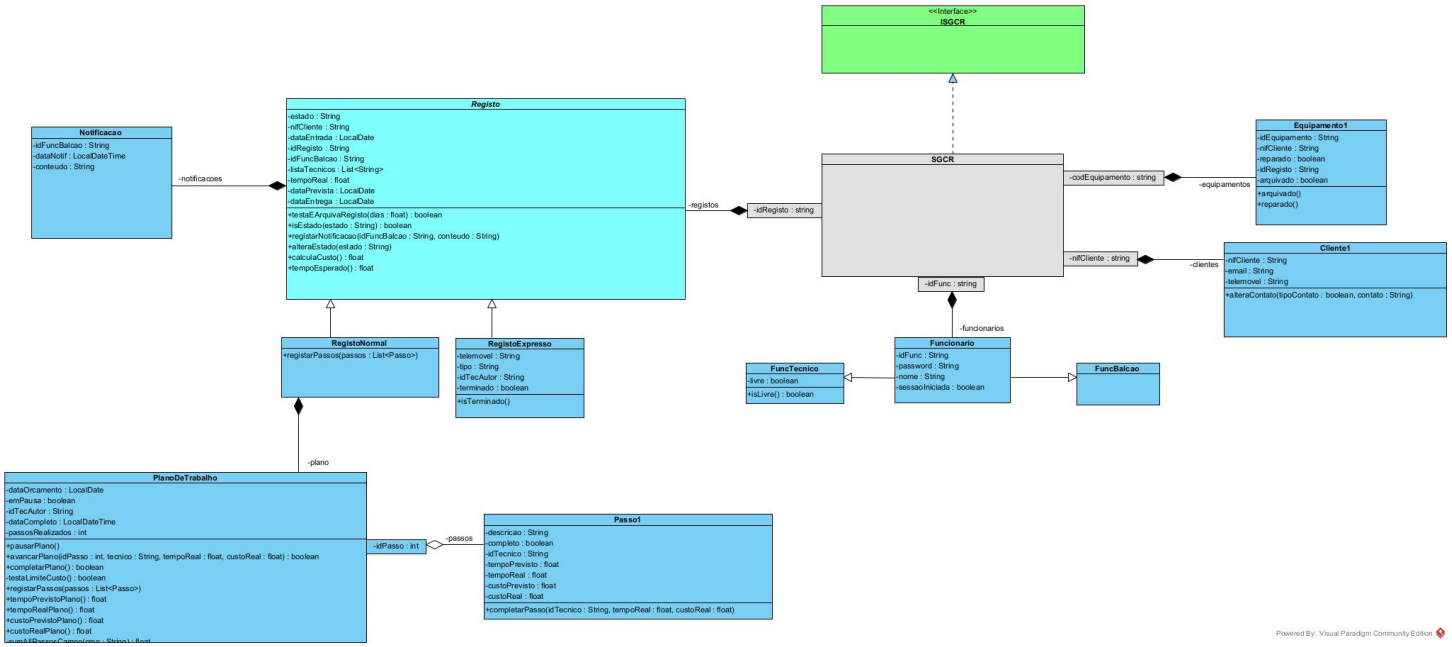


Figura 3 - Diagrama de Classes

Diagramas de Sequência

Os diagramas de sequência foram cruciais no planeamento e implementação do código pois determinam de forma especifica o funcionamento de cada método presente na interface lógica e as suas classes correspondentes.

Apresentamos aqui apenas alguns dos diagramas de sequência desenvolvidos, começando por apresentar o método de autenticação de funcionários e a criação de registos de reparação normal. De seguida apresentamos diagramas correspondentes à interação do plano de trabalho de um equipamento e por último os diagramas de sequência dos métodos responsáveis por recolher registos de reparação num dado estado e avaliação do tempo decorrido de uma reparação.

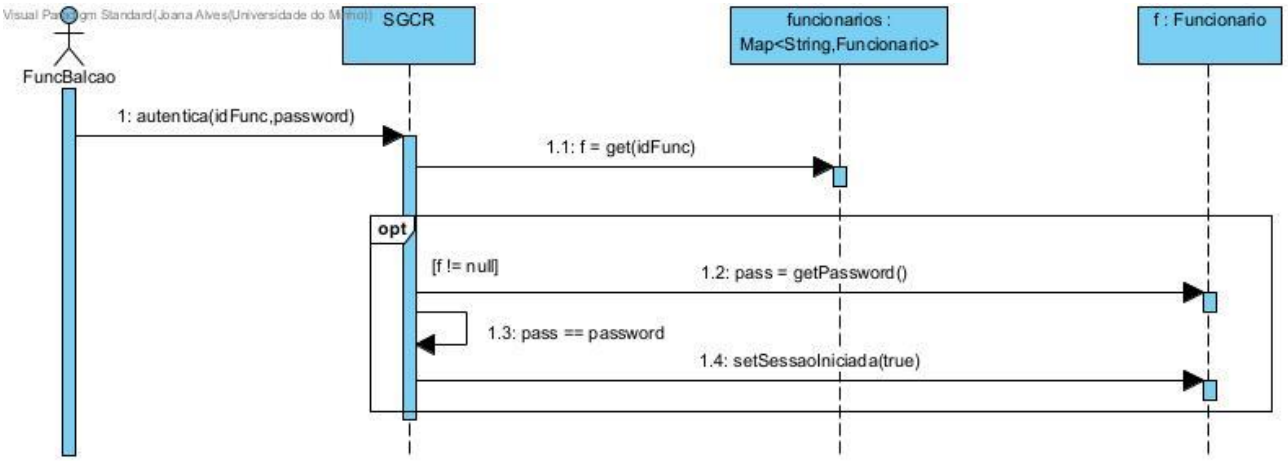


Figura 4 - Diagrama de Sequência "autentica"

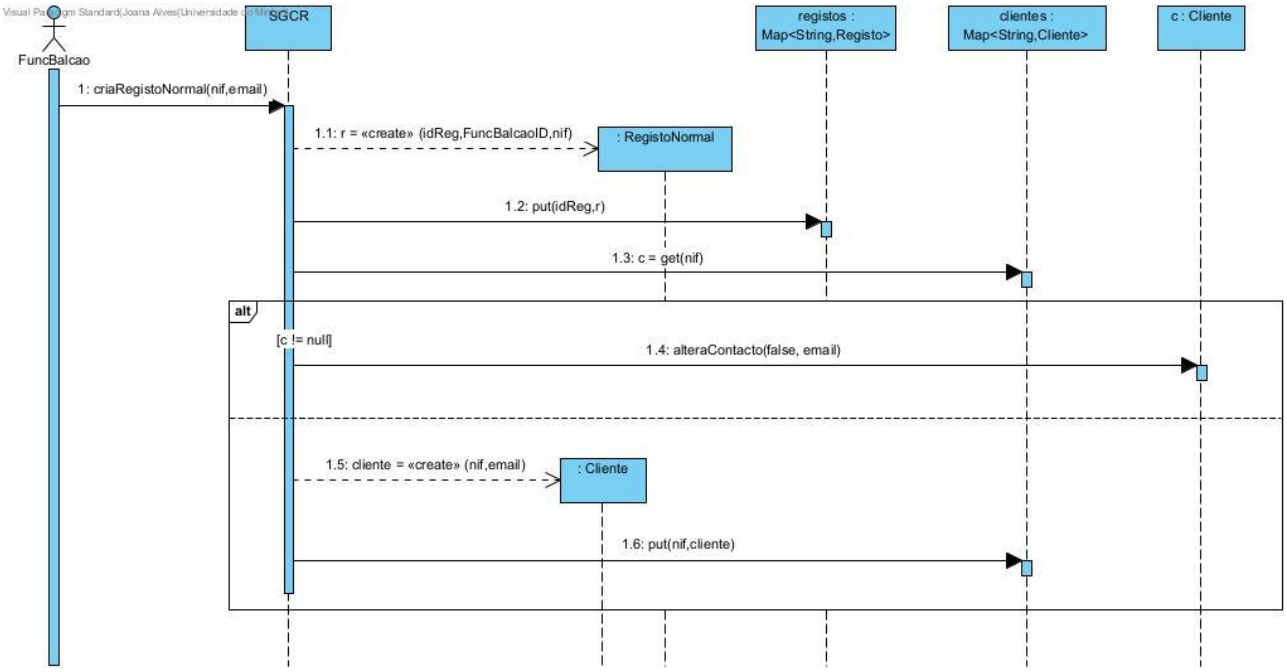


Figura 5 - Diagrama de Sequência "criaRegistoNormal"

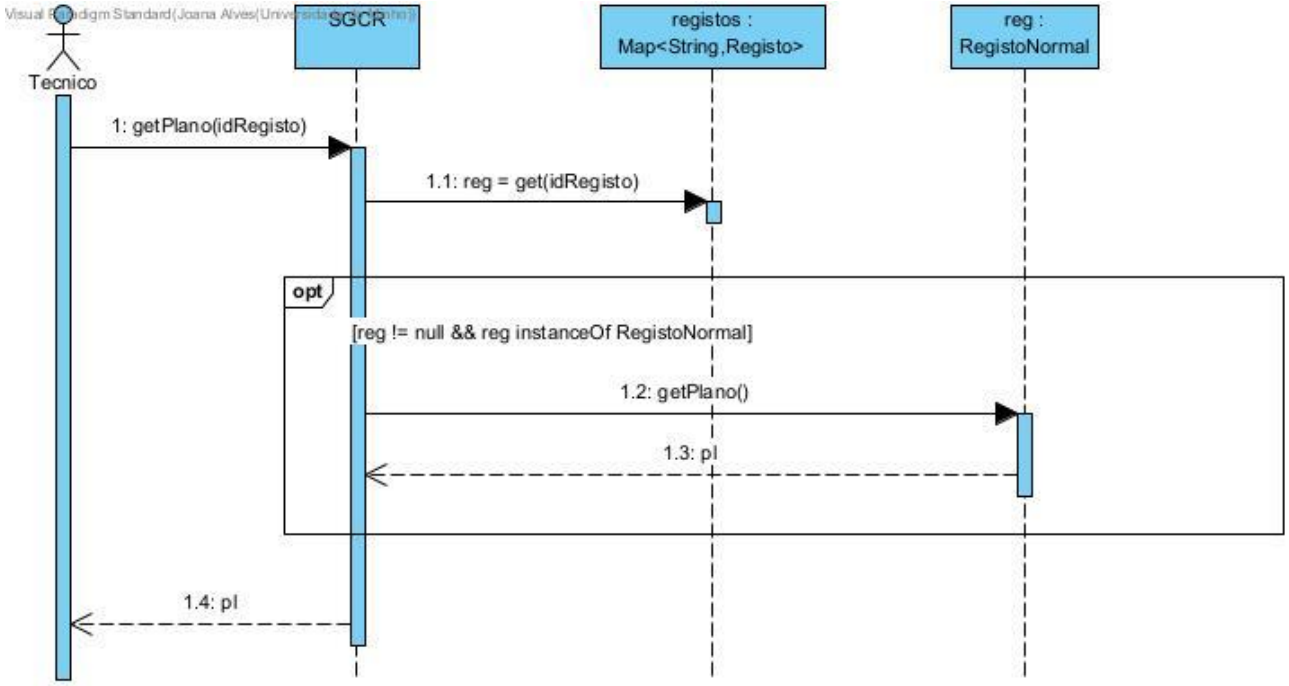


Figura 6 - Diagrama de Sequência "getPlano"

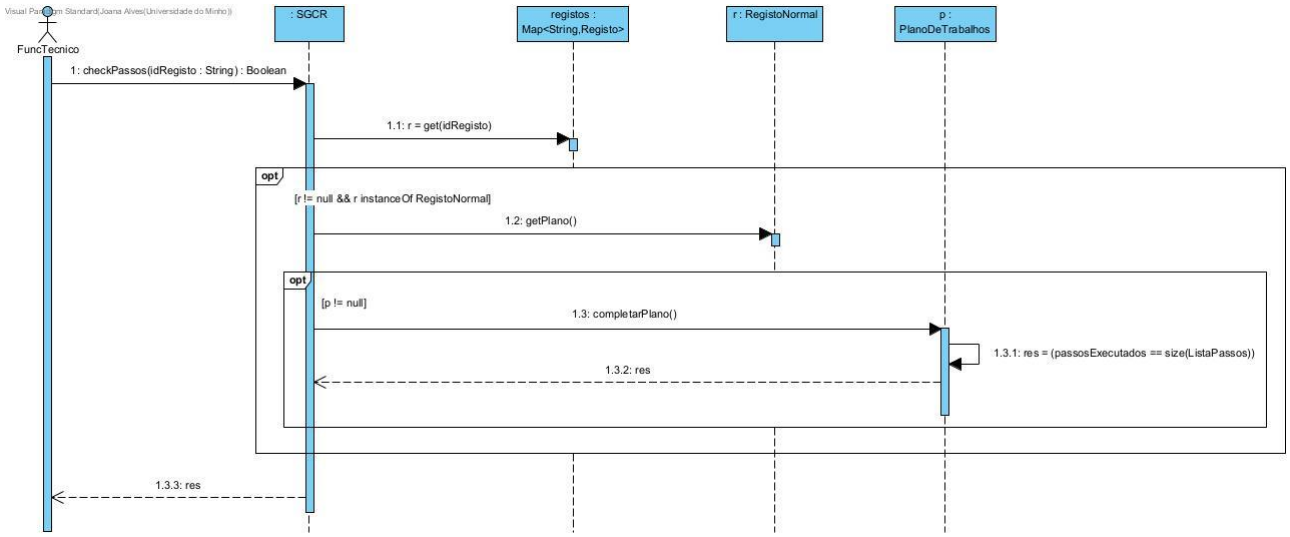


Figura 7 - Diagrama de Sequência "checkPassos"

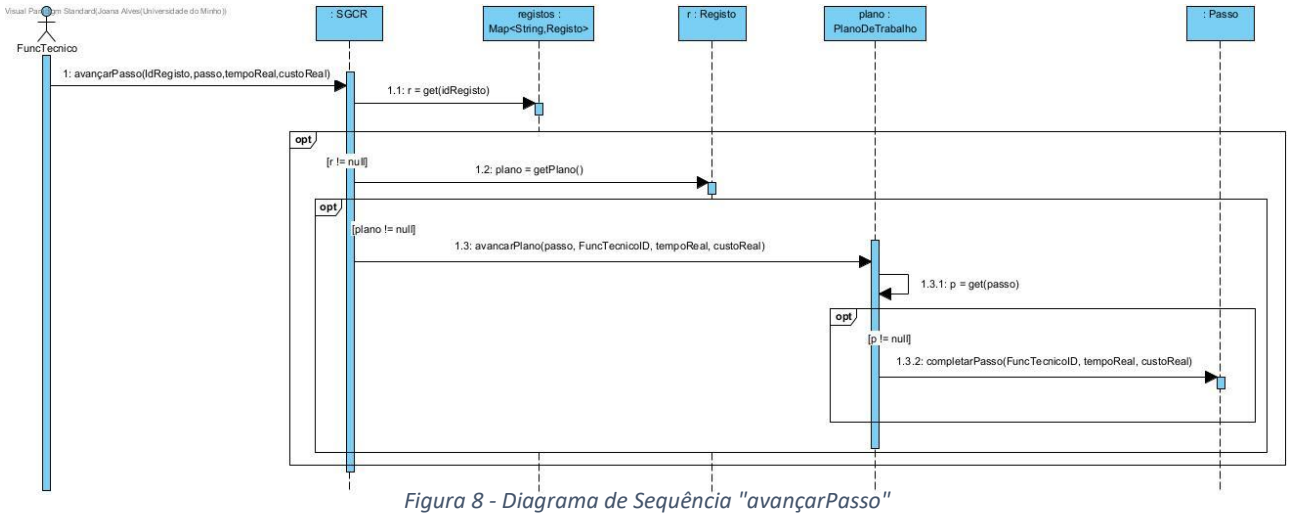


Figura 8 - Diagrama de Sequência "avançarPasso"

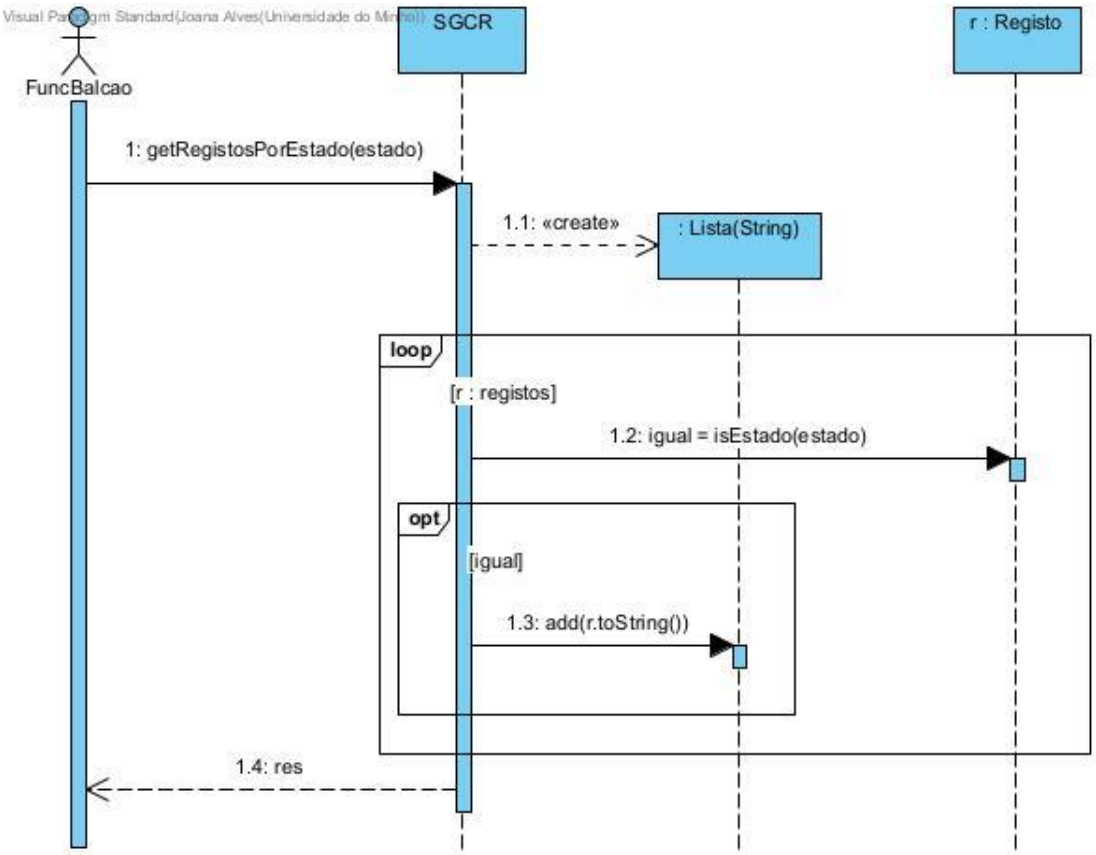


Figura 9 - Diagrama de Sequência "getRegistosPorEstado"

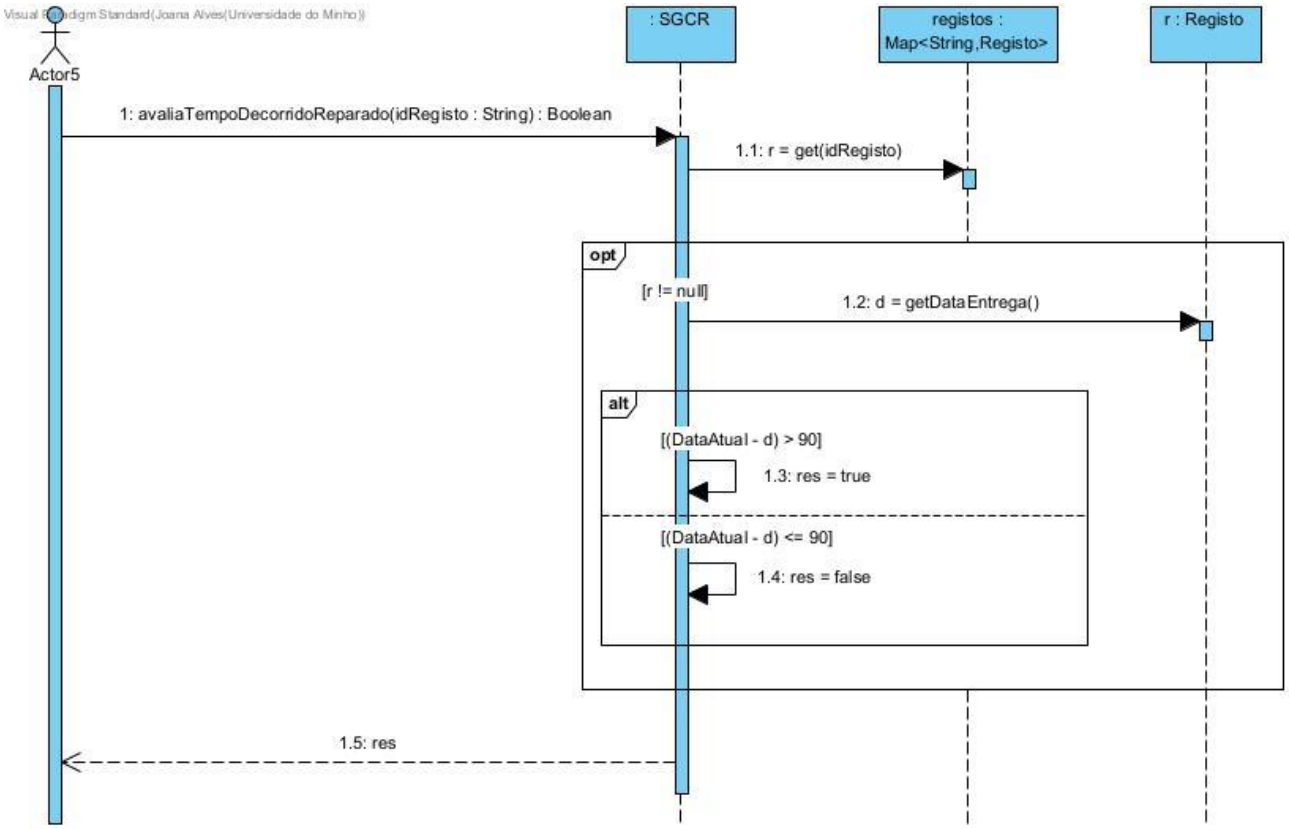


Figura 10 - Diagrama de Sequência "avaliaTempoDecorridoReparado"

Análise crítica dos resultados obtidos

Com a realização deste projeto, a nossa equipa está suficientemente satisfeita com os resultados obtidos, pois apesar de não ter sido capaz de completar a implementação devido à falta de tempo, acreditamos que o planeamento foi realizado de forma correta e satisfaz todos os requisitos necessários para uma futura realização da aplicação final.

Exemplo "criaRegistoNormal"

Este *UseCase* originou devido à necessidade de criar e introduzir registos de novas reparações no sistema. Determinamos que este funcionaria da seguinte forma: O funcionário de balcão recolhe o NIF e email do cliente. Caso estes não fossem fornecidos, o serviço seria rejeitado pois não há forma de obter os dados necessários. Caso o cliente os forneça, o funcionário de balcão introduz este novo registo no sistema, recolhendo também o equipamento a ser reparado.

Este método foi inserido no subsistema de registos, dado que este opera sobre a lista de registos no sistema. Depois passamos a definição do Diagrama de Sequência deste método. Este teve de ser desenvolvido com cuidado, não só pela necessidade de criar o registo e marcar este como "Por Orçamentar", mas também de adicionar o cliente na base de dados da aplicação. Por último, implementamos na linguagem de programação *JAVA* o método com o objetivo de seguir de forma precisa a especificação do diagrama de sequência (Figura 5), obtendo o seguinte resultado:

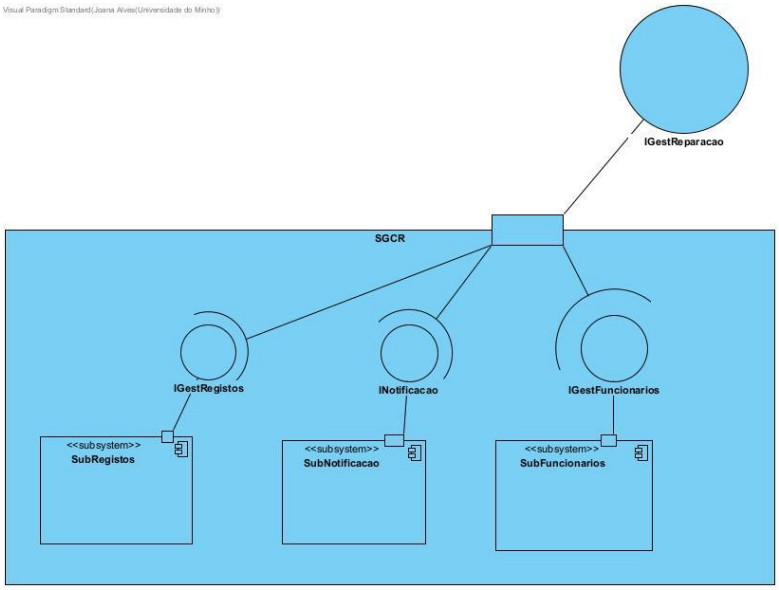
```
public void criaRegistoNormal(String nif, String email){
    String idReg = geraIdRegisto();
    RegistoNormal reg = new RegistoNormal(idReg,this.idFuncionario,nif);
    this.registos.put(idReg,reg);

    if(this.clientes.containsKey(nif))
        this.clientes.get(nif).alteraContacto( tipoContatoTelemovel: false,email);

    else {
        Cliente cliente = new Cliente(nif,email, t: "");
        this.clientes.put(nif,cliente);
    }
}
```

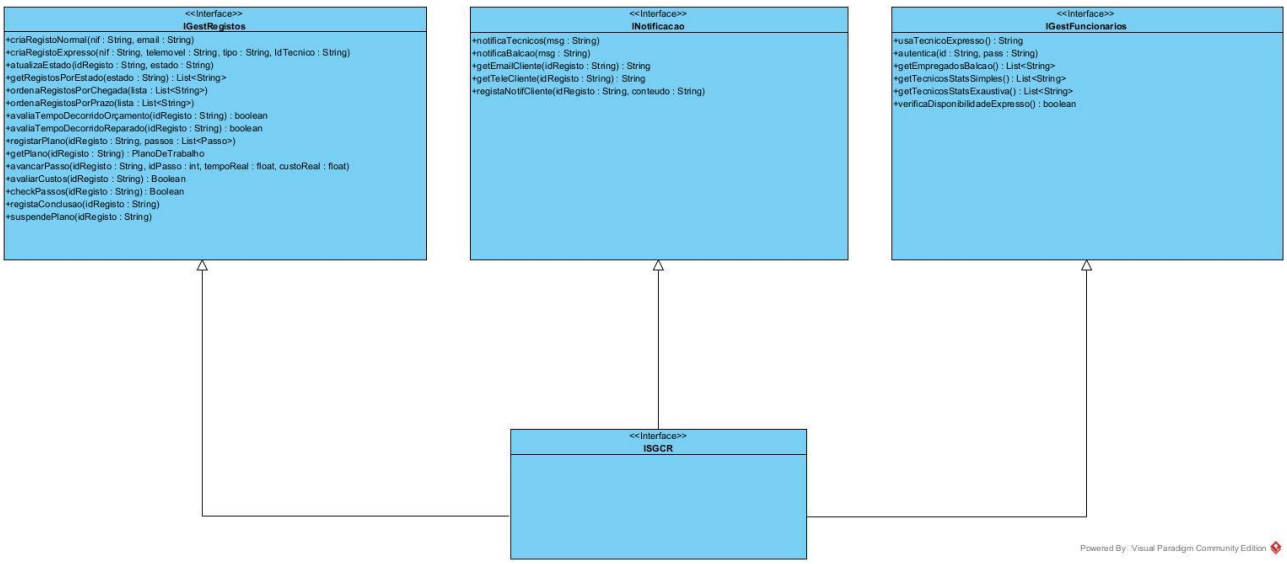
Conclusão

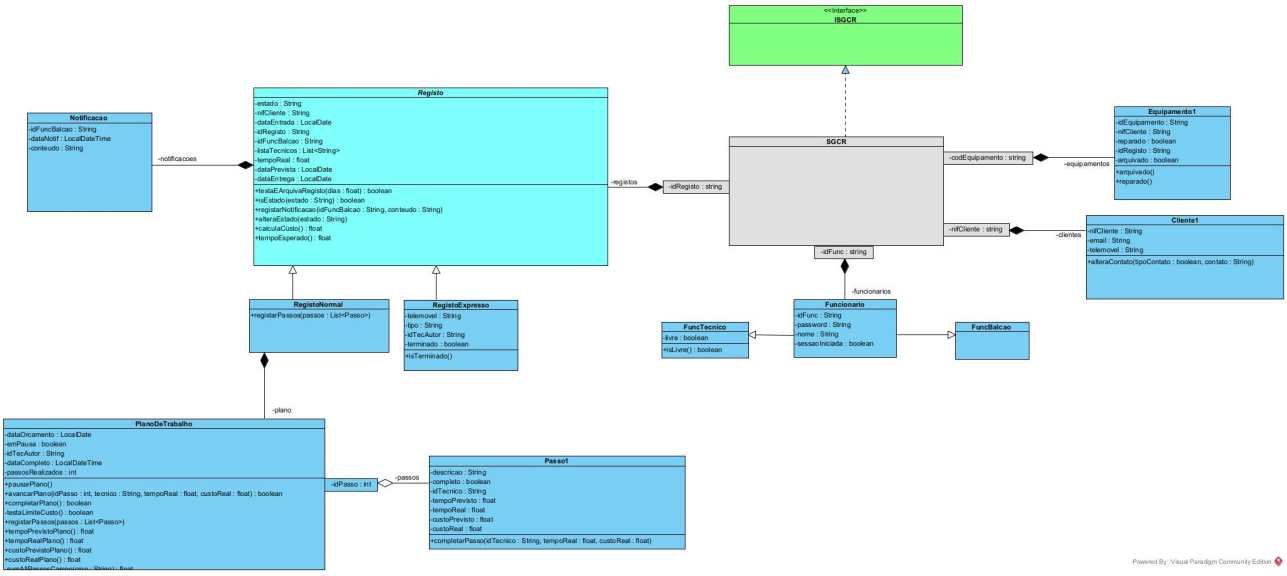
Concluímos assim este projeto para a conceção, planeamento e implementação de uma aplicação a ser utilizada no auxílio na gestão de tempo e recursos de um centro de reparações de equipamentos eletrónicos. Com este projeto, desenvolvemos uma melhor perspetiva da importância de um bom planeamento ao realizar projetos de engenharia de software e como os vários diagramas aprendidos durante as aulas podem ajudar não só na especificação do código a ser implementado na fase de programação como também na explicação clara a potenciais clientes e futuros programadores da estrutura da aplicação e todos os seus componentes.

Anexos

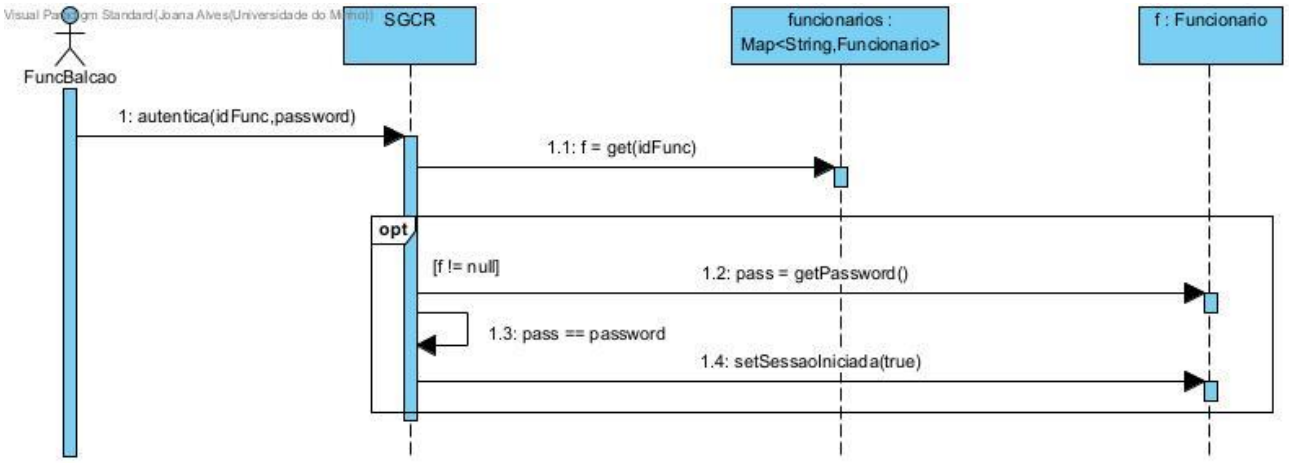
Diagramas de Componentes e Classes







Diagramas de Sequência IGestFuncionarios



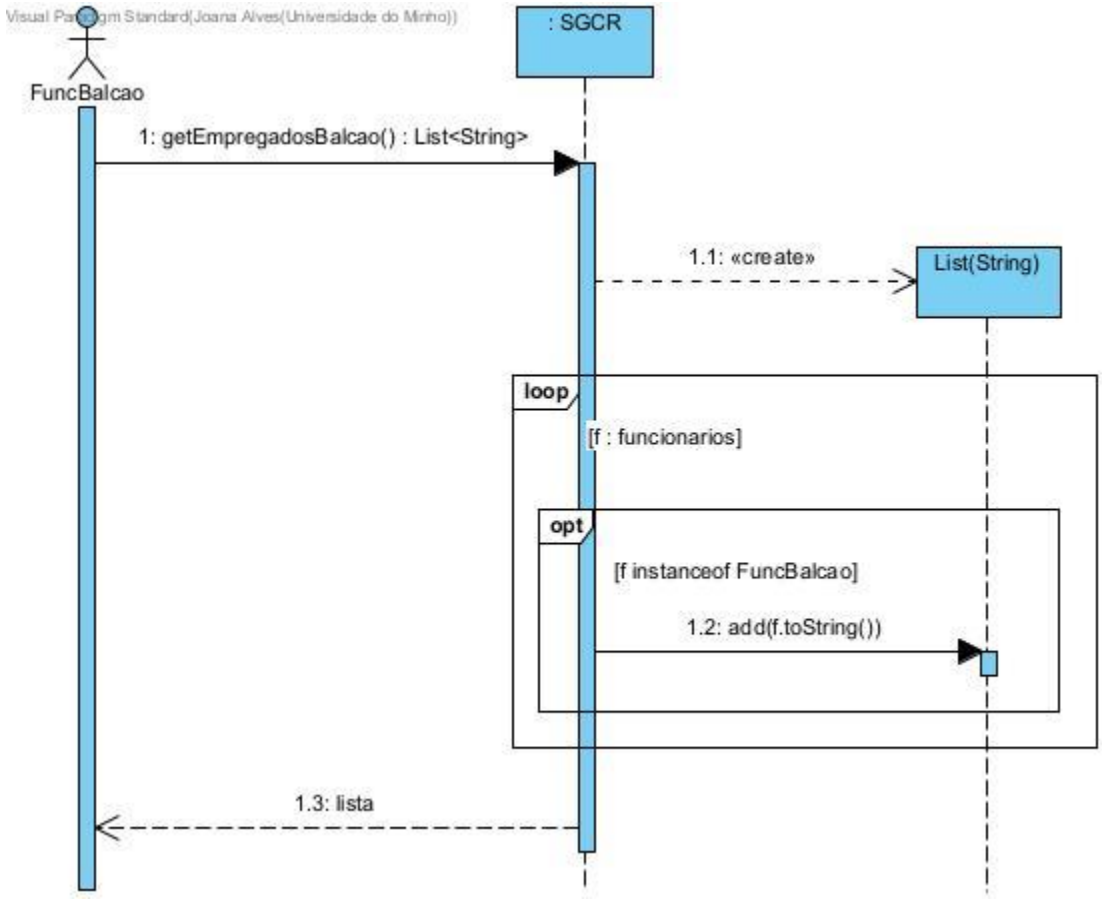


Figura 11 - Diagrama de Sequência "getEmpregadosBalcao"

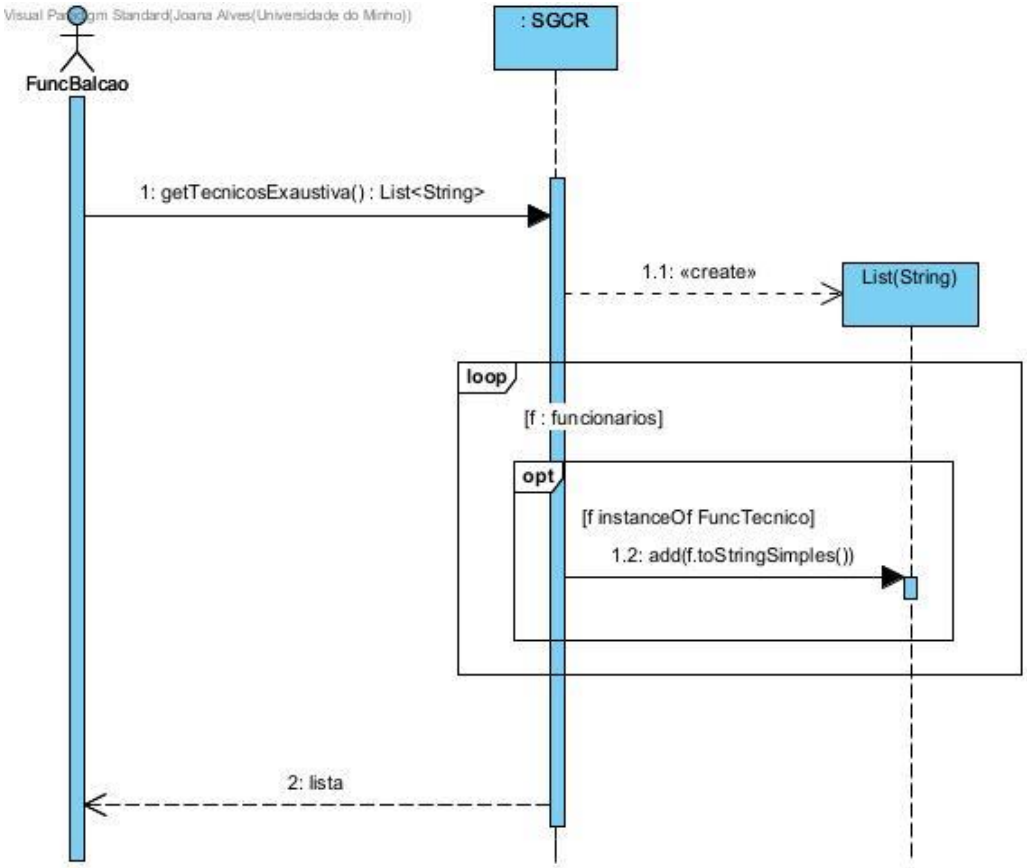


Figura 12 - Diagrama de Sequência "getTecnicosExaustiva"

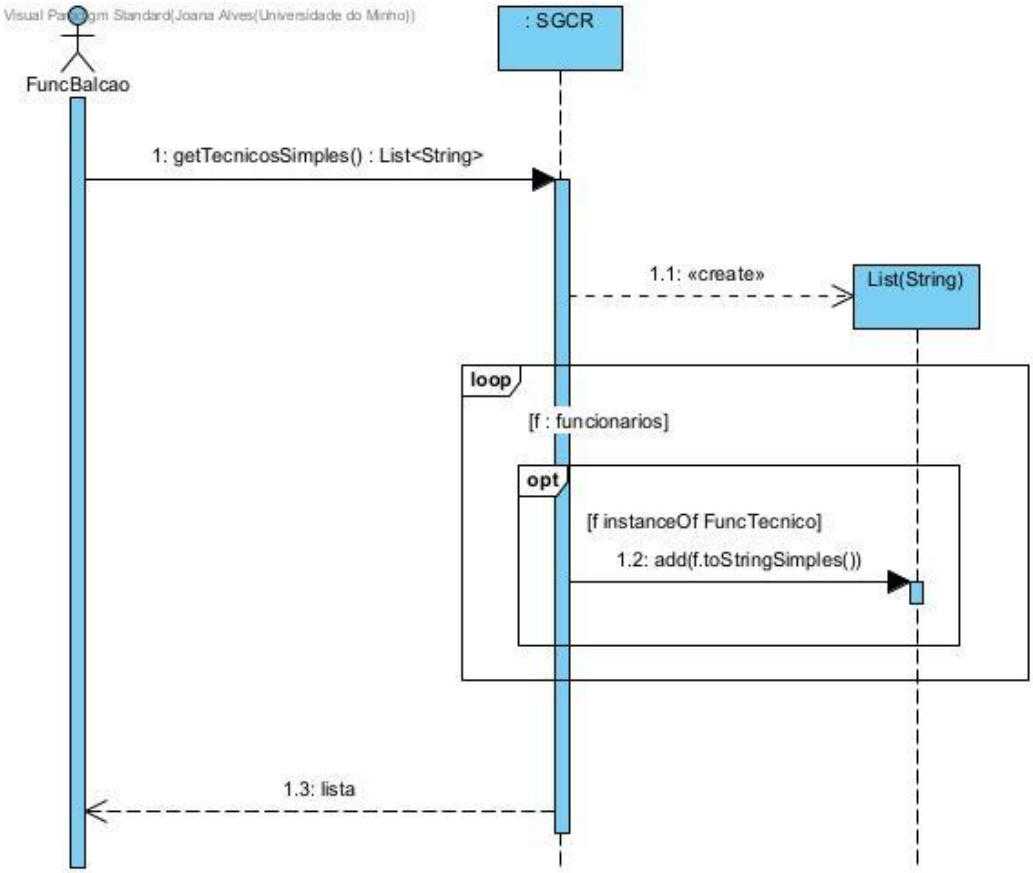


Figura 13 - Diagrama de Sequência "getTecnicosSimples"

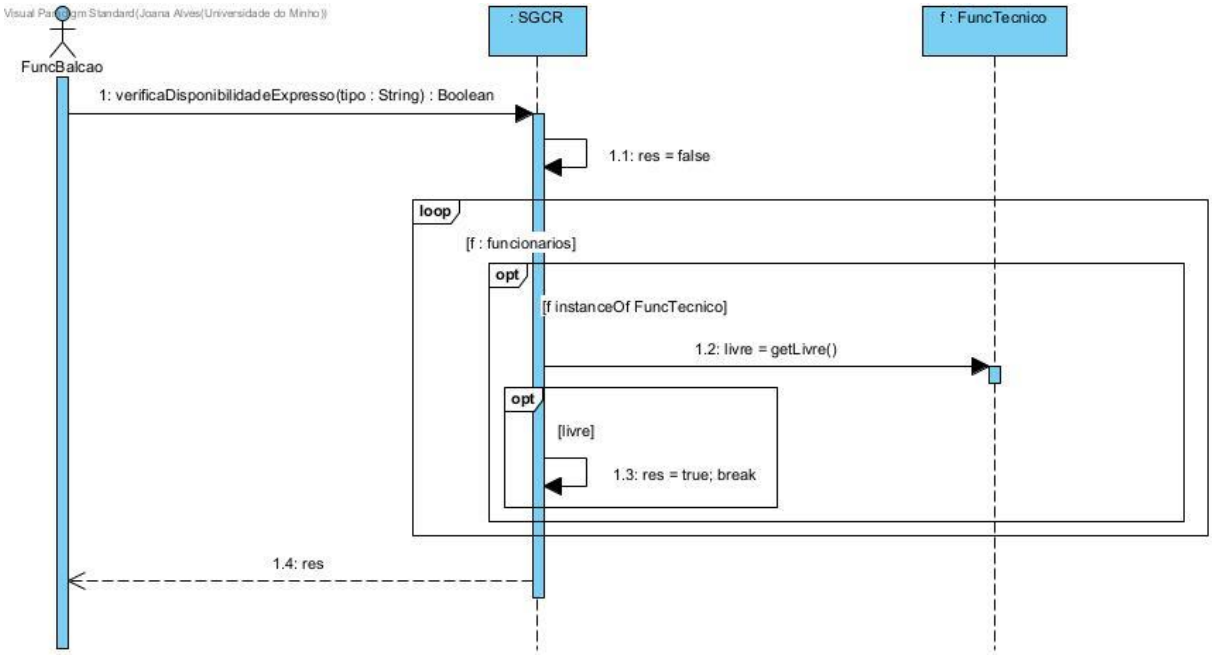


Figura 14 - Diagrama de Sequência "verificaDisponibilidadeExpresso"

Diagramas de Sequência IGestNotificacao

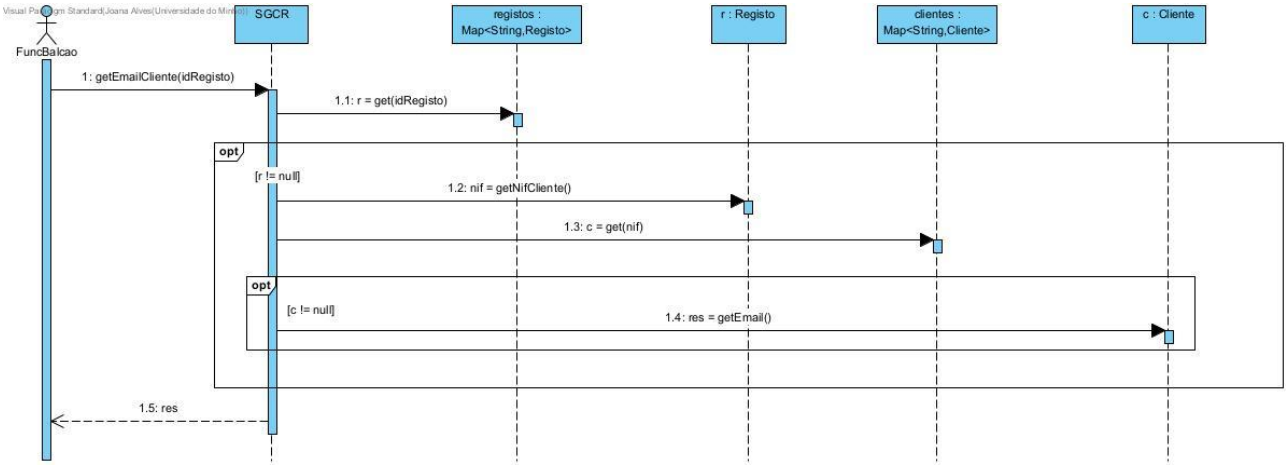


Figura 15 - Diagrama de Sequência "getEmailCliente"

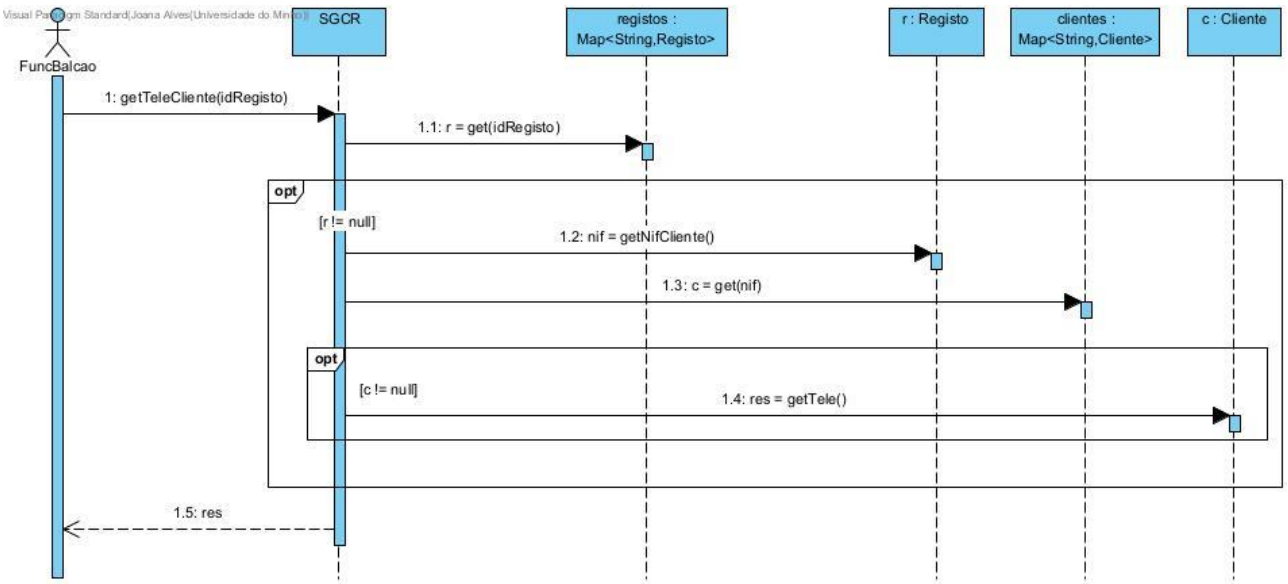


Figura 16 - Diagrama de Sequência "getTeleCliente"

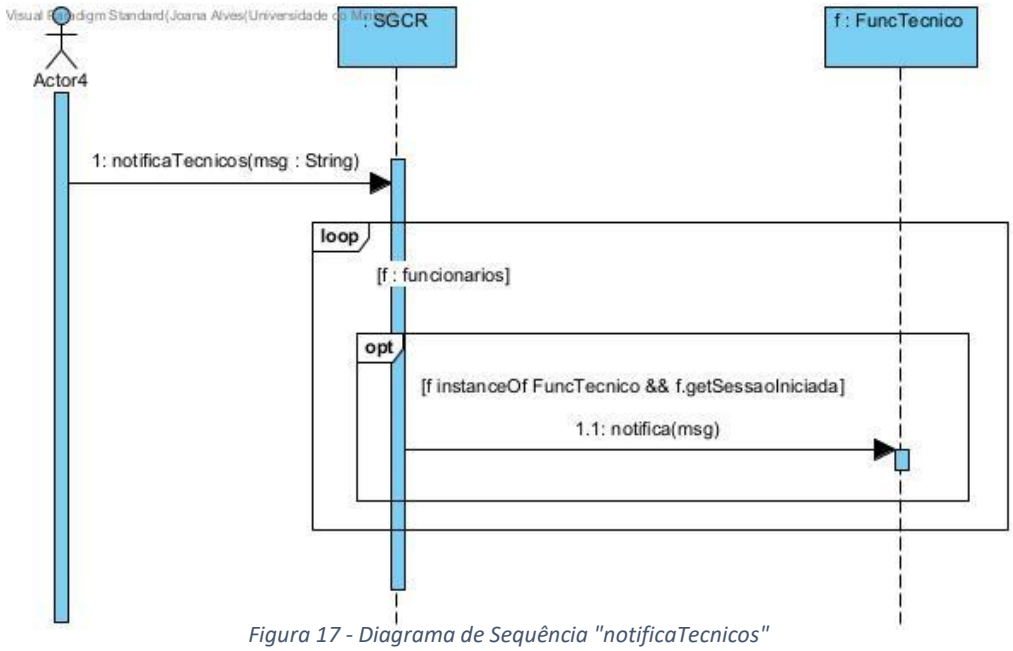


Figura 17 - Diagrama de Sequência "notificaTecnicos"

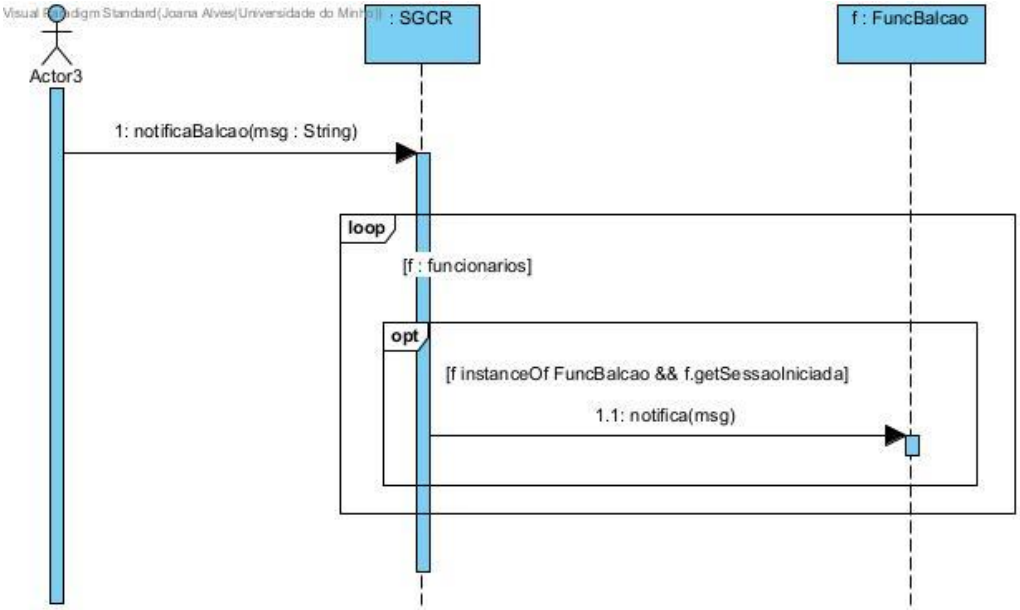


Figura 18 - Diagrama de Sequência "notificaBalcao"

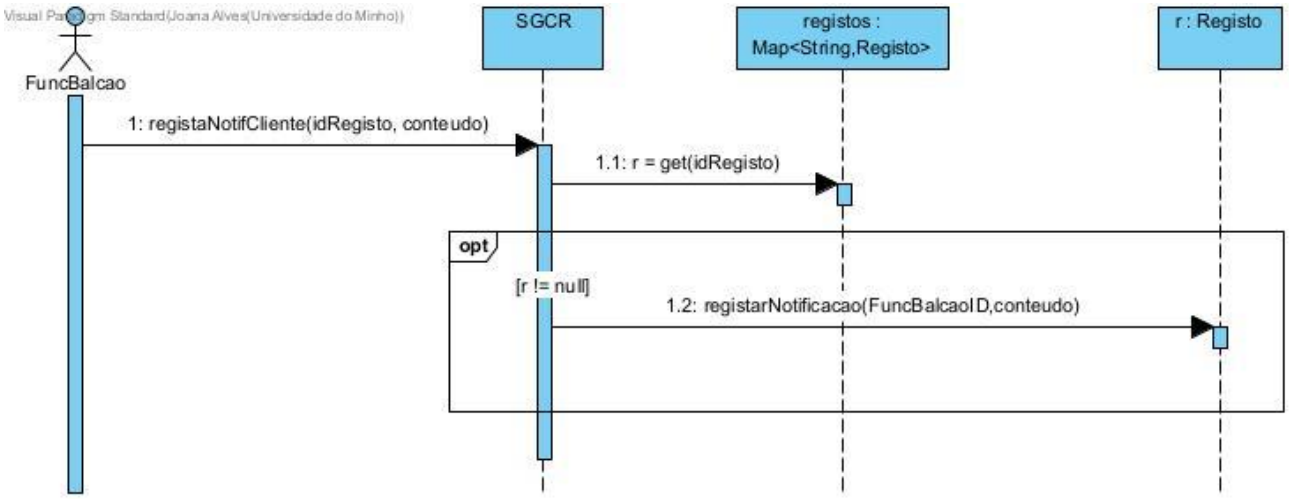


Figura 19 - Diagrama de Sequência "registaNotifCliente"

Diagramas de Sequência IGestRegistos

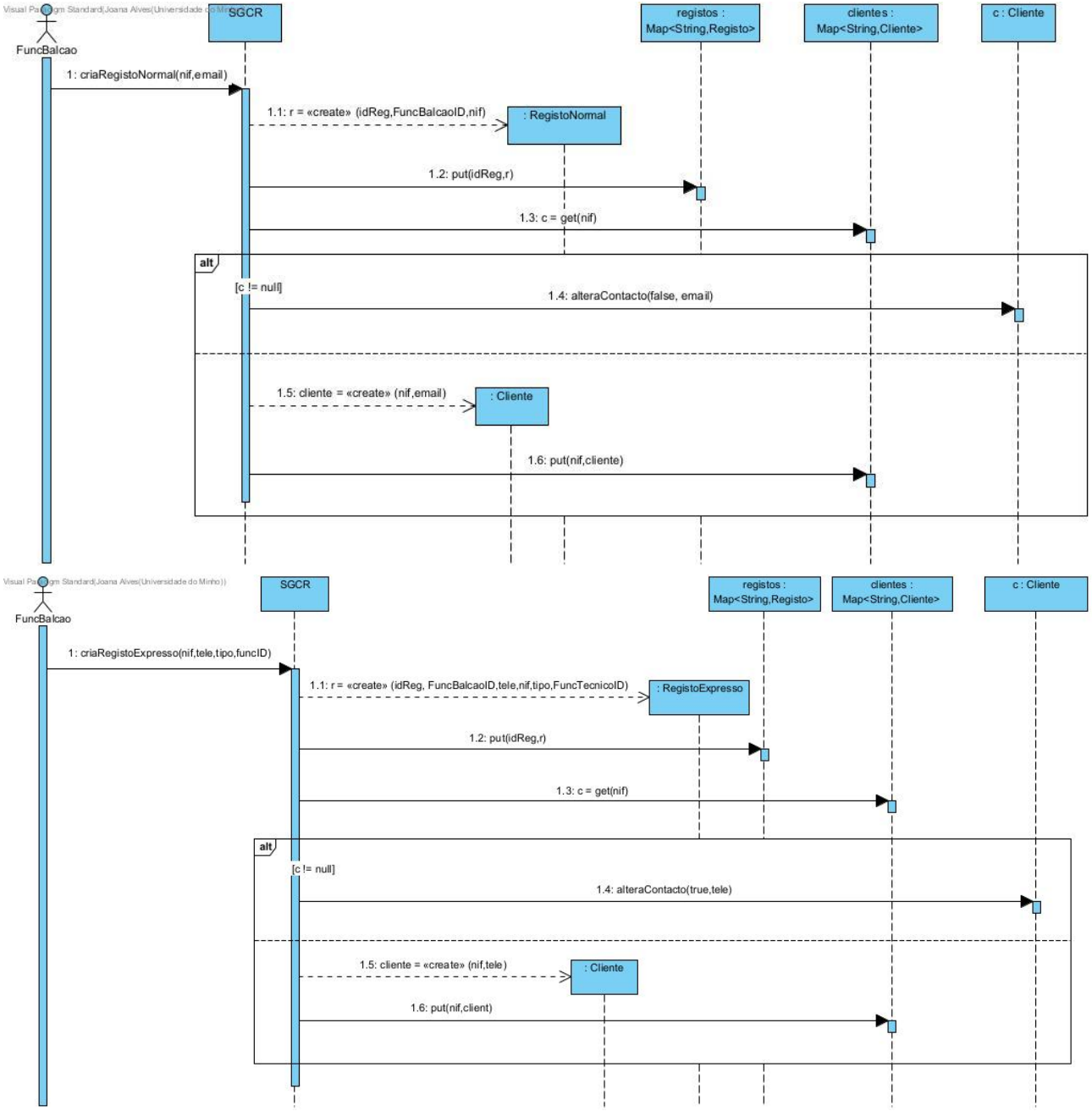


Figura 20 - Diagrama de Sequência "criaRegistoExpresso"

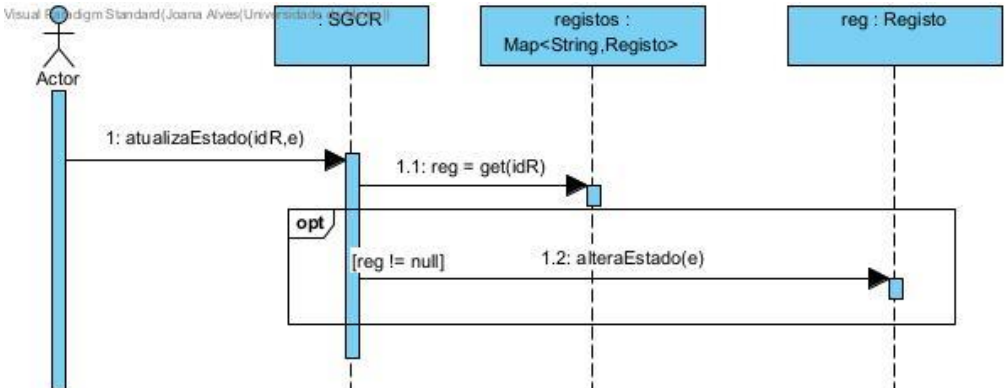


Figura 21 - Diagrama de Sequência "atualizaEstado"

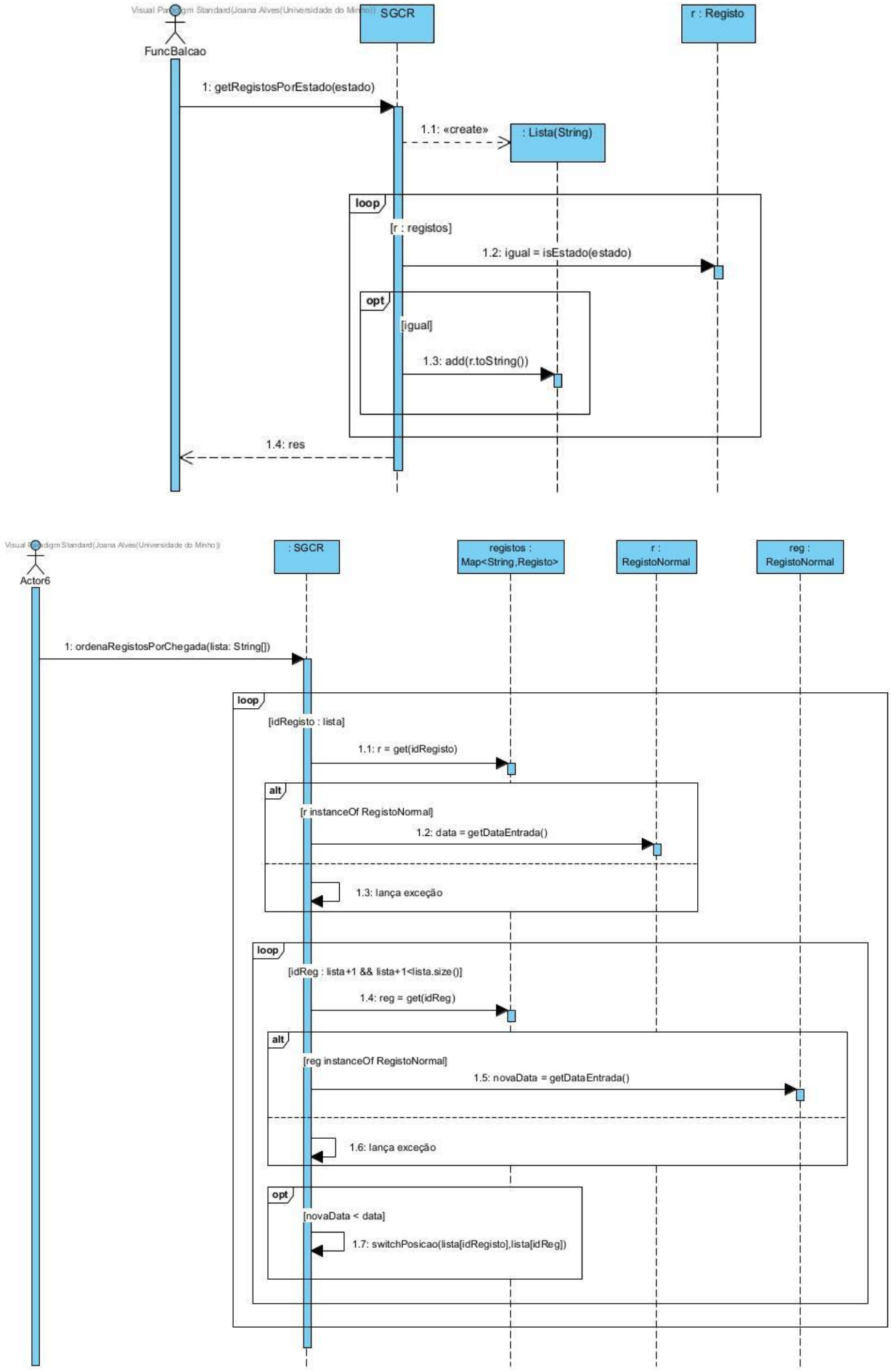


Figura 22 - Diagrama de Sequência "ordenaRegistosPorChegada"

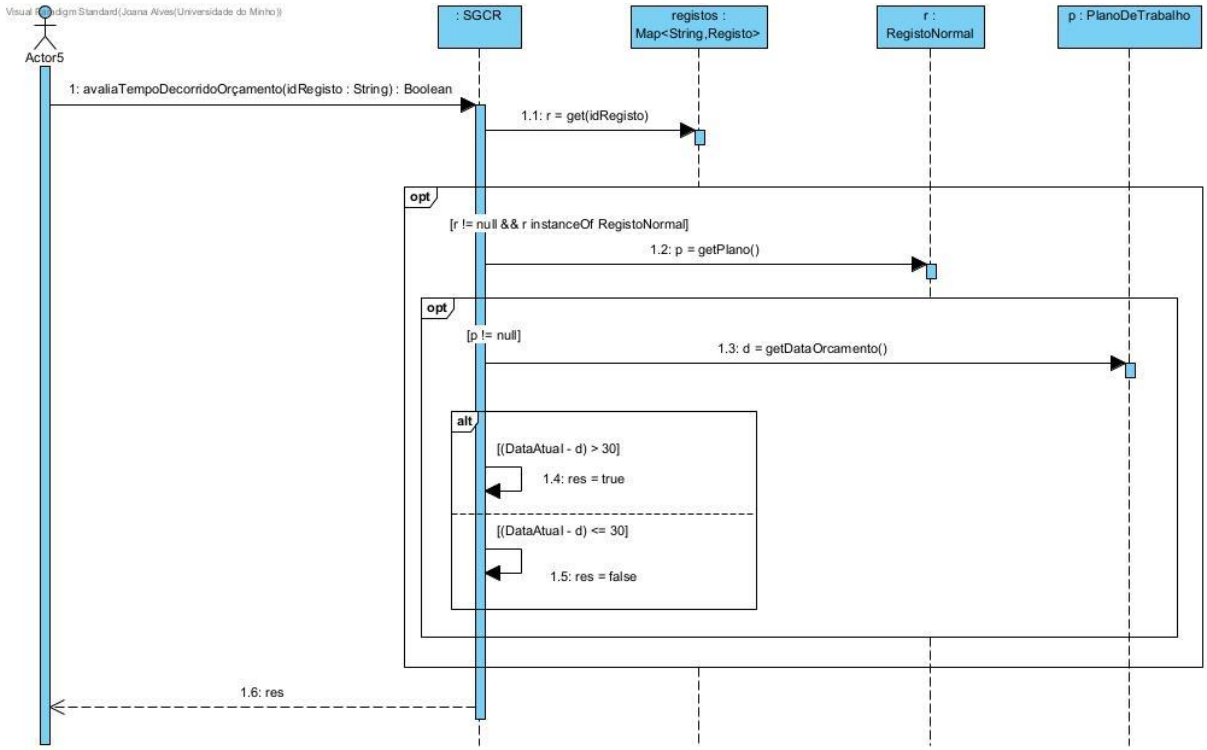


Figura 23 - Diagrama de Sequência "avaliaTempoDecorridoOrçamento"

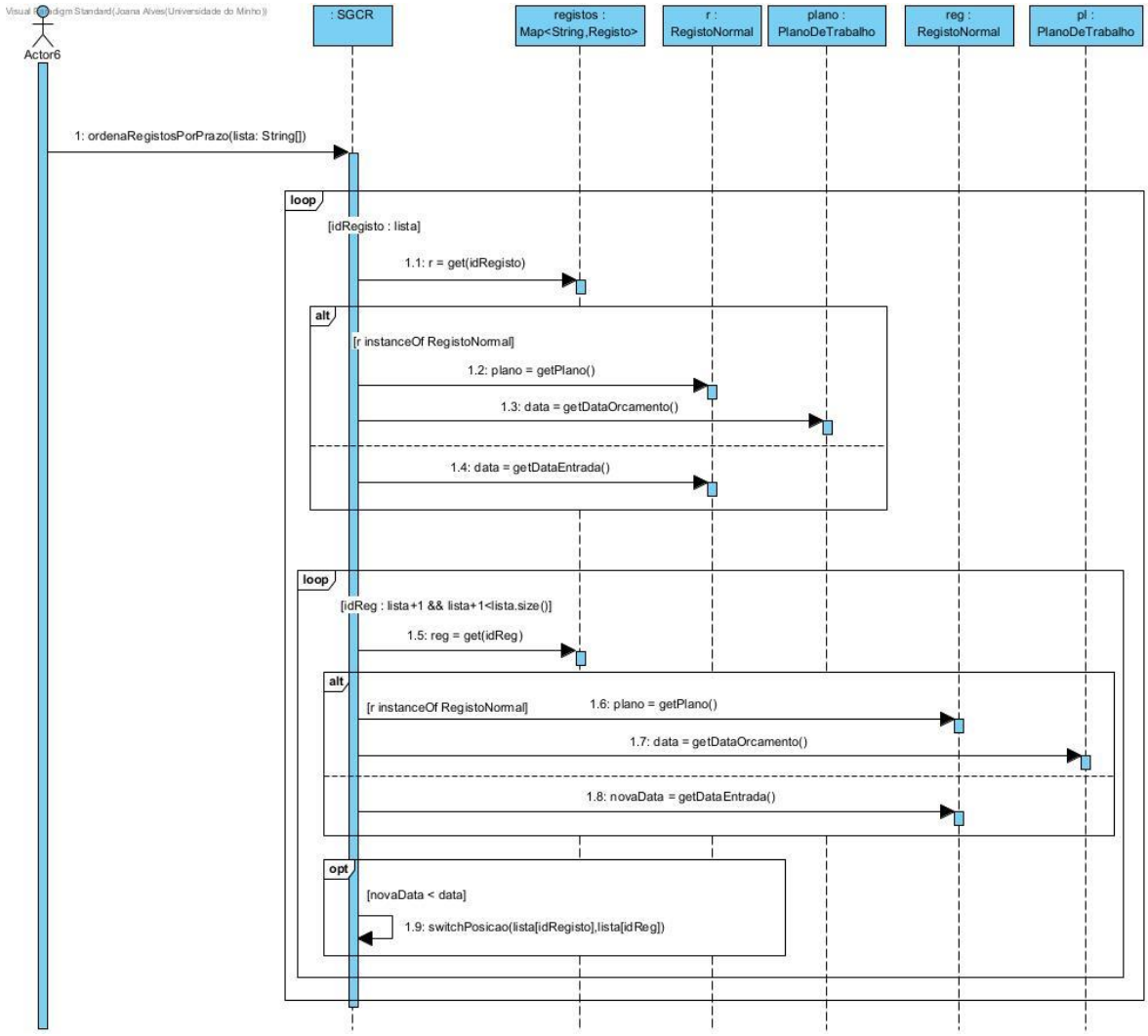


Figura 24 - Diagrama de Sequência "ordenaRegistosPorPrazo"

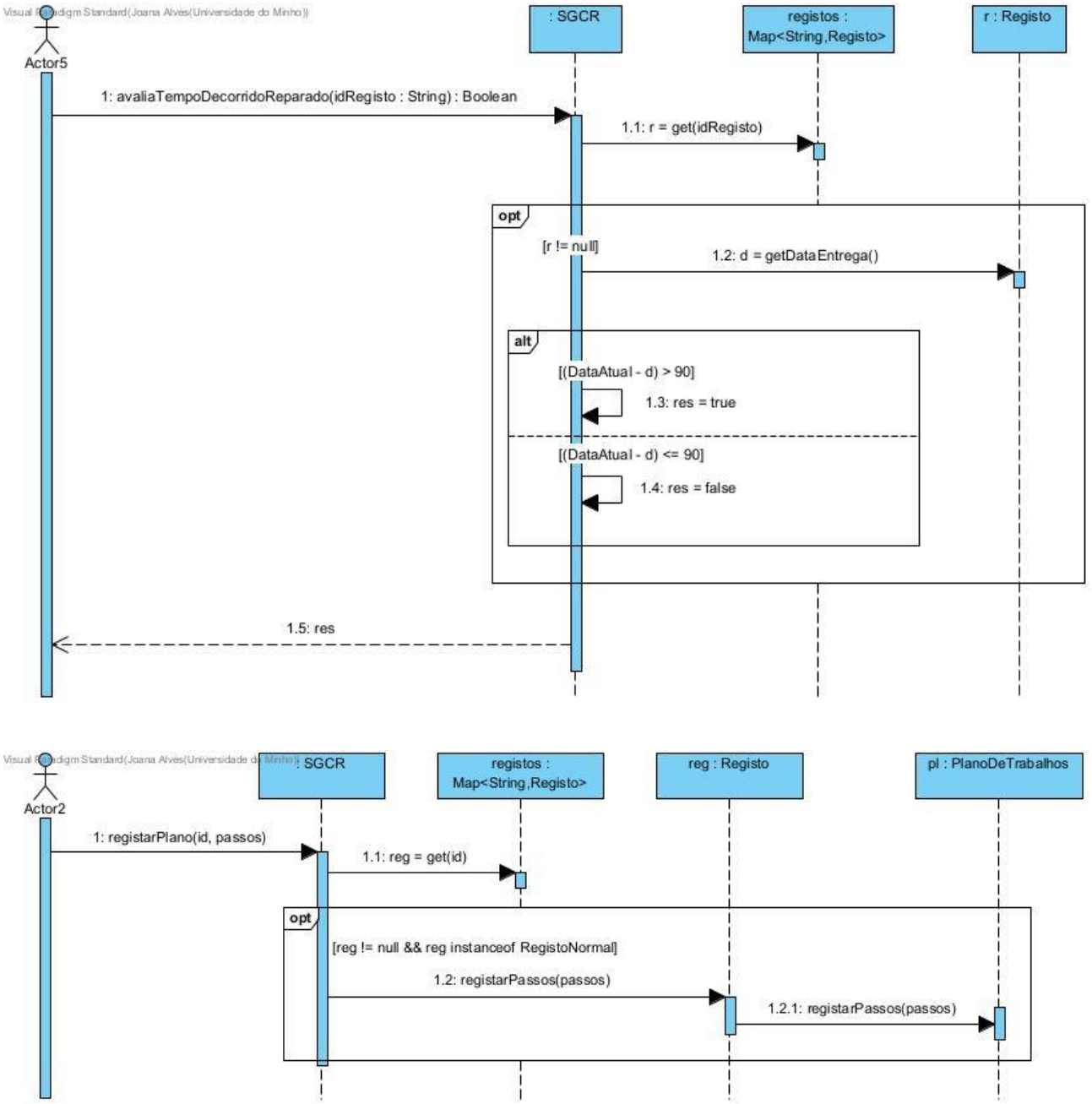
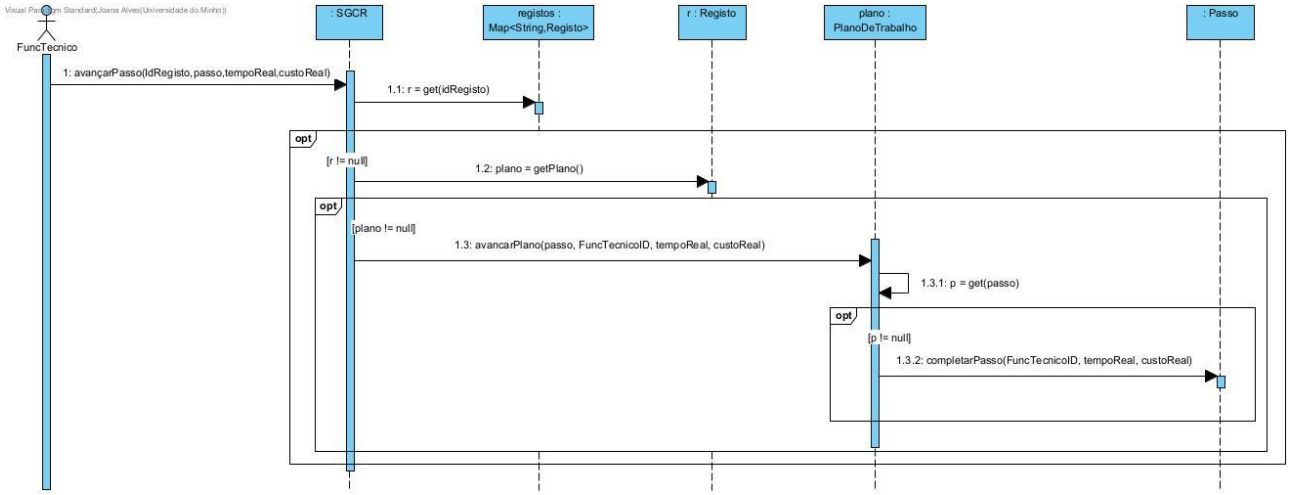
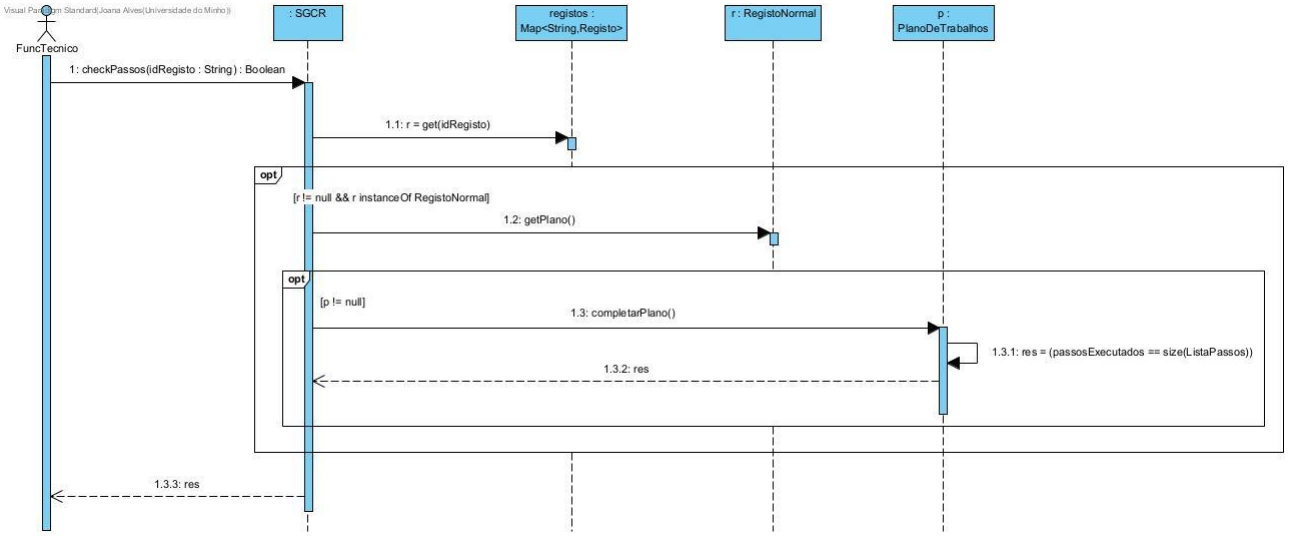


Figura 25 - Diagrama de Sequência "registarPlano"





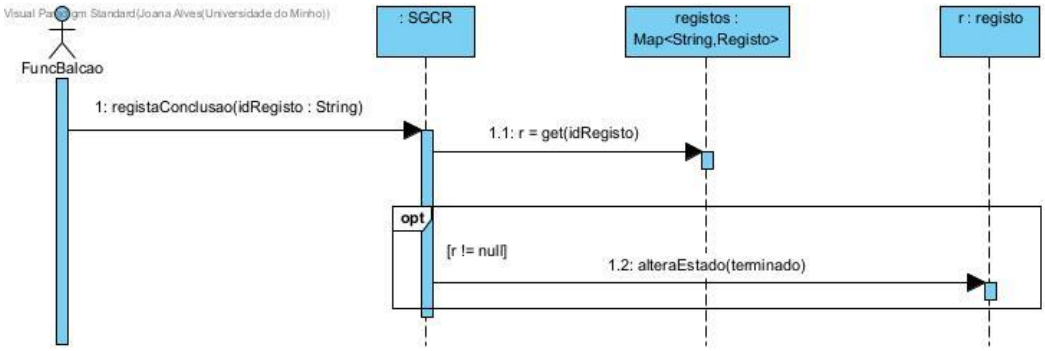


Figura 26 - Diagrama de Sequência "registaConclusao"

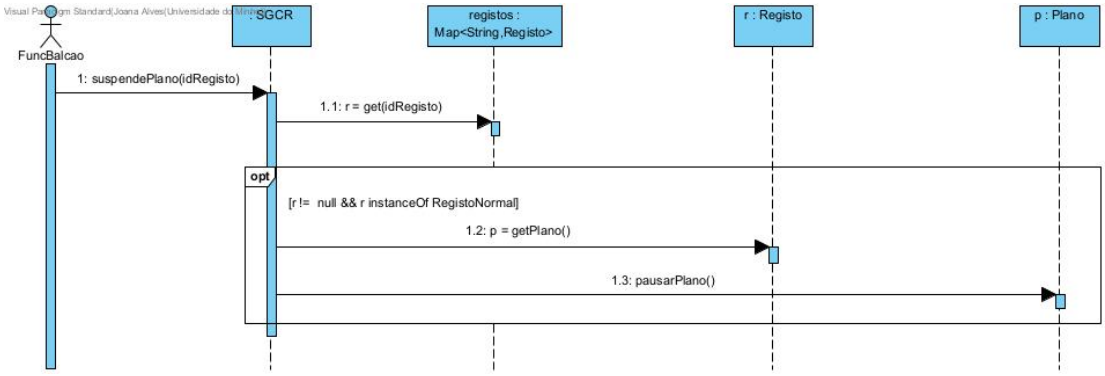


Figura 27 - Diagrama de Sequência "suspendePlano"