UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

José Luis Bressan Ruas

SISTEMA ASSISTENTE DE CERIMÔNIAS ICP (SACI)

Florianópolis

2018

José Luis Bressan Ruas

SISTEMA ASSISTENTE DE CERIMÔNIAS ICP (SACI)

Tese submetida ao Programa de Graduação em Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: Prof. Dr. Jean Everson Martina

Florianópolis

2018

José Luis Bressan Ruas

SISTEMA ASSISTENTE DE CERIMÔNIAS ICP (SACI)

Esta Tese foi julgada aprovada para a obtenção do Título de "Bacharel em Ciências da Computação", e aprovada em sua forma final pelo Programa de Graduação em Ciências da Computação.

Florianópolis, 1 de junho 2018.

Prof. Dr. Rafael Luiz Cancian
Coordenador do Curso

Banca Examinadora:

Me. Fernando Lauro Pereira
Presidente

Prof. Dr. Jean Everson Martina
Orientador

Me. Douglas Simões Silva

RESUMO

Em uma Infraestrutura de Chaves Públicas (ICP) é necessário que pessoas executem alguns procedimentos, estes chamados de cerimônias. Algumas destas cerimônias são executadas dentro de uma sala cofre de um Centro de Certificação Digital (CCD). Uma sala cofre é um ambiente estanque, onde não existe a troca de oxigênio, com o tempo os níveis de CO2 vão aumentando. A proposta deste trabalho é o desenvolvimento de um aplicativo Android para servir como um Sistema Assistente de Cerimônias ICP (SACI). Após o desenvolvimento o SACI foi avaliado nos termos de completude, ergonomia, utilidade e usabilidade, porém não se pode obter conclusões por causa da quantidade de entrevistados e do número de testes.

Palavras-chave: Cerimônias, Aplicativo, Sistema Assistente de Cerimônias ICP, SACI.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

A certificação digital pode ser utilizada para prover segurança digital. Para validar um certificado, este precisa pertencer a uma cadeia de certificação conhecida como Infraestrutura de Chaves Públicas (ICP). Uma ICP deve ser gerenciada por pessoas para controlar o ciclo de vida dos certificados digitais, portanto para gerenciar uma ICP é necessário executar cerimônias. Segundo (ELLISON, 2007), uma cerimônia é um protocolo de rede, que possui alguns nodos humanos e a troca de informação não está limitada a canais tradicionais de comunicação, incluindo desta forma todas as aplicações com suas interfaces gráficas e todas as instâncias do fluxo de trabalho. Além disso a entrada e saída das mensagens humanas é sempre suscetível a erro.

Atualmente as cerimônias apresentam diversos tipos de problemas acarretados por diferentes motivos, como a execução realizada por humanos; possuir muitos passos e sem a garantia do sequenciamento dos mesmos; serem executadas em softwares robustos, com muitas funcionalidades. Algumas das cerimônias devem ser executadas dentro de um Centro de Certificação Digital (CCD).

Segundo (LUZ, 2008), um CCD é um tipo de *datacenter* especializado em certificação digital e tem como principal objetivo garantir a segurança física e digital de uma ICP. Um CCD possui uma sala cofre, que é uma célula estanque, ou seja, não tem troca de ar. O ambiente também possui baixas temperaturas em torno de 21 graus Celsius.

Algumas cerimônias devem ser executadas dentro da sala cofre para ter acesso a chaves criptográficas, adicionando mais complexidade e protocolos de segurança para a execução do processo. Este ambiente também pode ser considerado insalubre, com baixas temperaturas, ar não renovado em um local pequeno gerando alto nível de CO_2, e pouco espaço físico para uma boa disposição dos equipamentos e pessoas.

Como as cerimônias são executadas por humanos, estes que são sempre suscetíveis a cometer erros, e ainda pelo fato delas serem executadas em um ambiente insalubre aumentando a complexidade do processo, teve-se a motivação de desenvolver um aplicativo para servir como um guia de cerimônias, buscando ajudar no processo de execução de cerimônias ICP.

Com o desenvolvimento de um aplicativo, denominado de Sistema Assistente de Cerimônia ICP (SACI) para ser utilizado como guia das cerimônias, o processo de documentação da cerimônia pode ser separado do processo de execução da cerimônia. O aplicativo também

busca criar uma interface de usuário que minimize os erros humanos.

O SACI tem como proposta gerar um relatório eletrônico da versão final da ata, tornando o processo de armazenamento do resultado das cerimônias digital e passível da utilização de assinatura digital dos membros que participaram da cerimônia.

1.1 DESCRIÇÃO DO PROBLEMA

O Laboratório de Segurança em Computação (LabSEC) da Universidade Federal de Santa Catarina (UFSC) tem desenvolvido vários trabalhos na área de certificação digital e suas aplicações. Alguns destes trabalhos relacionam-se com a utilização e manutenção de uma ICP, que possui um conjunto de regras e métodos a serem seguidos; para utilizar e fazer manutenção em uma ICP é necessário seguir protocolos bem definidos, denominados cerimônias, que são executados por humanos, elas têm como propósito descrever os passos para execução de uma tarefa executada por um grupo específico, o histórico da execução dos passos também deve ser salvo para fins de auditoria. Além disso, um dos requisitos para este tipo de sistema é trabalhar em um ambiente físico totalmente seguro, por exemplo, dentro da sala cofre de um Centro de Certificação Digital (CCD). Algumas cerimônias são executadas dentro da sala cofre e outras fora, porém todas elas têm um nível de complexidade e indução ao erro. Dentro de um CCD existem condições insalubres de trabalho, sendo elas baixa temperatura, falta de oxigênio, não renovação de ar e espaço limitado. Na Coordenadoria de Certificação Digital da UFSC (CCDUFSC) existe a Infraestrutura de Chaves Públicas para Ensino e Pesquisa (ICPEdu) que é financiada pela Rede Nacional de Ensino e Pesquisa (RNP). Na ICPEdu existem os problemas citados intrínsecos a execução de cerimônias dentro de uma sala cofre, e também existem outros problemas, no formato de ata utilizado para acompanhar os passos da cerimônia e no processo de execução da mesma, estes problemas identificados são listados abaixo:

1. A utilização do mesmo monitor e computador para ler e preencher a ata e executar os passos da cerimônia;

2. Utilização de uma aplicação genérica, planilha eletrônica, para leitura e edição da ata;

3. Formato de entrada e saída da ata de cerimônia não garante restrição ao usuário;

4. Formato de entrada e saída não permite automatização e verificação dos dados;
5. Armazenamento do conteúdo digital é feito em CD;
6. A ata é arquivada em papel;
7. As assinaturas dos participantes da cerimônia são colhida em papel;
8. Baixa ergonomia no processo de execução de cerimônia;
9. Algumas cerimônias demandam bastante tempo para serem executadas.

No item 1 se o operador está executando uma cerimônia lendo a ata e executando os passos no mesmo ambiente faz com que a ambientação dele sobre o processo seja baixa, pois ele tem que ficar trocando de tela para ler o que vai executar. Sobre o item 2, uma aplicação genérica não permite adicionar funcionalidades e aumentar a escalabilidade do sistema para ajudar na execução de uma cerimônia. Sobre o item 3, a planilha eletrônica utilizada é totalmente editável, podendo ter qualquer campo rasurado pelo usuário, intencionalmente ou não. Sobre o item 4, o formato de entrada e saída atual é o próprio texto plano da planilha eletrônica, sem nenhuma linguagem de marcação não tornando o texto sintaticamente distinguível. Sobre os itens de 5, 6 e 7 o armazenamento do conteúdo digital gerado é feito em CD e da ata em papel com as assinaturas a mão, esse processo é totalmente ultrapassado, não segue as políticas de utilizar menos papel e não utiliza assinatura digital. Sobre o item 8, o notebook que contém os programas para utilizar na cerimônia fica disposto em um *rack* em que o operador utiliza em pé e a altura não pode ser adaptada para um usuário em específico. Sobre o item 9, quanto mais tempo é necessário para executar uma cerimônia maior a quantidade de CO2 no ambiente, por que um CCD possui o princípio de estanque, ou seja, não tem troca de ar.

1.2 MOTIVAÇÃO E JUSTIFICATIVA

A motivação e justificativa deste trabalho é tentar resolver os problemas encontrados no processo de execução de cerimônias ICPEdu através do desenvolvimento do aplicativo para dispositivo móvel. Com o desenvolvimento do SACI pode-se resolver os problemas relatados nos itens 1 a 4 na seção 1.1.

Para o item 1, a utilização de um *tablet* com o aplicativo desenvolvido faz com que o ambiente de leitura e edição da ata seja separado do ambiente de execução dos passos da cerimônia. Para o item 2, com o desenvolvimento de uma aplicação específica para ajudar na execução de uma cerimônia é possível obter uma escalabilidade no sistema, podendo desenvolver novas funcionalidades para ajudar na execução da cerimônia que não sejam relativas à leitura e edição da ata da cerimônia. Para o item 3, no aplicativo é possível restringir quais dados o operador da cerimônia vai poder editar na ata. Para o item 4, elaborar um arquivo de entrada e saída para a ata da cerimônia que utilize uma linguagem de marcação, como por exemplo XML, faz com que o texto seja sintaticamente distinguível permitindo que sejam desenvolvidas funcionalidades que analisem estes dados de entrada e saída.

Além disso o desenvolvimento de uma aplicação específica, permite diversos trabalhos futuros, que podem eventualmente resolver os problemas relatados nos itens 5 a 9 da seção 1.1 e trazer muitas outras melhorias no processo de execução de cerimônias ICPEdu. Mais sobre os trabalhos futuros pode ser encontrado na seção 5.1

1.3 OBJETIVOS

1.3.1 Objetivo Geral

Desenvolver um aplicativo Android para prover assistência na execução de cerimônias da Infraestrutura de Chaves Públicas (ICP), com o intuito de substituir o modelo atual de leitura e edição de ata em planilha eletrônica utilizado pela Infraestrutura de Chaves Públicas para Ensino e Pesquisa (ICPEdu).

1.3.2 Objetivos Específicos

- Analisar as atas das cerimônias executadas pela ICPEdu para determinar os requisitos da aplicação;
- Elaborar um novo formato para a entrada de dados da ata da cerimônia que seja interpretável pelo aplicativo;
- Elaborar um novo formato de saída de dados da ata da cerimônia que seja legível para o usuário final;

- Desenvolver o aplicativo para fornecer leitura e edição de atas utilizadas em uma cerimônia ICPEdu, provendo melhorias à leitura e edição de atas no processo de execução de uma cerimônia;

- Testar o uso do aplicativo na execução de uma cerimônia ICPEdu e fazer um comparativo com o modelo atual.

1.4 METODOLOGIA

Inicialmente foi feito uma análise da ata no formato antigo de planilha eletrônica, extraindo requisitos funcionais para desenvolver um protótipo e um arquivo XML com linguagem de marcação para a entrada e saída da aplicação ao executar uma cerimônia.

Após o término, do protótipo foi apresentado para os clientes do CCDUFSC, e a partir desta apresentação foram feitas várias rodadas incrementais e iterativas para mudar e adicionar requisitos de acordo com o *feedback* gerado nas reuniões pós término de rodada. Para terminar, foram realizadas cerimônias teste utilizando o aplicativo ao invés da planilha eletrônica com um grupo pequeno e específico de pessoas. Com este mesmo grupo foi aplicado uma avaliação qualitativa através de entrevista via painel dos envolvidos com o objetivo de analisar as mudanças no processo de execução de cerimônias, com a utilização do aplicativo, do ponto de vista do usuário que executa as cerimônias.

1.5 ORGANIZAÇÃO DOS CAPÍTULOS

A organização dos capítulos do trabalho será da seguinte forma:

- Introdução: Será descrito os problemas relativos a execução de cerimônias ICPedu, juntamente com os motivos que tornam a sua resolução importante e quais objetivos pretende-se alcançar com o desenvolvimento do SACI.

- Fundamentação teórica: São apresentados os conceitos de ICP, cerimônias e CCD. Neste capitulo é descrito um pouco sobre o processo da ICPEdu.

- Desenvolvimento: Neste capítulo possui a descrição das atividades, seguidas dos requisitos funcionais e não funcionais e a apresentação e explicação sobre o aplicativo tela por tela.

- Avaliação: Apresenta uma avaliação via painel dos envolvidos com um conjunto pequeno de pessoas que utilizaram o aplicativo para auxiliar em uma cerimônia ICPEdu.
- Conclusão: Principais pontos constatados durante a realização do trabalho, juntamente com os resultados obtidos.
  - Trabalhos futuros: Possíveis melhorias ou usos para o material criado que não entraram no escopo deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Para o melhor entendimento do trabalho neste capítulo são apresentados conceitos de segurança, seguido de uma apresentação e análise comentada pelo autor sobre a execução de cerimônias na ICPEdu.

2.1 INFRAESTRUTURA DE CHAVES PÚBLICAS (ICP)

Uma Infraestrutura de Chaves Públicas (ICP) é projetada para prover assistência a aplicações de criptografia de chave pública, seu principal objetivo é a criação e distribuição de certificados digitais e das Listas de Certificados Revogados (LCR). Uma ICP é dividida em 4 componentes que são projetados para realizar tarefas específicas (HOUSLEY; POLK, 2001), são eles:

- A Autoridade Certificadora (AC)
- A Autoridade de Registro (AR)
- O repositório
- O arquivo

Conforme (SILVÉRIO, 2011), os componentes de uma ICP respeitam uma estrutura hierárquica, concretizando uma cadeia de confiança em que o ponto mais alto é chamado de âncora de confiança. Na prática, a ICP baseia-se em um sistema de confiança que segue a seguinte filosofia: duas entidades confiam em uma terceira (âncora de confiança) para verificar e confirmar a identidade das mesmas.

2.1.1 AC e AR

Segundo (HOUSLEY; POLK, 2001), a autoridade certificadora (AC) é composta por hardware, software e pessoas que a operam, e é o componente principal de uma infraestrutura de chaves públicas. Ela é responsável pela emissão de certificados e LCRs, gerenciamento de publicação das informações sobre certificados revogados.

Segundo (CARLOS et al., 2010), a AC pode delegar determinadas funções a outras entidades com o intuito de minimizar a carga de tarefas, assim podendo construir uma hierarquia de ACs. Existem 2 tipos

de ACs na hierarquia, uma AC Raiz, que pode ter o seu certificado auto-assinado e outras ACs intermediárias que terão seus certificados assinados pela AC superior à ela. Outra delegação de tarefa bastante comum é a de delegar o processo de identificação dos usuários para uma entidade chamada Autoridade de Registro (AR).

Segundo (CARLOS et al., 2010), a Autoridade de Registro (AR) tem como função verificar o conteúdo de requisições de certificados que foram enviadas pelas ACs. Uma AC pode delegar esta tarefa para várias ARs, as ARs por sua vez podem desempenhar seu papel para várias ACs.

2.1.2 ICPEdu

Segundo (CARLOS et al., 2010), ICPEdu é uma infraestrutura de chaves públicas acadêmica mantida pela Rede Nacional de Ensino e Pesquisa (RNP), ela tem como objetivo criar uma cultura em certificação digital difundido pelas instituições acadêmica, permitindo o uso de certificação digital para autenticação de pessoas e equipamentos dentro das instituições.

Segundo (CARLOS et al., 2010), a ICPEdu começou em 2003 com o início do desenvolvimento do Sistema de Gestão de Certificados ICPEdu (SGCI), em 2006 começou o projeto para a criação de uma AC Raiz da ICPEdu utilizando o SGCI. Entre 2012 e 2013 este modelo está presente em algumas instituições de ensino superior e a RNP começou a propor novos modelos de integração com a Comunidade Acadêmica Federada (CAFe), com o intuito de ampliar a abrangência do ICPEdu.

Na Figura 1 temos um exemplo de cadeia de certificação da ICPEdu com algumas das suas autoridades certificadoras e de registro, a AC P1 emite certificados para as pessoas com a validade de 1 ano e a AC P5 também emite certificados para as pessoas porém com a validade de 5 anos.

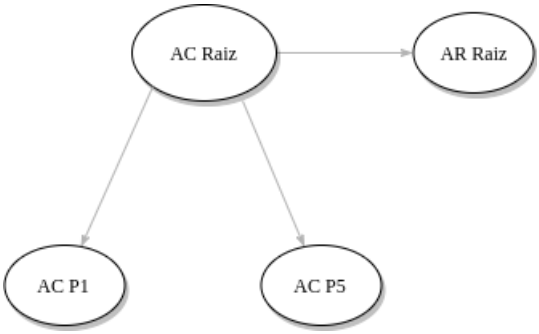


Figura 1 – Exemplo de cadeia de certificação da ICPEdu.

2.2 CERIMÔNIAS

Segundo (ELLISON, 2007), uma cerimônia é como um protocolo de rede, que possui alguns nodos humanos e os caminhos para a rede não são limitados a canais tradicionais de comunicação, desta forma as cerimônias incluem todos os protocolos de rede, mas principalmente todas as aplicações com suas interfaces gráficas de usuário e todas as instâncias do fluxo de trabalho.

O mesmo autor definiu que na comunidade de segurança, cerimônias para gerenciamento de chaves criptográficas nunca podem ser protocolos puros, então sempre devem ser cerimônias. Sem a definição e análise de cerimônias tais processos não poderiam ser especificados e analisados formalmente.

A definição simples de uma cerimônia esconde a sua complexidade. O problema surge ao tentar modelar o humano como um nodo em um protocolo com seus estados e máquinas de estados, ele recebe e emite mensagens, algumas vezes via interface de usuário em um computador e as vezes em uma comunicação humano a humano, feita por um computador ou não. Entretanto a entrada e saída das mensagens humanas é sempre sujeita a erro, provavelmente por causa de um mecanismo de conversão ou filtragem que não caracterizamos totalmente. (ELLISON, 2007)

2.3 CENTRO DE CERTIFICAÇÃO DIGITAL (CCD)

De acordo com (LUZ, 2008), Centro de Certificação Digital, é um tipo de *datacenter* especializado em certificação digital e tem como principal objetivo garantir a segurança física e digital de uma Infraestrutura de Chaves Públicas. Todos os requisitos impostos têm como objetivo preservar a chave privada da Autoridade Certificadora (AC). Alguns temas fundamentais abordados para garantir a segurança no CCD são a contingência, o princípio de estanqueidade, biometria, etc.

De acordo com (LUZ, 2008), um CCD em conformidade com os requisitos legais da ICP-Brasil deve seguir as normas e detalhes de ambiente descritos nesta seção.

Um CCD possui dois ambientes com características diferentes:

- Uma Área Administrativa onde são executados os processos administrativos;
- Uma Sala Cofre, que é uma célula estanque, sendo inteiramente imune a infiltrações de fumaça, gases corrosivos, etc, onde são executados os processos mais sensíveis.

Um CCD é baseado na seguinte premissa: "Tudo deve ser proibido a menos que seja permitido".

O CCD é composto por vários níveis de segurança, a segurança e o nível atribuído é proporcional à sensibilidade das informações armazenadas ou das cerimônias realizadas nos ambientes.

Um CCD é composto por pelo menos quatro níveis de acesso, e mais dois dependendo da essência das operações e negócios realizados, são exemplos de ambientes e níveis:

- Nível 1: Recepção;
- Nível 2: Corredor de acesso à sala cofre e sala de gerência;
- Nível 3: Sala de arquivo, sala dos administradores e corredor interno da sala cofre;
- Nível 4: Sala offline (OFF), de servidores críticos e sala de servidores online (ON);
- Nível 5: Cofres e *racks* nas salas ON e OFF;
- Nível 6: Cofres dentro dos cofres das salas ON e OFF.

Sobre a climatização da sala-cofre: A temperatura deve ser de 21 graus Celsius, podendo variar em mais ou menos dois graus, estas características devem ser suportadas com até 5 pessoas no interior da sala-cofre.

2.4 CERIMÔNIAS ICPEDU

Na ICPEdu são realizadas cerimônias para manter a ICP operacional. As cerimônias mais comuns são as de Emissão de Lista de Certificados Revogados (LCR). Estas cerimônias foram realizadas por membros do Laboratório de Segurança em Computação (LABSec) por vários anos e atualmente o processo operacional está sendo passado para a Coordenadoria de Certificação Digital da UFSC (CCDUFSC).

Uma cerimônia ICPEdu é composta geralmente por operadores e administradores. Os operadores devem utilizar de um computador para executar passos da cerimônia, utilizando os programas Sistema de Emissão de Certificados ICPEdu (SGCI) e o ASI-HSM, com o propósito de gerenciar uma ICP e de armazenar de maneira segura as chaves criptográficas respectivamente.

A próxima seção descreve o processo de execução de cerimônias ICPEdu.

2.4.1 Descrição do Processo de Execução de Cerimônias ICPEdu

Algumas cerimônias devem ser executadas dentro da sala cofre do CCD-UFSC, esta sala cofre possui temperaturas abaixo do normal do clima da região, não possui ciclagem de ar, após algumas horas dentro da sala cofre o nível de CO2 aumenta bastante, fazendo com que as pessoas que estão do cofre comecem a operar com dificuldade para raciocinar, aumentando a chance de erro ao executar uma cerimônia. Além disso, o operador que vai utilizar o computador com os programas para executar a cerimônia fica em pé utilizando o computador que está em um *rack*, em uma condição com baixa ergonomia.

Os passos de uma cerimônia ICPEdu estão descritas atualmente em planilhas eletrônicas, estas chamadas de atas de cerimônias, para cada tipo de processo envolvendo humanos e a ICPEdu deve existir um modelo de cerimônia descrito em planilha eletrônica. Ao executar uma cerimônia, anotações são feitas na ata pelos operadores, e o registro de

tudo que foi executado nesta cerimônia deve estar atrelado de alguma maneira a esta ata, como por exemplo os *logs* dos programas utilizados na cerimônia.

Ao terminar uma cerimônia com sucesso ou por falha, chamada de aborto, todo o conteúdo gerado é armazenado em CDs e a ata, impressa em papel, é assinada pelos membros participantes. Este conteúdo é então armazenado em um cofre e só é revisitado em um processo de auditoria.

A próxima seção apresenta um modelo de ata de cerimônia no formato de planilha eletrônica da ICPEdu e comentários sobre as práticas adotadas para o preenchimento da mesma.

2.4.2 Apresentação da Ata de Cerimônias ICPEdu

Conforme comentado anteriormente, as atas das cerimônias ICPEdu estão descritas no formato de planilha eletrônica, tal como nas figuras 2 e 3, nas quais é apresentado uma parte de uma ata de emissão de LCR. A partir desta ata será feita uma apresentação sobre os campos presentes nela e uma comparação com as atividades levantadas no Capítulo 3.

Na figura 2 podemos ver os seguintes campos com informações gerais, local, data, hora inicial, hora final e duração. Logo abaixo temos informações sobre os participantes da cerimônia com seus nomes completos, cargo, unidade, endereço eletrônico e assinatura. Abaixo temos um campo para informar quem é o redator da cerimônia, ou seja quem escreveu ela. Em seguida temos um campo para inserir qual o assunto principal da ata e observações gerais. No final temos os assuntos tratados com encaminhamento que foram considerados como o que deve ser feito após terminar a cerimônia.

Na figura 2, podemos notar que existe um campo assinatura para a linha de cada participante, ao final da cerimônia a ata é impressa e todos os membros assinam a caneta a ata da cerimônia.

ATA DE REUNIÃO

Local	Data	Hora		Duração
		Inicio	Fim	
Sala cofre da UFSC	09/04/2014	14:47	14:35	48min

Participantes

Nome	Cargo	Unidade	Endereço Eletrônico	Assinatura
	Operador do GOPAC	LABSEC		
	Auditor	UFSC		
	Auditor	UFSC		
José Ruas	Operador do GOPAC	LABSEC	jose.ruas@inf.ufsc.br	

Redator

Nome	Unidade	Endereço Eletrônico	Telefone

Obs.: Qualquer dúvida ou discordância em relação ao conteúdo deste documento, favor entrar em contato com o redator.

Assunto Principal

Emissão LCR #7 da AC Raiz ICPEDU v2

Observações

utilizou os smartcards adm#XXXX e oper#XXXX
José Luis Bressan Ruas utilizou os smartcards adm#XXXX e oper#XXXX

Assuntos Tratados com Encaminhamento	[1] Tipo	Responsável	Data Prevista Conclusão
Os backups serão guardados na sala cofre da UFSC	Ação		09/04/2014

1 Tipo: Corresponde à natureza dos assuntos tratados (Apresentação, Esclarecimento, Definição, Solicitação, Decisão, Pendência, Sugestão).

Figura 2 – Informações gerais da ata da cerimônia de emissão de LCR

Na Figura 3, pode-se ver os passos da cerimônia. Cada linha da tabela é considerado um passo, o qual possui uma breve descrição, um estado em que a cerimônia está ao entrar neste passo e o estado em que a cerimônia se encontrará ao sair do passo, o horário em que o passo foi

concluído, que é anotado manualmente pelo operador, e as observações que possuem informações mais específicas de como concluir o passo.

Conforme pode ser visto na figura 3, antes de se iniciar uma cerimônia os participantes e os requisitos devem ser confirmados, se algo estiver faltando ela é cancelada. Se em algum passo o operador não conseguir continuar a execução da cerimônia para concluí-la ela é abortada, é feito uma anotação no campo observações informando que a cerimônia foi abortada.

Descrição	Entradas	Saídas	Horário	Observações
Verificação de pré-requisitos básicos	Lista de materiais necessários a realização da cerimônia.	Confirmação de disponibilidade de todos os itens da lista de materiais.	14:47	Como parte da lista de materiais estão: smartcards dos operadores, flash usb ("pen drive") ou dispositivo similar, HSM, host hospedeiro, régua de energia, no-break para hos envelopes para smartcards, PINs e backup de PINs, cola, caneta, entre outros. Se pelo menos 1 pré-requisito básico não for satisfeito a cerimônia não pode ser realizada.
Abertura da Ata	Modelo de Ata	Ata aberta	14:47	
Conferência da Lista de Presença	Lista de Presença é igual a lista do documento de divulgação e apresentação, acrescida das pessoas adicionais que entraram na sala de cerimônia (presencialmente ou remotamente)	Lista de Presença Auditada	14:47	
Criar pasta na Maquina Virtual	Logado na maquina Virtual com o usuário icpedu do SO	Pasta Criada	15:05	Pasta criada no Desktop com o nome X
Download da LCR	LCR emitida	Arquivo lcr-ac-raiz.crl salvo na pasta X da maquina hospedeira. Feito no Logoff do SGCI	15:11	Exportar a LCR para um arquivo com o nome lcr-ac-raiz-v2.crl Copiar a LCR e colocá-la em um pendrive ou outra mídia.
Fechamento da Ata	Ata aberta	Ata datada, fechada e assinada	15:30	Preparar e imprimir uma cópia da ata do cerimonial. Solicita que todos os presentes assinem a ata. Os arquivos de backup deverão ser gravados em CD e Pendrive (USB flash e guardados no cofre disponível no IDC.
Fim da Cerimônia	Pen drive com todos os arquivos entregue ao auditor externo	Encerramento oficial da Cerimonia de Emissão de Certificado	15:34	Descrever como será mantido o sistema e o trabalho que será realizado em novas cerimônias de emissão de Certificados.
Relatório Final do cerimonial			15:34	Preparar o relatório final do cerimonial contendo as seguintes informações: a) atas; b) arquivos de backup gerados; c) Arquivos de logs do hsm e sgci; d) arquivo com o certificado; e) cópia deste documento; f) lista de presença desta etapa. (Enviar uma cópia deste relatório para o GT e para o Comit Assessor.)
Guarda dos back-ups	CD com os Backups do HSM e SGCI, Chave do Cofre	CD guardado no Cofre da sala cofre da UFSC	15:35	Transportar a mídia com o bakcup para o cofre

Figura 3 – Passos da ata da cerimônia de emissão de LCR

Normalmente o operador interage com a ata da seguinte maneira, ele lê as informações do passo em que ele deve executar, no próprio computador em que estão as aplicações utilizadas para a execução da cerimônia, então minimiza a aplicação que contém a ata e executa o passo, ao terminar a execução do passo ele anota o tempo na ata e

repete o processo para o próximo passo.

A próxima seção apresenta uma breve descrição do processo de abstração do conteúdo da ata em planilha eletrônica para o conteúdo da ata apresentada no aplicativo.

2.4.3 Abstração e Análise da Ata de Cerimônias ICPEdu

Com a análise de vários modelos de ata de cerimônias ICPEdu diferentes foi feito uma abstração para um modelo utilizado pelo aplicativo para ler os dados (entrada) e apresentar os dados após o término da cerimônia (saída). Além disso é levado em conta as práticas para anotar na ata e também as práticas de interação com a ata também descritas na seção anterior.

Na Figura 2, sobre as informações gerais foram consideradas como entrada para o aplicativo somente o local. Os campos data e hora são gerados automaticamente ao interagir com o aplicativo. Os participantes são cadastrados e salvos no aplicativo, ao iniciar uma cerimônia o navegador, quem está utilizando o *tablet* com o aplicativo, pode selecionar os participantes presentes na cerimônia e o seu cargo. As observações são geradas no processo de execução dos passos da cerimônia, se o operador quiser registrar alguma observação ele pode utilizar uma funcionalidade para fazer anotações sobre aquele passo. Os assuntos tratados com encaminhamento são dados de entrada também e ao finalizar a cerimônia existe uma tela no aplicativo chamada Fechamento de Ata onde devem ficar os estas informações.

Na Figura 3, os passos com seus respectivos campos de descrição, entrada, saída e observações são considerados dados de entrada para o aplicativo. O campo horário é gerado automaticamente ao interagir com o aplicativo. As três primeiras linhas da tabela da Figura X2 foram considerados informações necessárias para se iniciar uma cerimônia e ficam em uma tela chamada Abertura de Ata, as quatro últimas linhas são consideradas informações necessárias para se concluir uma cerimônia e ficam na tela chamada Fechamento de Ata. As demais linhas são passos normais apresentadas na tela Executar Passos da Cerimônia.

Esta abstração e análise foram feitas antes do início de processo de desenvolvimento do aplicativo, as informações mais precisas sobre o aplicativo e sobre as funcionalidades realmente desenvolvidas estão descritas no capítulo 3.

3 DESENVOLVIMENTO DO SACI

O Assistente de Cerimônias tem como objetivo principal servir como um leitor e editor de atas das cerimônias ICPEdu, substituindo o modelo anterior de planilha eletrônica. Para levantar as atividades e os requisitos iniciais do aplicativo foi feito uma análise do modelo anterior de ata. Com isso um protótipo foi desenvolvido com as atividades Escolher Cerimônia e Navegar na Cerimônia, basicamente o aplicativo inicial listava arquivos de ata de cerimônia e mostrava os passos em campos de texto e permitia avançar para o próximo passo.

O processo de desenvolvimento foi feito de maneira iterativa, a cada iteração o aplicativo era apresentado para os administradores do Centro de Certificação Digital da UFSC que apresentavam sugestões de novas funcionalidades e mudanças, assim novas atividades e requisitos eram identificados, após o desenvolvimento das mesmas uma nova iteração começava.

Além do desenvolvimento de funcionalidades baseadas na opinião dos usuários, funcionalidades foram desenvolvidas a fim de sanar os problemas encontrados na análise feita sobre as planilhas eletrônicas das atas e também no processo de execução de cerimônias ICPEdu.

Outras mudanças não relativas ao desenvolvimento do aplicativo também foram feitas no processo de execução de cerimônias ICPEdu. A utilização de um *tablet* juntamente com o aplicativo para ler e editar atas de cerimônias fez com que o ambiente que executa os passos da cerimônia fosse separado do ambiente que é feito o manuseio da ata. Além de tudo um suporte é utilizado para deixar o *tablet* fixo na sala em que a cerimônia vai ser executada, buscando melhorar a ergonomia no processo de execução de cerimônias.

A análise de requisitos apresentadas na próxima seção é relativo a versão final do aplicativo, sem entrar em detalhes de como foram as iterações e as mudanças do aplicativo ao decorrer do desenvolvimento.

3.1 ANÁLISE DE REQUISITOS

Através da análise do modelo anterior de ata e da utilização dela no processo de execução de uma cerimônia foram identificadas algumas atividades mais gerais a fim de auxiliar na identificação dos requisitos do aplicativo. A Tabela 1 descreve as atividades com suas descrições e artefatos de entrada e saída.

Tabela 1 – Levantamento de Atividades do SACI

Atividade	**Descrição**	**Entrada**	**Saída**
A1. Definir possíveis participantes.	Serve para definir os possíveis participantes de uma cerimônia, a lista de participantes que podem executar uma cerimônia.	Lista de usuários cadastrados no aplicativo.	Lista de usuários capacitados para executar uma cerimônia.
A2. Escolher cerimônia.	Serve para definir qual tipo de cerimônia será executada.	Lista de cerimônias cadastradas.	Os dados específicos da cerimônia escolhida.
A3. Configurar cerimônia.	Antes de iniciar uma cerimônia é necessário confirmar os requisitos e selecionar os participantes e sua função.	Detalhes da cerimônia, lista de participantes e lista de requisitos.	Requisitos confirmados e participantes selecionados.
A4. Navegar na cerimônia.	Serve para navegar entre os passos de uma cerimônia.	Um passo da cerimônia.	Um passo da cerimônia.
A5. Interagir na cerimônia.	O navegador da cerimônia pode registrar alguma ocorrência, pode ser através de foto ou texto.	Um passo da cerimônia.	Uma ação registrada na cerimônia, uma foto, anotação ou até mesmo o abortamento da cerimônia.
A6. Resultado da cerimônia.	Ao finalizar o processo de navegação da cerimônia, o navegador recebe instruções do que fazer para concluir a cerimônia e também informa se a cerimônia foi bem sucedida ou não.	As informações necessárias para finalizar a cerimônia e o status da mesma.	A leitura das informações necessárias pelo navegador.

Além das atividades descritas na tabela existe uma outra que não será desenvolvida no sistema, que é permitir os membros participantes de uma cerimônia assinarem a ata digitalmente comprovando a sua

participação na mesma, porém o sistema gera um arquivo digital com as informações da cerimônia e este pode ser assinado por todos os membros fora do escopo da aplicativo.

3.1.1 Requisitos Funcionais e Não-Funcionais

As funcionalidades da ferramenta foram identificadas através da análise de vários tipos de modelo de atas, entrevista com os clientes do aplicativo e com os operadores de cerimônias, conhecimentos prévios do autor como operador de cerimônias e também com a ajuda das atividades identificadas e apresentadas anteriormente. Os requisitos funcionais derivados das atividades são listados abaixo.

- RF1. Listar participantes.
- RF2. Excluir Participante.
- RF3. Adicionar Participante.
- RF4. Listar Cerimônias.
- RF5. Escolher cerimônia.
- RF6. Confirmar Participantes.
- RF7. Selecionar função do participante.
- RF8. Listar requisitos.
- RF9. Confirmar requisitos.
- RF10. Visualizar detalhes de uma cerimônia.
- RF11. Iniciar cerimônia.
- RF12. Listar passos de uma cerimônia.
- RF13. Avançar passo na cerimônia.
- RF14. Visualizar detalhes de um passo.
- RF15. Visualizar o status da cerimônia.
- RF16. Abortar Cerimônia.
- RF17. Bater Foto.

- RF18. Tomar notas.
- RF19. Finalizar cerimônia.
- RF20. Visualizar informações para terminar a cerimônia.

Além dos requisitos funcionais identificados a partir das atividades, outros requisitos são necessários para o funcionamento do aplicativo.

- RF22. Traduzir dados do XML de entrada para o modelo de objetos do aplicativo
- RF23. Traduzir dados dos participantes no arquivo *JavaScript Object Notation* (JSON) para o modelo de objetos do aplicativo
- RF24. Persistir participantes no arquivo em formato JSON
- RF25. Salvar Cerimônia no formato XML.
- RF26. Salvar Cerimônia no formato TXT.
- RF27. Criar Pasta para a cerimônia iniciada.
- RF28. Listar em páginas as configurações para iniciar a cerimônia.

Os requisitos não-funcionais identificados foram os seguintes:

- RNF1. O aplicativo deve ser desenvolvido para *tablet*, modelo Samsung Galaxy Tab E 9.6;
- RNF2. Desenvolvido para ser utilizado no CCD-UFSC em cerimônias ICPEdu.

3.2 MODELAGEM DO SISTEMA

A ferramenta é desenvolvida na linguagem de programação Java Android, que é uma das linguagens no paradigma de desenvolvimento para dispositivo móveis, possui uma vasta documentação auxiliando no desenvolvimento do projeto. A IDE utilizada para o desenvolvimento é a AndroidStudio, os dados são persistidos em arquivos, no formato JSON para a lista de possíveis participantes e XML para o relatório final da execução de uma cerimônia.

3.2.1 Design de Tela

Nesta seção são apresentadas as telas do SACI, ou seja é apresentado a interface gráfica do aplicativo, com comentários sobre o funcionamento de cada uma delas. Na figura 4 pode ser visto um fluxograma das telas do aplicativo.

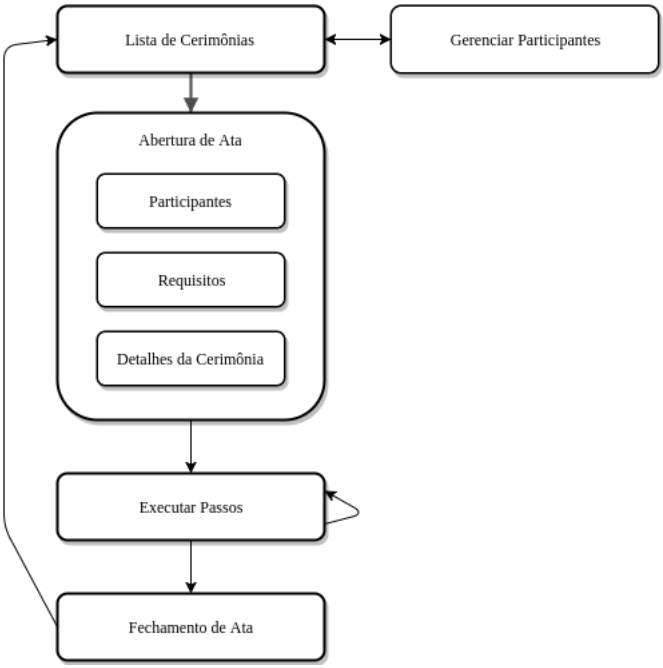


Figura 4 – Fluxograma das telas do SACI

Lista de Cerimônias

A visualização dos modelos de cerimônia é feita na tela inicial, que pode ser vista na figura 5, após o navegador, usuário que utiliza o aplicativo durante a cerimônia, abrir o aplicativo. Requisitos funcionais compreendidos nesta tela:

- RF4. Listar Cerimônias: Após o usuário abrir o aplicativo, o sistema mostra todas os modelos de cerimônias para possíveis execução de cerimônia.
- RF5. Escolher cerimônia: Ao tocar em uma das linhas na lista

o usuário é redirecionado para Abertura de Cerimônia, contendo informações sobre aquele modelo de cerimônia em específico.

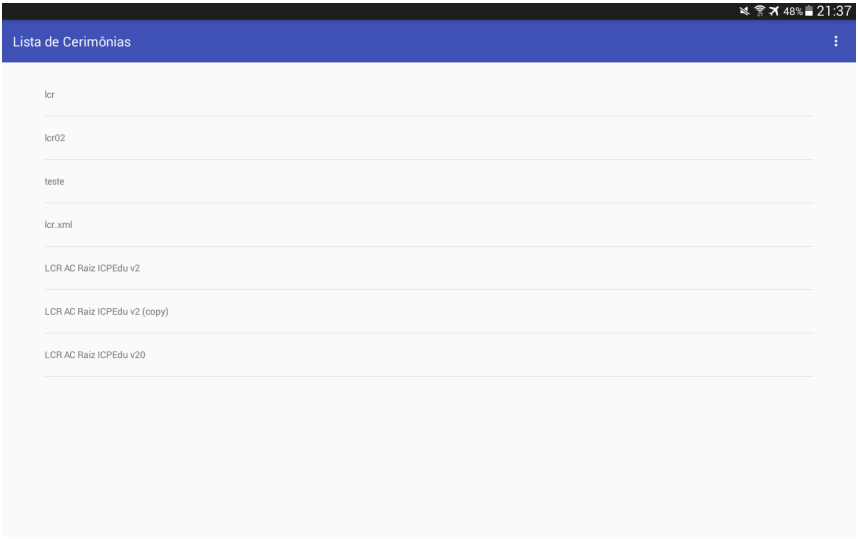


Figura 5 – Tela Lista de Cerimônias

Gerenciar Participantes

Na barra do menu, ao ir em opções no canto superior direito e selecionar a opção gerenciar participantes o usuário se depara com a tela da figura 6. Esta tela contém os seguintes requisitos funcionais descritos anteriormente:

- RF1. Listar participantes: O sistema lista os participantes já cadastrados que estão salvos no *participants.json*.
- RF2. Excluir Participante: Ao clicar no botão REMOVER, o sistema exibe uma tela pedindo a confirmação do usuário para confirmar a ação.
- RF3. Adicionar Participante: Os 3 campos com uma linha traçada embaixo na última linha da tabela são para preencher as informações relativas ao participante, ao clicar em um dos campos e preencher os dados corretamente o usuário clica no botão adicionar, o sistema mostra uma mensagem mostrando se o participante foi cadastrado com sucesso ou não.

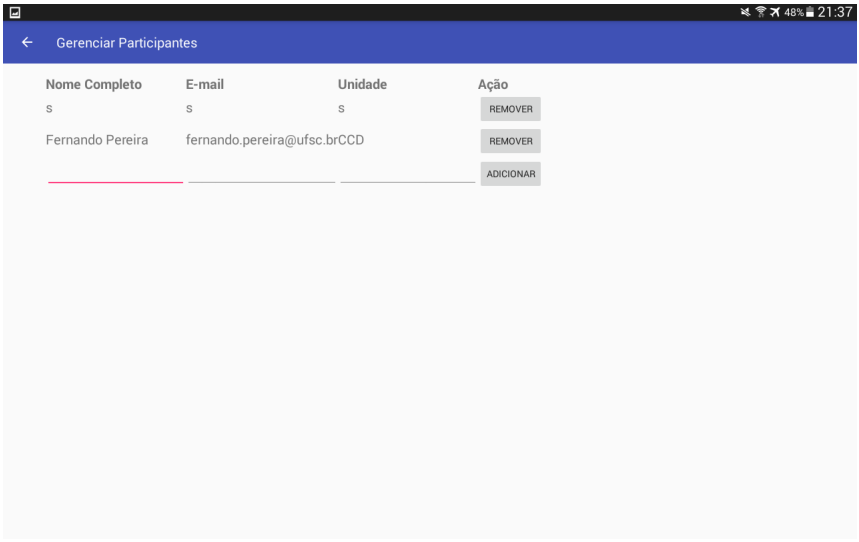


Figura 6 – Tela Gerenciar Participantes

Abertura de Ata

Antes da cerimônia ser iniciada oficialmente o usuário deve cumprir alguns requisitos para inicia-lá, na tela abertura de cerimônia é onde esse processo é feito, ela é dividida em 3 telas, Lista de Participantes, Lista de Requisitos e Detalhes da Cerimônia. Os requisitos que compreendem estas telas são os listados abaixo:

- RF6. Confirmar Participantes: Quando o usuário toca no botão CONFIRMAR PARTICIPANTES, o sistema informa se foi possível ou não, para ser possível pelo menos um dos participantes da lista devem ter alguma função selecionada, função está diferente de nenhuma, apenas os participantes com uma função selecionada são salvos como participantes desta cerimônia. Em caso de sucesso o usuário é direcionado pelo sistema para a Tela dos Requisitos.

- RF7. Selecionar Função do Participante: Ao tocar na seta para baixo localizada no final da linha de um dos participantes na figura 7, o sistema mostra as seguintes funções: Nenhuma, Auditor, Operador, Administrador, Navegador e Visitante, ao tocar em uma das opções a função do participante é modificada.

- RF8. Listar requisitos: Lista os requisitos para a execução desta cerimônia, que podem ser vistos na figura 8, os requisitos estão definidos no XML de entrada do modelo da cerimônia. O *checkbox* de cada requisito deve estar marcado para poder confirmar os requisitos.

- RF9. Confirmar requisitos: Ao clicar no botão CONFIRMAR REQUISITOS, se os *checkboxes* estarem marcados o sistema redireciona o usuário para a Tela dos Detalhes da Cerimônia, se não o sistema avisa que falta confirmar alguns requisitos.

- RF10. Visualizar detalhes de uma cerimônia: Mostra os detalhes da cerimônia, que podem ser vistos na figura 9 são: Nome completo do modelo da cerimônia, estado atual da cerimônia, quais os participantes confirmados e sua função, e se os requisitos foram ou não confirmados.

- RF11. Iniciar cerimônia: Ao clicar no botão INICIAR CERIMÔNIA o usuário é redirecionado para a próxima tela, somente se os requisitos estiverem confirmados e pelo menos um participante estiver confirmado.

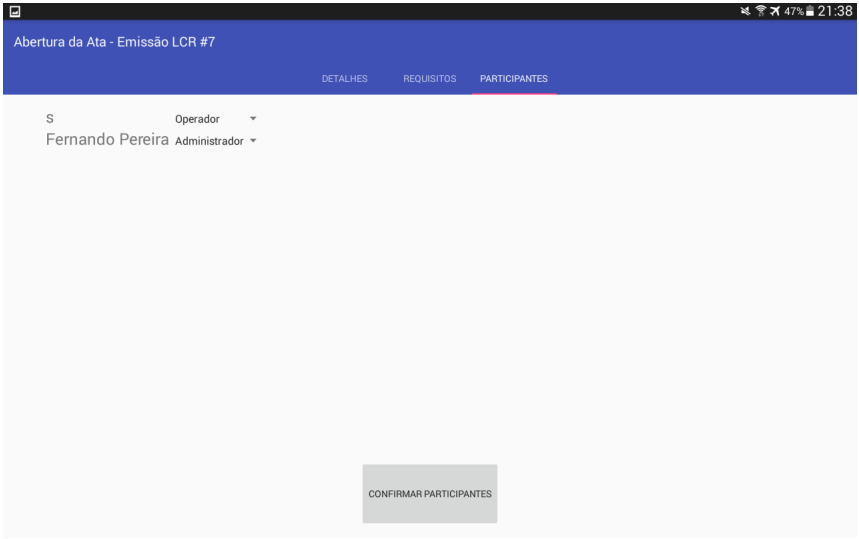


Figura 7 – Tela Participantes

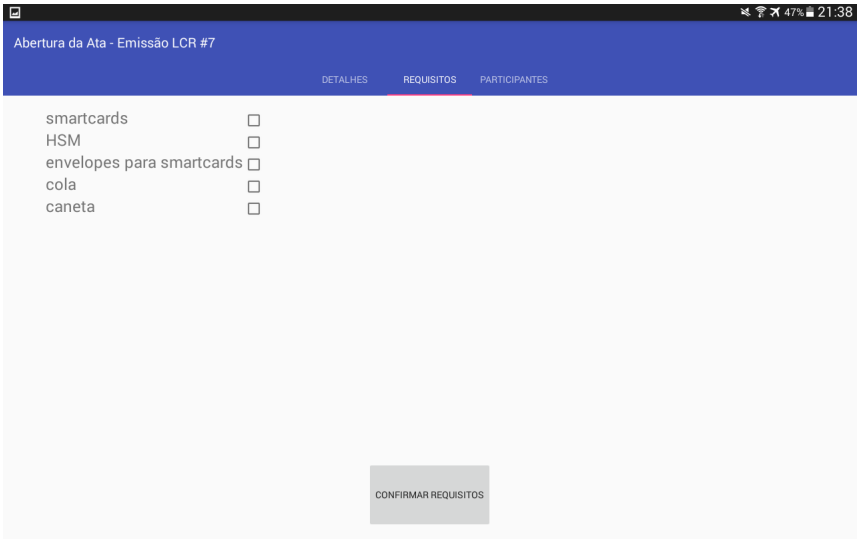


Figura 8 – Tela Requisitos

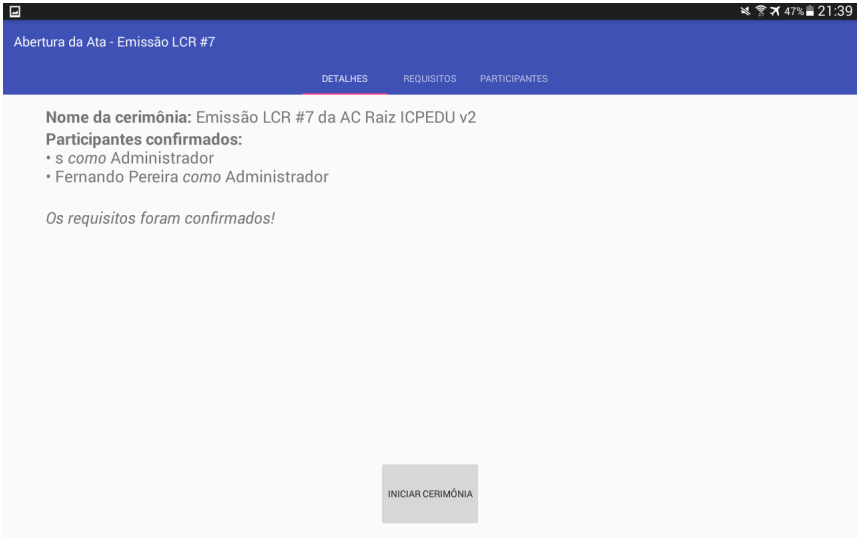


Figura 9 – Tela Detalhes da Cerimônia

Executar Passos

Na figura 10 pode ser visto a tela principal do aplicativo, neste ponto a cerimônia foi iniciada, e o usuário pode interagir de diversas maneiras para registrar informações da cerimônia e para ser auxiliado no processo de execução da cerimônia, se situando no contexto atual, qual passo ele deve executar e no contexto geral, em que passo ele está e quais são os passos necessários para concluir esta cerimônia. Os requisitos compreendidos por esta tela são os seguintes:

- RF12. Listar passos de uma cerimônia: No canto esquerdo o usuário pode ver o contexto geral da cerimônia que são basicamente todos os passos que ele deverá executar em ordem.
- RF13. Avançar passo na cerimônia: Ao tocar no botão AVANÇAR o usuário é direcionado para o próximo passo.
- RF14. Visualizar detalhes de um passo: No canto direito o usuário pode ver o contexto atual da cerimônia, que são os detalhes do passo que ele deve executar.
- RF15. Visualizar o status da cerimônia: No canto superior direito o usuário pode ver em que passo ele está atualmente e quantos faltam para ele completar a cerimônia.
- RF16. Abortar Cerimônia: Ao tocar no botão ABORTAR, o sistema exibirá uma janela pedindo para o usuário escrever um motivo para abortar a cerimônia, após o usuário escrever o motivo ele deve clicar no botão OK, então a cerimônia é abortada e o usuário é direcionado para a tela final Fechamento da Ata.
- RF17. Bater Foto: Ao selecionar a opção Bater Foto que está no menu de opções do canto superior direito, o usuário é direcionado para o aplicativo de bater fotos do dispositivo, após o usuário bater uma foto o usuário é redirecionado novamente para o aplicativo e a foto é salva como uma evidência da cerimônia.
- RF18. Tomar notas: Ao selecionar a opção Tomar notas que está no menu de opções do canto superior direito, o sistema abre uma janela para o usuário escrever as suas anotações sobre o passo atual, se o usuário clicar em OK as anotações são salvas.
- RF19. Finalizar cerimônia: Ao chegar no último passo da cerimônia o botão AVANÇAR é substituído pelo botão FINALIZAR na cor verde, ao clicar no botão o usuário é direcionado para a tela final Fechamento da Ata.

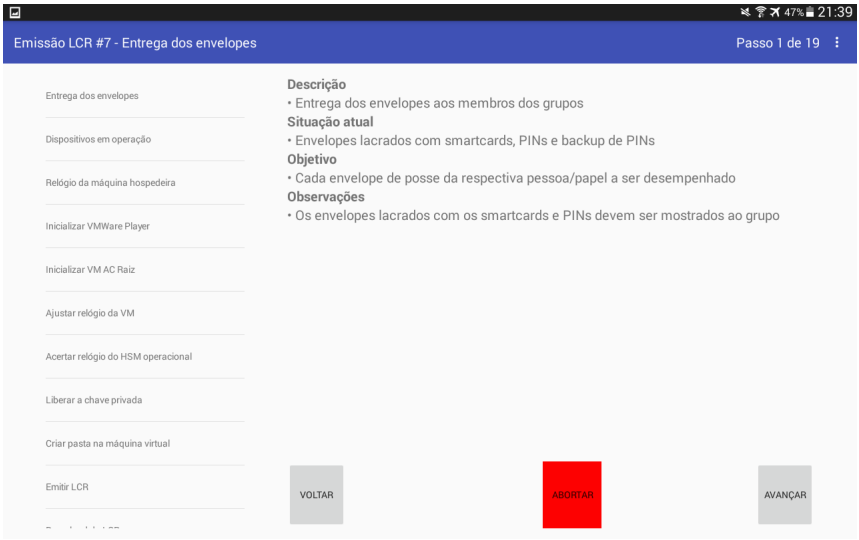


Figura 10 – Tela Executar Passos

Fechamento de Ata

Este é o último passo da execução de uma cerimônia, terminando o ciclo da aplicação, aqui o usuário pode verificar se a cerimônia foi abortada ou concluída com sucesso e também visualizar as recomendações necessárias para concluir e arquivar a cerimônia, a parte das recomendações necessárias não foi implementada.

O requisito que compreende esta tela é RF20. Visualizar informações para terminar a cerimônia. a tela pode ser vista na figura 11, onde o sistema mostra que a cerimônia foi abortada pelo motivo “hsm não liga” no passo 0.

Figura 11 – Tela Fechamento de Ata

3.2.2 Funcionalidades não visíveis

As funcionalidades não visíveis são aquelas em que o usuário não interage diretamente pelo aplicativo, anteriormente foram listados como funcionalidades não visíveis os requisitos funcionais 22 até o 28. Estes requisitos abrangem as entradas e saídas do aplicativo, a persistência de dados no meio da execução de uma cerimônia e a persistência dos dados em arquivos. Nesta seção serão comentadas somente as entradas e saídas do aplicativo.

Padrão do modelo de ata de uma cerimônia em XML

O modelo de ata de uma cerimônia em XML é uma entrada do aplicativo, lá estão contidas as informações sobre um tipo de cerimônia em específico. Na figura 12 contém um pedaço da ata da cerimônia de emissão de LCR 7 da AC Raiz ICPEDu v2 em XML.

As informações contidas na ata da cerimônia são mostradas para o usuário nas telas Execução dos Passos da Cerimônia e Abertura de Ata da Cerimônia.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<cerimony>
    <name>Emissão LCR #7 da AC Raiz ICPEDU v2</name>
    <short_name>Emissão LCR #7</short_name>
    <local>Sala cofre da UFSC</local>
    <requirements>
      <item>smartcards</item>
      <item>HSM</item>
      <item>envelopes para smartcards</item>
      <item>cola</item>
      <item>caneta</item>
    </requirements>
    <steps>
        <step name="Desligar a VM">
            <description>Desligar a VM</description>
            <input>Máquina Virtual ligada</input>
            <output>Maquina Virtual desligada</output>
        </step>
        <step name="Desligar a Maquina Hospedeira">
            <description>Desligar a Maquina Hospedeira</description>
            <input>Maquina Hospedeira ligada</input>
            <output>Maquina hospedeira desligada</output>
        </step>
    </steps>
</cerimony>
```

Figura 12 – Exemplo de ata de uma cerimônia em formato XML

Ata preenchida no formato de texto

A ata preenchida no formato de texto é uma saída do aplicativo, após o usuário terminar a cerimônia com sucesso ou abortando a mesma é gerado um documento de texto com as informações da ata da cerimônia executada.

O aplicativo salva diversas informações sobre a execução da cerimônia, elas podem ser vistas na figura 13. O campo Resultado, informa se a cerimônia foi abortada ou não, se ela foi abortada tem o motivo pelo qual ela foi abortada seguido do número do passo, do horário e da data do acontecimento. Na ata preenchida também é registrado se os requisitos foram confirmados e quais participantes participaram da cerimônia, mostrando o nome completo, função, instituição de vínculo e endereço eletrônico respectivamente.

Em um passo ao tocar no botão AVANÇAR o aplicativo salva o tempo em que esse passo foi concluído, se o operador utilizar as funcionalidades de Tomar Notas e Bater Foto estas também são salvas

no passo, nos campos Anotações e Evidências respectivamente.

Emissão LCR #7 da AC Raiz ICPEDU v2
Local: Sala cofre da UFSC
Resultado: A cerimônia foi abortada pelo motivo máquina hospedeira não liga no passo 1 às 19:12 do dia 4/5/2018.
Data de início: 4/5/2018
Horário de início: 19:10
Data de término: 4/5/2018
Horário de término: 19:12
Requisitos confirmados: smartcards, HSM, envelopes para smartcards, cola e caneta.
Participantes confirmados:
Zézinho - Auditor - UFSC - ze@ufsc.br

Passo 0: Entrega dos envelopes - 19:11
Descrição: Entrega dos envelopes aos membros dos grupos
Entrada: Envelopes lacrados com smartcards, PINs e backup de PINs
Saída: Cada envelope de posse da respectiva pessoa/papel a ser desempenhado
Observações: Os envelopes lacrados com os smartcards e PINs devem ser mostrados ao grupo
Anotações: mudamos x
Evidências: evidencia-passo0.jpg

Passo 1: Dispositivos em operação -
Descrição: Colocar a máquina hospedeira e o HSM em operação
Entrada: Máquina hospedeira pré-configurada. Máquina hospedeira e HSMs não bootados. Envelopes com usuário e senha de acesso ao SO. Envelopes com smartcards e PINs já distribuidos. Datashow, pen drive.
Saída: Máquina hospedeira e HSMs bootados
Observações: Os equipamentos com a AC Raiz e o HSM devem ser ligados e disponibilizados aos operadores. Login na máquina hospedeira com usuário e senha criado na cerimônia da AC Raiz.
Anotações:
Evidências:

Figura 13 – Exemplo de ata preenchida de uma cerimônia em formato TXT

Mais sobre as saídas do SACI

Além da saída de texto comentado anteriormente, outras saídas são geradas. Quando uma cerimônia é iniciada no aplicativo, ou seja, o aplicativo entra na Tela de Execução dos Passos da Cerimônia, uma pasta é criada seguindo o seguinte padrão: nomecurto-AAAA-MM-DD-HH-MM. Todas as saídas geradas pelo aplicativo durante a execução de uma cerimônia são salvas na pasta criada para esta cerimônia, outros

arquivos não gerados pelo aplicativo também são salvos nesta pasta. Em alguns passos o navegador recebe instruções para salvar arquivos gerados na execução da cerimônia, como por exemplo um arquivo contendo uma LCR.

O resultado final do processo de execução de uma cerimônia está todo salvo nesta pasta criada pelo aplicativo. As fotos, chamadas de evidências, também são salvas dentro desta pasta.

3.2.3 Implantação do SACI

O aplicativo foi instalado em um ativo da sala cofre do CCD da UFSC com o intuito de auxiliar cerimônias ICPEdu. O acesso ao ativo é restrito às pessoas que executam as cerimônias ICPEdu. Para a instalação e utilização de um ativo em um ambiente seguro foi feito uma cerimônia de formatação e configuração do *tablet*, e posteriormente instalação do aplicativo.

Essa cerimônia visa garantir que o *tablet* não contenha nenhuma aplicação maliciosa e deixar o mesmo OFFLINE, a cerimônia no formato XML pode ser vista em detalhes no anexo D.

4 AVALIAÇÃO

Este capítulo apresenta uma avaliação inicial do aplicativo e da mudança do processo de execução de cerimônias ICPEdu de forma a obter um *feedback* com relação a utilidade do conceito proposto por este trabalho. Para isto foi realizada uma avaliação com o objetivo de avaliar a completude, ergonomia, utilidade e usabilidade por meio de uma inspeção por um painel de envolvidos.

4.1 AVALIAÇÃO VIA PAINEL DE ENVOLVIDOS

Com o objetivo de avaliar se o aplicativo desenvolvido pode auxiliar na execução de cerimônias ICPEdu de maneira eficaz, foi realizada uma avaliação via painel de envolvidos.

4.1.1 Definição da avaliação

O objetivo da avaliação é de analisar o aplicativo Assistente de Cerimônias juntamente com a mudança no processo de execução de cerimônias ICPEdu, em termos de completude, ergonomia, utilidade e usabilidade.

Com a definição do objetivo da avaliação definido foram identificadas as perguntas para a análise. A partir destas perguntas mais gerais foram derivadas questões mais específicas referente ao objetivo a ser alcançado com o aplicativo juntamente com a mudança no processo de execução de cerimônias ICPEdu. Com a definição das questões um questionário foi elaborado com diversas questões de múltipla escolha contendo as alternativas “Sim”, “Não sei responder” e “Não”. O questionário está disponível no anexo A, este foi disponibilizado aos entrevistados por meio da ferramenta Google Forms.

Para a avaliação do aplicativo foi escolhido a cerimônia de Emissão de LCR que normalmente é a cerimônia mais executada na ICPEdu. Os operadores e administradores de cerimônias tiveram que utilizar o aplicativo passando pelo fluxo normal da aplicação descrito na figura Fluxo de telas do aplicativo.

4.1.2 Execução

A execução da avaliação do aplicativo e da mudança no processo de execução de cerimônias ICPEdu foi realizada por quatro membros que já participaram de cerimônias ICPEdu, três membros vinculados com o CCDUFSC e o outro com o LabSEC (Figura 14). Metade dos participantes já foram operadores em uma cerimônia de Emissão de LCR da ICPEdu v1 (Figura 15). A avaliação ocorreu no dia 10 de maio de 2018. Os dados completos coletados são apresentados no anexo B.

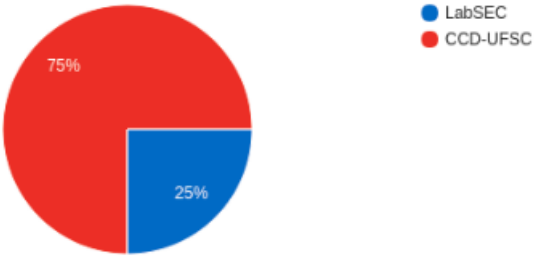


Figura 14 – Instituição de vínculo do entrevistado.

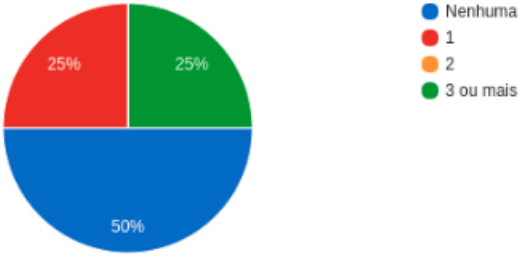


Figura 15 – Quantidade de cerimônias em que foi operador.

A limitação no número de participantes e na quantidade de testes efetuados está relacionado com o fato do ambiente seguro ser um lugar altamente restrito e de difícil acesso. A própria cerimônia executada para teste do aplicativo e das mudanças intrínsecas ao uso do aplicativo

na execução de cerimônias ICPEdu foi uma cerimônia real com impacto em uma das ACs da ICPEdu v1, a ata preenchida pelo SACI após a finalização da cerimônia pode ser visto no anexo C, a ata preenchida foi assinada digitalmente pelos participantes envolvidos na cerimônia.

4.1.3 Análise das Respostas

A análise das respostas foi feita de maneira estruturada guiado pelas perguntas de análise. As respostas para os 4 avaliadores são apresentadas em forma de tabela agrupando a porcentagem de respostas "Sim" e "Não, as respostas "Não sei responder" foram omitidas.

PA1. Completude: A funcionalidade do aplicativo está completa?

Tabela 2 – Análise de completude

Item do questionário	Sim	Não
O aplicativo fornece todas as funcionalidades que o modelo anterior de planilha eletrônica fornecia?	3	1
Você acha que existem funcionalidades que podem ser desenvolvidas no aplicativo para melhorar o auxílio na execução de cerimônias ICPEdu?	3	1

Pode-se observar que o aplicativo atende as necessidades para auxiliar como leitor e editor de atas para a execução de uma cerimônia ICPEdu. Porém, algumas colocações foram feitas pelos entrevistados, dentre elas poder editar os campos de um passo da cerimônia, voltar para o passo anterior e acrescentar novos passos. Além disso sobre a geração de ata automática no final de cerimônia comentaram que seria interessante ela ser gerada em formato PDF ou Látex, atualmente o aplicativo gera um arquivo TXT.

PA2. Ergonomia: A mudança do processo de execução de cerimônias ICPEdu dentro do ambiente seguro utilizando o aplicativo trouxe mais ergonomia ao processo em relação a planilha eletrônica?

Tabela 3 – Análise de ergonomia

Item do questionário	Sim	Não
Você acha que a separação do ambiente de execução da cerimônia (computador para executar os passos da cerimônia) do ambiente para a leitura e edição de ata trouxe melhorias ergonômicas para o processo de execução de cerimônias?	4	0

O resultado neste quesito foi totalmente positivo.

PA3. Utilidade: O aplicativo trouxe alguma utilidade para a execução de cerimônias ICPEdu?

Tabela 4 – Análise de utilidade

Item do questionário	Sim	Não
Você acha que o aplicativo é útil para a execução de cerimônias ICPEdu?	4	0
Você acha que o aplicativo é tão útil quanto o modelo anterior de planilha eletrônica?	3	0

O resultado nesse quesito também foi satisfatório, apenas um entrevistado marcou a opção "Não sei responder" para a segunda pergunta deste quesito.

PA4. Usabilidade: O aplicativo e a mudança no processo melhoraram a eficiência para a execução de cerimônias e a satisfação do usuário?

Tabela 5 – Análise de usabilidade

Item do questionário	Sim	Não
Você acha que a separação do ambiente de execução e de leitura e edição de ata reduziu o tempo gasto ao registrar e ler na ata?	3	1
Você se sente mais satisfeito ao utilizar o aplicativo ao invés da planilha eletrônica para auxiliar na execução de uma cerimônia ICPEdu?	2	0

O resultado para a primeira questão foi satisfatório, já para a segunda questão dois dos quatro entrevistados responderam que se sen-

tem satisfeitos, os outros dois alegaram que falta experiência na execução de cerimônias e que teriam que utilizar o aplicativo mais vezes para poder responder esta questão.

PA5. O uso do aplicativo melhorou a visibilidade do status da cerimônia?

Tabela 6 – Análise de visibilidade do status do sistema

Item do questionário	Sim	Não
Ao utilizar o aplicativo você sabe em que passo da cerimônia está e quantos faltam?	3	1
Ao utilizar o aplicativo você consegue visualizar e entender todos os passos que deverá executar para concluir a cerimônia?	3	1
Você acha que ficou mais fácil de se ambientar na cerimônia com a divisão da tela 1/3 para o contexto global (todos os passos) e 2/3 para o contexto atual (passo atual) do que no modelo anterior, todos os passos em formato de tabela?	4	0

Embora o aplicativo tenha recebido uma resposta positiva em relação a visibilidade do status da cerimônia, um dos entrevistados respondeu que não sabia em que passo da cerimônia estava e que também não conseguia visualizar e entender todos os passos que deveria executar para concluir a cerimônia.

4.1.4 Discussão

Por meio da avaliação realizada por um painel de envolvidos, foi possível notar que o aplicativo recebeu uma avaliação positiva nos critérios completude, ergonomia, utilidade e usabilidade, que foram todos os critérios analisados. De maneira geral, os entrevistados deixaram respostas positivas sobre o aplicativo.

Estas pequenas mudanças foram comentadas pela maioria dos entrevistados ao decorrer das perguntas, sendo as maiores críticas em relação ao tamanho da fonte dos textos no aplicativo, a falta do botão voltar passo e a flexibilidade na edição de ata, poder adicionar passos e editar o conteúdo de um passo. Uma sugestão foi dada para melhorar a visibilidade do status da cerimônia no aplicativo que é marcar os passos já finalizados no contexto global (lista de passos).

Novas funcionalidades foram criadas para acatar as sugestões e críticas dos entrevistados, entre elas, poder voltar passo, a edição do conteúdo de um passo e a marcação dos passos já finalizados no contexto global.

A funcionalidade de adicionar passo já havia sido notada em uma das iterações feitas no processo de desenvolvimento do aplicativo juntamente com os clientes, porém resolvemos deixar fora do escopo. Para suprimir a necessidade da criação de um novo passo o usuário pode simplesmente anotar estas novas informações em um passo já existente, de qualquer maneira a falta desta funcionalidade deixa o aplicativo menos flexível para a gama de cerimônias possíveis de ser criadas e acompanhadas com o aplicativo.

Esta avaliação inicial do aplicativo é de longe uma avaliação completa, visto que o número de avaliadores foi baixa e a execução de somente uma cerimônia utilizando o aplicativo também não nos ajuda a concluir muitas coisas. Conforme comentado anteriormente a quantidade de entrevistados e de testes é limitada pela restrição do ambiente seguro.

Com isto, então obteve se uma primeira avaliação do aplicativo e da mudança do processo de execução de cerimônias ICPEdu em relação aos critérios escolhidos, mas pela amostra pequena de entrevistados não se pode chegar em conclusões generalizadas sobre o aplicativo. Porém este pequeno grupo que avaliou o aplicativo é um grupo que participa de cerimônias no ambiente seguro, e como o resultado da pesquisa para essa pequena amostra de entrevistados foi positiva, pode-se considerar pelo menos que o modelo de leitura e edição de atas pelo aplicativo poderá substituir o modelo anterior de planilha eletrônica muito em breve. Também as sugestões e críticas apontadas pelos entrevistados ajudou na correção e desenvolvimento de novas funcionalidades para o aplicativo.

5 CONCLUSÃO

O objetivo geral deste projeto foi desenvolver um aplicativo para prover assistência na execução de cerimônias de ICP, com o intuito de substituir o modelo atual de leitura e edição de ata em planilha eletrônica utilizado pela ICPEdu, para isto inicialmente foi feito uma análise de vários modelos de ata de cerimônias ICPEdu do modelo atual. Com a análise do modelo atual foi possível desenvolver um aplicativo com melhorias em alguns aspectos e defeitos em outros, estas apontadas na avaliação dos entrevistados via questionário.

A avaliação dos usuários via questionário obteve um resultado positivo, porém por ser uma amostra muito pequena e não generalizada de avaliadores e também por ter sido feito somente um teste não se pode afirmar o desempenho do aplicativo sobre os critérios avaliados. Porém mesmo sem poder fazer afirmações sobre o aplicativo com as respostas obtidas, o fato do retorno das respostas serem positivas faz pensar que o aplicativo realmente será utilizado no ambiente seguro do CCD-UFSC e posteriormente em outros lugares que utilizam cerimônias.

Este trabalho entretanto nada mais é que a porta de entrada para outros trabalhos, ao utilizar uma aplicação de propósito específico para concluir uma tarefa ao invés de uma de propósito geral muitas coisas podem ser trabalhadas/implementadas para melhorar o aplicativo visando melhorias no processo de execução e operação de uma ICP, neste caso a ICPEdu.

Muitos trabalhos futuros podem ser feitos com a inserção do SACI como um novo componente no processo de execução de cerimônias ICPEdu, alguns deles estão descritos na seção 5.1.

5.1 TRABALHOS FUTUROS

O SACI tem como objetivo melhorar o ambiente no processo de execução de cerimônias ICP, algumas melhorias no processo de execução de cerimônias da ICPEdu já podem ser garantidas com as funcionalidades já desenvolvidas. Muitas outras áreas podem ser exploradas no desenvolvimento do SACI, entre elas:

1. Melhorar a usabilidade e a interface para o usuário;
2. Garantir segurança e armazenamento dos dados digitais;
3. Automatização dos dados gerados na execução de uma cerimônia;

4. Generalização dos aspectos de uma cerimônia;
5. Elaborar um design responsivo;
6. Melhorar o desempenho do aplicativo.

O item 1, a melhoria da interface para o usuário e da usabilidade do SACI pode garantir uma melhoria no processo de execução de cerimônias.

O item 2, os dados gerados por uma cerimônia devem ser armazenados de uma maneira segura garantindo integridade, não-repúdio e autenticidade. Com a utilização de assinatura digital estas propriedades podem ser garantidas, para isso deve-se estudar a viabilidade e os benefícios de desenvolver uma funcionalidade para o SACI assinar digitalmente os arquivos gerados em uma cerimônia.

Sobre o item 3, a utilização de entrada e saída para as cerimônias com a utilização de linguagem de marcação permite que seja desenvolvida diversas funcionalidades para analisar de maneira automatizada os dados da cerimônia, por exemplo, verificar se a cerimônia obteve sucesso através dos *logs* das aplicações e da saída do SACI, verificações de auditoria, análise de métricas de tempo, geração de métricas de tempo e sanitização do documento da ata.

Sobre o item 4, generalizar os aspectos de uma cerimônia, permite que o aplicativo seja utilizado para outros tipos de cerimônias como por exemplo as cerimônias do Sistema de Votação Helios e também pode tornar mais inclusivo a utilização do SACI para a utilização do mesmo em outras ICPs.

O item 5 permite com que o aplicativo seja utilizado em qualquer dispositivo móvel, porém como os dados que devem ser apresentados para executar uma cerimônia ocupam bastante espaço, não é qualquer dispositivo móvel que vai apresentar os dados necessários de uma maneira agradável.

Sobre o item 6, o aplicativo foi desenvolvido por um amador, em aprendizado, o SACI pode ter seu desempenho melhorado, com melhorias nos padrões de desenvolvimento, buscando melhorar a utilização de memória e melhorar o tratamento de erro. Além disso para o desenvolvimento de qualquer funcionalidade nova no SACI seria interessante fazer uma refatoração no código, recomeçando com algum padrão de desenvolvimento e utilizando um padrão de arquitetura de software, como o Modelo-Visão-Controle (MVC).

REFERĘNCIAS

CARLOS, M. et al. *Introdução a Infraestrutura de Chaves Públicas e Aplicações*. [S.l.]: Escola Superior de Redes RNP, 2010. 219 p.

ELLISON, C. Ceremony design and analysis. *Cryptology ePrint Archive*, 2007.

HOUSLEY, R.; POLK, T. *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. [S.l.]: John Wiley Sons, Inc, 2001. 327 p.

LUZ, C. P. d. *Centro de Certificação Digital - Construção, Administração e Manutenção*. [S.l.]: Editora Ciência Moderna Ltda, 2008. 338 p.

SILVÉRIO, A.

Análise e Implementação de um protocolo de gerenciamento de certificados — Universidade Federal de Santa Catarina, 2011.

ANEXO A – Questionário para avaliação do processo de execução de cerimônia

Survey - Avaliação do processo de execução de cerimônia utilizando o aplicativo como leitor e editor de ata

Você está sendo convidado(a) a participar da pesquisa de avaliação do processo de execução de cerimônia utilizando o aplicativo como leitor e editor de ata sendo realizada como parte do TCC do aluno José Luis Bressan Ruas orientado pelo Professor Doutor Jean Everson Martina. O objetivo da ferramenta é auxiliar na execução de cerimônias como leitor e editor de atas substituindo o modelo anterior de ata que utilizava uma planilha eletrônica.
Todos os dados coletados serão confidenciais de forma a assegurar a sua privacidade. Os resultados divulgados serão apresentados somente de forma acumulada não possibilitando a sua identificação.
Obrigado pela contribuição!
* Obrigatório

Questionário

Nome *

_____

Instituição de vínculo *

• LabSEC
• CCD-UFSC

Você já participou de quantas cerimônias de Emissão de LCR da ICPEdu? *

• Nenhuma
• 1
• 2
• 3 ou mais

Você já foi operador de quantas cerimônias de Emissão de LCR da AC UFSC v1? *

• Nenhuma
• 1
• 2
• 3 ou mais

Avaliação do processo de execução de cerimônia utilizando o aplicativo como leitor e editor de ata

O aplicativo fornece todas as funcionalidades que o modelo anterior de planilha eletrônica fornecia? *

• Sim
• Não sei responder
• Não

Se não, qual funcionalidade está faltando?

_____

Se não, a falta desta funcionalidade impediu ou atrapalhou a execução e conclusão da cerimônia?

• Sim
• Não sei responder
• Não

Você acha que existem funcionalidades que podem ser desenvolvidas no aplicativo para melhorar o auxílio na execução de cerimônias ICPEdu? *
• Sim
• Não sei responder
• Não
Se sim, quais mais deveriam ser desenvolvidas?
_____
Você acha que a separação do ambiente de execução da cerimônia (computador para executar os passos da cerimônia) do ambiente para leitura e edição de ata trouxe melhorias ergonométricas para o processo de execução de cerimônias? *
• Sim
• Não sei responder
• Não
Você acha que o aplicativo é útil para a execução de cerimônias ICPEdu? *
• Sim
• Não sei responder
• Não
Você acha que o aplicativo é tão útil quanto o modelo anterior de planilha eletrônica? *
• Sim
• Não sei responder
• Não
Você acha que a separação do ambiente de execução e de leitura e edição de ata reduziu o tempo gasto ao registrar e ler a ata? *
• Sim
• Não sei responder
• Não
Você se sente mais satisfeito ao utilizar o aplicativo ao invés da planilha eletrônica para auxiliar na execução de uma cerimônia ICPEdu? *
• Sim
• Não sei responder
• Não
Se não, porquê?
_____
Ao utilizar o aplicativo você sabe em que passo da cerimônia está e quantos faltam? *
• Sim
• Não sei responder
• Não
Ao utilizar o aplicativo você consegue visualizar e entender todos os passos que deverá executar para concluir a cerimônia? *
• Sim
• Não sei responder
• Não
Você acha que ficou mais fácil de se ambientar na cerimônia com a divisão da tela 1/3 para o contexto global (todos os passos) e 2/3 para o contexto atual (passo atual) do que no modelo anterior, todos os passos em formato de tabela? *
• Sim

• Não sei responder
• Não

Se não, você tem alguma sugestão de como melhorar a ambientação na cerimônia?

_____

Alguma sugestão de melhoria referente ao aplicativo Assistente de Cerimônias?

_____

Mais algum comentário?

_____

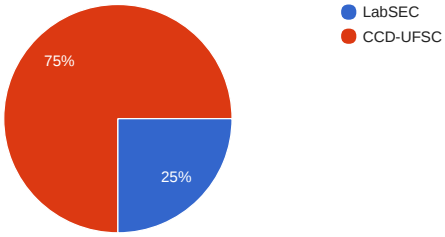
ANEXO B – Respostas das avaliações

Avaliação do processo de execução de cerimônia utilizando o aplicativo como leitor e editor de ata

4 respostas

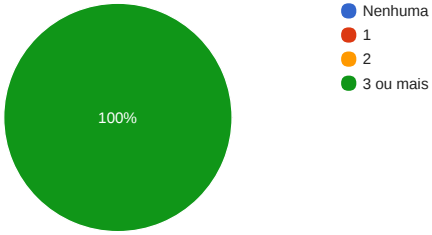
Instituição de vínculo?

4 respostas



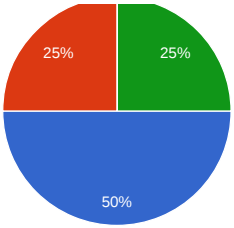
Você já participou de quantas cerimônias de Emissão de LCR da ICPEdu?

4 respostas



Você já foi operador de quantas cerimônias de Emissão de LCR da AC UFSC v1?

4 respostas

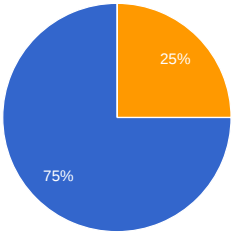


Nenhuma
1
2
3 ou mais

Avaliação do processo de execução de cerimônia utilizando o aplicativo como leitor e editor de ata

O aplicativo fornece todas as funcionalidades que o modelo anterior de planilha eletrônica fornecia?

4 respostas



Sim
Não sei responder
Não

Se não, qual funcionalidade está faltando?

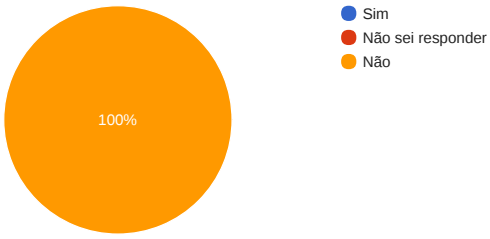
2 respostas

Um passo-a-passo mais detalhado, especifico para a cerimônia

Poder editar os campos evoltar,visualizar de forma fácil os passos da cerimônia e aumentar a fonte para melhor visualização.

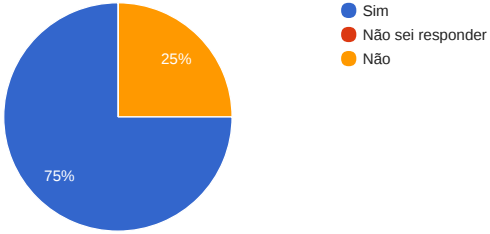
Se não, a falta desta funcionalidade impediu ou atrapalhou a execução e conclusão da cerimônia?

2 respostas



Você acha que existem funcionalidades que podem ser desenvolvidas no aplicativo para melhorar o auxílio na execução de cerimônias ICPEdu?

4 respostas



Se sim, quais mais deveriam ser desenvolvidas?

3 respostas

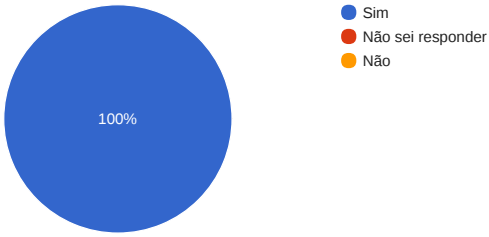
Deve-se poder editar os campos, acrescentar passos, acompanhar qual passo está sendo executado, aumentar fonte, assim como deve-se poder visualizar o horário que está sendo executado os passos, voltar e etc. O aplicativo ainda está bem incompleto.

Geração de ATA automática no final da cerimônia, em formato PDF (ou latex que permita gerar um PDF) para que seja possível colher a assinatura digital dos participantes. Este PDF poderia ter o cabeçalho e o rodapé personalizáveis.

Faltou aumentar o detalhamento das informações necessárias para o operador atuar com mais segurança no que está fazendo.

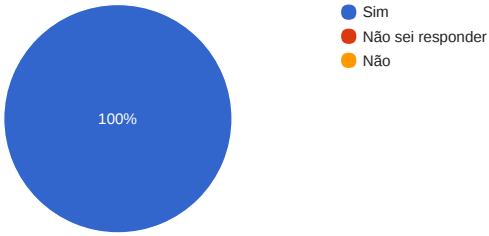
Você acha que a separação do ambiente de execução da cerimônia (computador para executar os passos da cerimônia) do ambiente para leitura e edição de ata trouxe melhorias ergonométricas para o processo de execução de cerimônias?

4 respostas



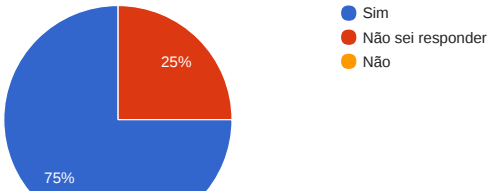
Você acha que o aplicativo é útil para a execução de cerimônias ICPEdu?

4 respostas



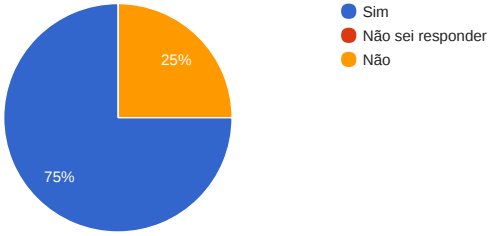
Você acha que o aplicativo é tão útil quanto o modelo anterior de planilha eletrônica?

4 respostas



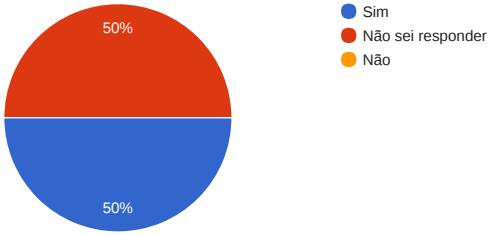
Você acha que a separação do ambiente de execução e de leitura e edição de ata reduziu o tempo gasto ao registrar e ler na ata?

4 respostas



Você se sente mais satisfeito ao utilizar o aplicativo ao invés da planilha eletrônica para auxiliar na execução de uma cerimônia ICPEdu?

4 respostas



Se não, porquê?

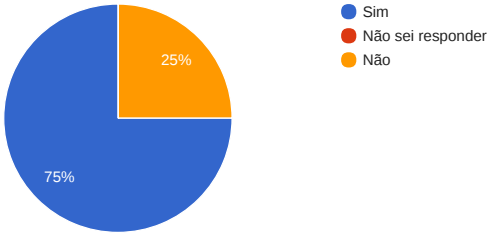
2 respostas

Utilizei apenas uma vez. É difícil tirar alguma conclusão em apenas uma vez com um aplicativo inacavado.

Falta para mim mais experiência na execução das cerimônias. Até agora tenho apenas acompanhado a execução das mesmas.

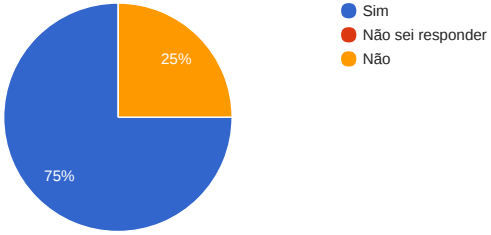
Ao utilizar o aplicativo você sabe em que passo da cerimônia está e quanto faltam?

4 respostas



Ao utilizar o aplicativo você consegue visualizar e entender todos os passos que deverá executar para concluir a cerimônia?

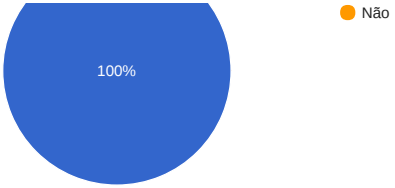
4 respostas



Você acha que ficou mais fácil de se ambientar na cerimônia com a divisão da tela 1/3 para o contexto global (todos os passos) e 2/3 para o contexto atual

(passo atual) do que no modelo anterior, todos os passos em formato de tabela?

4 respostas



Se não, você tem alguma sugestão de como melhorar a ambientação na cerimônia?

0 resposta

Ainda não há respostas para esta pergunta.

Alguma sugestão de melhoria referente ao aplicativo Assistente de Cerimônias?

3 respostas

São sugestões pontuais, mas que não invalidam o trabalho já realizado:
- Aumento do tamanho da fonte
- Ativar o botão de "retornar ao passo anterior"
- Tornar alguns campos e passos editáveis, de forma que o usuário final possa adaptar e customizar o software conforme suas necessidades específicas de uso (lembrando sempre que há mais de um tipo de cerimônia)

Deve-se finalizar todas as funções do aplicativo, deixar os campos editáveis, deve-se poder criar novos passos e aumentar a fonte. Além disso, deve ser possível saber em qual passo no contexto global encontra-se a cerimônia.

Os botões e fontes precisam ser maiores, o layout como um todo pode ser melhorado para oferecer melhor usabilidade ao participante que utiliza o aplicativo.

Mais algum comentário?

2 respostas

De maneira geral, como usuário leigo (sou da área administrativa e não da informática), achei a experiência de uso bastante positiva. Acredito que bastam pequenos ajustes para que o Aplicativo possa ser adotado nas cerimônias da ICPEdu.

Parabéns pelo trabalho! Estamos muito animados e ansiosos para começar a usar efetivamente a versão final do aplicativo. Acreditamos muito neste projeto e desejamos que siga sua pesquisa em trabalhos futuros nesta área. Conte com o apoio da Coordenadoria de Certificação Digital da Sala Cofre UFSC.

ANEXO C – Ata preenchida pelo SACI

Emissão LCR AC UFSC ICPEDU v1
Local: Sala cofre da UFSC
Resultado: A cerimônia terminou com sucesso
Data de início: 10/5/2018
Horário de início: 15:8
Data de término: 10/5/2018
Horário de término: 15:56
Requisitos confirmados: smartcards, HSM, envelopes para smartcards, cola e caneta.
Participantes confirmados:
Rick Lopes de Souza - Visitante - LabSEC - ricksouza87@gmail.com
Jose Luis Bressan Ruas - Visitante - UFSC - joseluis_ruas@gmail.com
Jean Everson Martina - Administrador - LabSEC - jean.martina@ufsc.br
Paulo Alcaraz - Operador - CCD-UFSC - paulo.alcaraz@ufsc.br
Fernando Pereira - Operador - CCD-UFSC - fernando.pereira@ufsc.br
Vilton Wronski - Administrador - CCD-UFSC - vilton.ricardo@ufsc.br
---
Passo 0: Entrega dos envelopes - 15:9
Descrição: Entrega dos envelopes aos membros dos grupos
Entrada: Envelopes lacrados com smartcards, PINs e backup de PINs
Saída: Cada envelope de posse da respectiva pessoa/papel a ser desempenhado
Observações: • Os envelopes lacrados com os smartcards e PINs devem ser mostrados ao grupo
Anotações:
Evidências:

Passo 1: Dispositivos em operação - 15:10
Descrição: Colocar a máquina hospedeira e o HSM em operação
Entrada: Máquina hospedeira pré-configurada. Máquina hospedeira e HSMs não bootados. Envelopes com usuário e senha de acesso ao SO. Envelopes com smartcards e PINs já distribuidos. Datashow, pen drive.
Saída: Máquina hospedeira e HSMs bootados.
Observações: • Os equipamentos com a AC UFSC e o HSM devem ser ligados e disponibilizados aos operadores. Login na máquina hospedeira com usuário e senha criado na cerimônia da AC Raiz.
Anotações:
Evidências:

Passo 2: Inicializar Máquina Hospedeira - 15:15
Descrição: Inicializar Máquina Hospedeira
Entrada: Iniciar servidor HP, seu monitor de vídeo e o HSM AC-UFSC n. de série#01113
Saída: HP e HSM iniciados
Observações: • servidor HP iniciar, escolher o usuário icpedu informando a senha, executar o comando startx para inicializar a interface gráfica do servidor
Anotações:
Evidências:

Passo 3: Iniciar Banco de Dados - 15:18
Descrição: Inicializar BD postgres
Entrada: Abrir terminal; comandos: sudo su (senha do root); Date MMDDhhmm; su - postgres /etc/init.d/postgres start

Saída: BD iniciado
Observações:
Anotações:
Evidências:

Passo 4: Acertar relógio do HSM operacional - 15:25
Descrição: Acertar relógio do HSM AC-UFSC
Entrada: ASI-HSM-Gui-1.1.jar botão direito; Opção Conectar ao HSM;
Endereço: 192.168.1.1 e Porta 5000; Opção Sistemas -> horário -> obter o
horário do HSM;
Saída: Relógio do HSM AC-UFSC acertado
Observações: • Não fechar o ASI-HSM, minimizar
Anotações:
Evidências:

Passo 5: Acertar relógio do Servidor HP - 15:25
Descrição: Verificar e caso necessário acertar o relógio do Servidor HP
Entrada: caso o horario esteja diferente do HSM utilizar o comando no
terminal Date MMDDhhmm
Saída: Relógio do servidor HP igual do HSM
Observações:
Anotações:
Evidências:

Passo 6: Liberar a chave privada - 15:28
Descrição: Liberar a chave privada da AC UFSC para uso
Entrada: Em HSM operacional Contas de usuários > Chaves > Carregar
chaves. Parametros chave-AC-UFSC-V1 , tempo infinito e uso infinito
Saída: Chave chave-AC-UFSC-V1 carregada.
Observações: • A chave privada será usada pelo SGCI para assinar o
certificado e gerar a LCR
Anotações:
Evidências:

Passo 7: Emitir a LCR - 15:29
Descrição: Emitir a LCR
Entrada: Abrir navegador firefox; http://localhost/sgci/lib/login.php ;
Autoridade Certificadora, entidade AC UFSC V1, função Operador, login
oper-ac-ufsc; LCRs->Gerar lista de certificados revogados
Saída: LCR emitida
Observações:
Anotações:
Evidências:

Passo 8: Download da LCR - 15:32
Descrição: Download da LCR e loggof do SGCI
Entrada: 112 no campo validade, informar senha e clicar Gerar LCR-
>Download; salvar o arquivo no desktop
Saída: Arquivo lcr-ac-*.crl salvo na pasta da cerimônia. Feito logoff no
SGCI
Observações:
Anotações:
Evidências:

Passo 9: Criar pasta - 15:33
Descrição: Criar pasta para a cerimônia
Entrada: Criar pasta no desktop com o nome DD-MM-AAAA e salvar o arquivo lcr dentro
Saída: Pasta criada
Observações:
Anotações:
Evidências:

Passo 10: Descarregar a chave da AC Raiz - 15:34
Descrição: Descarregar a chave da AC UFSC
Entrada: Interface do HSM no menu Accounts and Keys->Keys->Unload Key-> Unload key
Saída: Chave da AC Raiz descarregada
Observações:
Anotações:
Evidências:

Passo 11: Desligar o HSM - 15:35
Descrição: Desligar o HSM
Entrada: Interface do HSM no menu System -> Shutdown -> shutdown HSM
Saída: HSM desligado
Observações:
Anotações:
Evidências:

Passo 12: Parar o BD - 15:35
Descrição: Parar o banco de dados
Entrada: na janela do banco de dados rodando apertar control+c exit e enter
Saída: BD parado
Observações:
Anotações:
Evidências:

Passo 13: Copiar arquivos para o aplicativo - 15:51
Descrição: Copiar arquivos da pasta do desktop pro aplicativo
Entrada: Copiar os arquivos da pasta de cerimônia para a pasta criada no aplicativo para a cerimônia
Saída: Arquivos salvos na pasta do aplicativo, no terminal su, df para identificar o volume, rodar comando umount /dev/sda1
Observações: • conectar o tablet no computador com o cabo usb, ir na pasta cerimony-assistant/final/pasta-com-a-data-horario
Anotações:
Evidências:

Passo 14: Desligar o servidor - 15:55
Descrição: Desligar o servidor
Entrada: shutdown -h now
Saída: Servidor desligado
Observações:

Anotações:
Evidências:

Passo 15: Guardar novamente os smartcards - 15:56
Descrição: Guardar novamente os smartcards
Entrada: Envelopes e smartcards
Saída: Envelopes lacrados e assinados
Observações: • Os membros de todos os grupos devem colocar os smartcards em envelopes, com seus respectivos PINS, lacrá-los, datá-los e assiná-los
Anotações:
Evidências:

ANEXO D – Ata da Cerimônia de Implantação do SACI

anexos/Implantacao–do–App–SACI.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<cerimony>
    <name>Implantação Aplicativo Sistema Assistente
        de Cerimônias ICP</name>
    <short_name>Implantação do SACI</short_name>
    <local>CCD-UFSC Nível 2</local>
    <requirements>
      <item>APK</item>
      <item>caneta</item>
      <item>computador</item>
      <item>tablet</item>
      <item>cabo mini usb para conectar no
          computador</item>
    </requirements>
    <steps>
        <step name="Formatar o tablet">
            <description>Formatar o
                tablet</description>
            <input>Tablet</input>
            <output>Tablet formatado na versão de
                fabrica</output>
            <observation>Desligue o tablet, pressione
                ao mesmo tempo volume para cima + home
                + power, assim que aparecer a logo da
                samsung libere os botões
Selecione a opção wipe data/factory reset
Para navegar volume = setas, power = enter
    button</observation>
        </step>
        <step name="Fazer o tutorial de configuração
            inicial">
            <description>Fazer o tutorial de
                configuração do tablet</description>
            <input>Tablet formatado</input>
            <output>Tablet configurado</output>
            <observation>Selecionar o idioma e avançar
Selecionar uma rede wifi avançar - não énecessário
Definir hora manter GMT 00 avançar
Concordar com os termos avançar
Já tem conta google não, cadastrar agora não
Desmarcar todas as opções de localização avançar
Colocar nome e sobrenome no tablet
Pular conta samsung
Observações: da pra fazer sem rede wifi, o tablet
    não faz nenhuma atualização</observation>
        </step>
        <step name="Verificar se existe atualização">
```

```
          <description>Verificar se existe
              atualização da versão
              android</description>
          <input>Tablet configurado</input>
          <output>Tablet atualizado</output>
          <observation>Ir em Configurações -> Sobre o
              dispositivo -> Atualização de Software,
              desmarcar a opção atualização
              automática, tocar no botão atualizar
Observações: não énecessário se não cadastrar
    wifi</observation>
        </step>
        <step name="Esquecer rede wifi">
          <description>Esquecer rede wifi
              cadastrada</description>
          <input>Tablet atualizado</input>
          <output>rede esquecida</output>
          <observation>Configurações -> tocar sobre a
              rede cadastrada -> esquecer
              rede</observation>
        </step>
        <step name="Tablet em modo avião">
            <description>Deixar o tablet em modo avião
                e desativar wifi</description>
            <input>Tablet configurado</input>
            <output>Tablet em modo avião</output>
            <observation>Ir em Configurações, clicar
                na opção wifi desativando ela, e
                clicar no botão modo
                offline</observation>
        </step>
        <step name="Configurar o horario">
            <description>Deixar o tablet em GMT
                00</description>
            <input>Tablet em modo avião</input>
            <output>Tablet em GMT 00</output>
            <observation>Configurações -> Data e Hora
                -> Definir Hora
Observações: Para ajustar uma hora verifique um
    fonte de tempo confiável brasileira
            </observation>
        </step>
        <step name="Conectar o tablet em um
            computador">
            <description>Conectar o tablet em um
                computador offline</description>
            <input>Tablet configurado</input>
            <output>Tablet conectado com sucesso em um
                computador offline</output>
```

```
            <observation>Conectar tablet com cabo mini
                USB no computador</observation>
        </step>
        <step name="Criar pastas">
            <description>Criar pastas no
                tablet</description>
            <input>Tablet conectado com sucesso em um
                computador offline</input>
            <output>Pastas criadas</output>
            <observation>Abrir a estrutura de arquivos
                do tablet no computador
Na raiz da estrutura criar pasta ceremony-assistant
Dentro dessa pasta criar pasta final e new
            </observation>
        </step>
        <step name="Copiar atas de cerimonias">
          <description>Copiar cerimonias</description>
          <input>Tablet com as pastas criadas</input>
          <output>Tablet com os modelos de atas em
              xml salvas</output>
          <observation>Se já existirem atas de
              cerimonias escritas em XML copia-las
              para dentro da pasta new</observation>
      </step>
      <step name="Copiar APK do aplicativo SACI">
        <description>Copiar APK do aplicativo
            SACI</description>
        <input>Tablet conectado com sucesso em um
            computador offline que contenha o APK do
            aplicativo SACI</input>
        <output>Tablet com o APK salvo em sua
            estrutura de arquivos</output>
        <observation>No computador localizar o
            arquivo do APK do aplicativo SACI
Apertar Control+c
No computador localizar a estrutura de arquivos do
    Tablet, clicar em entrar na pasta
Apertar Control+v
        </observation>
    </step>
      <step name="Instalar o APK">
        <description>Instalar o aplicativo atráves do
            APK</description>
        <input>Tablet com as pastas criadas</input>
        <output>Aplicativo instalado</output>
        <observation>No icone Meus arquivos -> pasta
            onde se localiza o apk, tocar no arquivo,
            clicar em instalar
```

```
Observações: Énecessário garantir permissão para
    instalar apks desconhecidos, o próprio tablet te
    redirecionará para lá, marcar o checkbox e
    continuar</obsrvation>
      </step>
      <step name="Verificar o aplicativo">
        <description>Verificar se o aplicativo esta
            funcionando</description>
        <input>APK instalado com sucesso no
            tablet</input>
        <output>Aplicativo aberto com sucesso</output>
        <observation>
Na barra de aplicativos, tocar no aplicativo
    assistente de cerimonias
Observações: Énecessário cadastrar um participante
    antes de tocar em um modelo de ata de cerimônia,
    o aplicativo da crash com 0 participantes
    cadastrados</obsrvation>
      </step>
    </steps>
</cerimony>
```

ANEXO E – Artigo

Sistema Assistente de Cerimônias ICP (SACI)

José Luis Bressan Ruas[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil
joseluis.ruas@gmail.com

Resumo. *Em uma Infraestrutura de Chaves Públicas (ICP) é necessário que pessoas executem alguns procedimentos, estes chamados de cerimônias. Algumas destas cerimônias são executadas dentro de uma sala cofre de um Centro de Certificação Digital (CCD). Uma sala cofre é um ambiente estanque, onde não existe a troca de oxigênio, com o tempo os níveis de CO2 vão aumentando. A proposta deste trabalho é o desenvolvimento de um aplicativo Android para servir como um Sistema Assistente de Cerimônias ICP (SACI). Após o desenvolvimento o SACI foi avaliado nos termos de completude, ergonomia, utilidade e usabilidade, porém não se pode obter conclusões por causa da quantidade de entrevistados e do número de testes.*

1. Introdução

A certificação digital pode ser utilizada para prover segurança digital. Para validar um certificado, este precisa pertencer a uma cadeia de certificação conhecida como Infraestrutura de Chaves Públicas (ICP). Uma ICP deve ser gerenciada por pessoas para controlar o ciclo de vida dos certificados digitais, portanto para gerenciar uma ICP é necessário executar cerimônias. Segundo Ellison, uma cerimônia é um protocolo de rede, que possui alguns nodos humanos e a troca de informação não está limitada a canais tradicionais de comunicação, incluindo desta forma todas as aplicações com suas interfaces gráficas e todas as instâncias do fluxo de trabalho. Além disso a entrada e saída das mensagens humanas é sempre suscetível a erro.

Atualmente as cerimônias apresentam diversos tipos de problemas acarretados por diferentes motivos, como a execução realizada por humanos; possuir muitos passos e sem a garantia do sequenciamento dos mesmos; serem executadas em softwares robustos, com muitas funcionalidades. Algumas das cerimônias devem ser executadas dentro de um Centro de Certificação Digital (CCD).

Segundo Luz, um CCD é um tipo de *datacenter* especializado em certificação digital e tem como principal objetivo garantir a segurança física e digital de uma ICP. Um CCD possui uma sala cofre, que é uma célula estanque, ou seja, não tem troca de ar. O ambiente também possui baixas temperaturas em torno de 21 graus Celsius.

Algumas cerimônias devem ser executadas dentro da sala cofre para ter acesso a chaves criptográficas, adicionando mais complexidade e protocolos de segurança para a execução do processo. Este ambiente também pode ser considerado insalubre, com baixas temperaturas, ar não renovado em um local pequeno gerando alto nível de CO2, e pouco espaço físico para uma boa disposição dos equipamentos e pessoas.

Como as cerimônias são executadas por humanos, estes que são sempre suscetíveis a cometer erros, e ainda pelo fato delas serem executadas em um ambiente insalubre aumentando a complexidade do processo, teve-se a motivação de desenvolver um

aplicativo para servir como um guia de cerimônias, buscando ajudar no processo de execução de cerimônias ICP.

Com o desenvolvimento de um aplicativo, denominado de Sistema Assistente de Cerimônia ICP (SACI) para ser utilizado como guia das cerimônias, o processo de documentação da cerimônia pode ser separado do processo de execução da cerimônia. O aplicativo também busca criar uma interface de usuário que minimize os erros humanos.

O SACI tem como proposta gerar um relatório eletrônico da versão final da ata, tornando o processo de armazenamento do resultado das cerimônias digital e passível da utilização de assinatura digital dos membros que participaram da cerimônia.

2. Desenvolvimento

O Assistente de Cerimônias tem como objetivo principal servir como um leitor e editor de atas das cerimônias ICPEdu, substituindo o modelo anterior de planilha eletrônica. Para levantar as atividades e os requisitos iniciais do aplicativo foi feito uma análise do modelo anterior de ata. Com isso um protótipo foi desenvolvido com as atividades Escolher Cerimônia e Navegar na Cerimônia, basicamente o aplicativo inicial listava arquivos de ata de cerimônia e mostrava os passos em campos de texto e permitia avançar para o próximo passo.

O processo de desenvolvimento foi feito de maneira iterativa, a cada iteração o aplicativo era apresentado para os administradores do Centro de Certificação Digital da UFSC que apresentavam sugestões de novas funcionalidades e mudanças, assim novas atividades e requisitos eram identificados, após o desenvolvimento das mesmas uma nova iteração começava.

Além do desenvolvimento de funcionalidades baseadas na opinião dos usuários, funcionalidades foram desenvolvidas a fim de sanar os problemas encontrados na análise feita sobre as planilhas eletrônicas das atas e também no processo de execução de cerimônias ICPEdu.

Outras mudanças não relativas ao desenvolvimento do aplicativo também foram feitas no processo de execução de cerimônias ICPEdu. A utilização de um *tablet* juntamente com o aplicativo para ler e editar atas de cerimônias fez com que o ambiente que executa os passos da cerimônia fosse separado do ambiente que é feito o manuseio da ata. Além de tudo um suporte é utilizado para deixar o *tablet* fixo na sala em que a cerimônia vai ser executada, buscando melhorar a ergonomia no processo de execução de cerimônias.

A análise de requisitos apresentadas na próxima seção é relativo a versão final do aplicativo, sem entrar em detalhes de como foram as iterações e as mudanças do aplicativo ao decorrer do desenvolvimento.

2.1. Requisitos Funcionais e Não-Funcionais

As funcionalidades da ferramenta foram identificadas através da análise de vários tipos de modelo de atas, entrevista com os clientes do aplicativo e com os operadores de cerimônias, conhecimentos prévios do autor como operador de cerimônias e também com a ajuda das atividades identificadas e apresentadas anteriormente. Os requisitos funcionais derivados das atividades são listados abaixo.

- RF1. Listar participantes.

- RF2. Excluir Participante.
- RF3. Adicionar Participante.
- RF4. Listar Cerimônias.
- RF5. Escolher cerimônia.
- RF6. Confirmar Participantes.
- RF7. Selecionar função do participante.
- RF8. Listar requisitos.
- RF9. Confirmar requisitos.
- RF10. Visualizar detalhes de uma cerimônia.
- RF11. Iniciar cerimônia.
- RF12. Listar passos de uma cerimônia.
- RF13. Avançar passo na cerimônia.
- RF14. Visualizar detalhes de um passo.
- RF15. Visualizar o status da cerimônia.
- RF16. Abortar Cerimônia.
- RF17. Bater Foto.
- RF18. Tomar notas.
- RF19. Finalizar cerimônia.
- RF20. Visualizar informações para terminar a cerimônia.

Além dos requisitos funcionais identificados a partir das atividades, outros requisitos são necessários para o funcionamento do aplicativo.

- RF22. Traduzir dados do XML de entrada para o modelo de objetos do aplicativo
- RF23. Traduzir dados dos participantes no arquivo *JavaScript Object Notation* (JSON) para o modelo de objetos do aplicativo
- RF24. Persistir participantes no arquivo em formato JSON
- RF25. Salvar Cerimônia no formato XML.
- RF26. Salvar Cerimônia no formato TXT.
- RF27. Criar Pasta para a cerimônia iniciada.
- RF28. Listar em páginas as configurações para iniciar a cerimônia.

Os requisitos não-funcionais identificados foram os seguintes:

- RNF1. O aplicativo deve ser desenvolvido para *tablet*, modelo Samsung Galaxy Tab E 9.6;
- RNF2. Desenvolvido para ser utilizado no CCD-UFSC em cerimônias ICPEdu.

3. Avaliação

O objetivo da avaliação é de analisar o aplicativo Assistente de Cerimônias juntamente com a mudança no processo de execução de cerimônias ICPEdu, em termos de completude, ergonomia, utilidade e usabilidade.

Com a definição do objetivo da avaliação definido foram identificadas as perguntas para a análise. A partir destas perguntas mais gerais foram derivadas questões mais específicas referente ao objetivo a ser alcançado com o aplicativo juntamente com a mudança no processo de execução de cerimônias ICPEdu. Com a definição das questões um questionário foi elaborado com diversas questões de múltipla escolha contendo as alternativas “Sim”, “Não sei responder” e “Não”. O questionário foi disponibilizado aos entrevistados por meio da ferramenta Google Forms.

Para a avaliação do aplicativo foi escolhido a cerimônia de Emissão de LCR que normalmente é a cerimônia mais executada na ICPEdu. Os operadores e administradores de cerimônias tiveram que utilizar o aplicativo passando pelo fluxo normal da aplicação descrito na figura Fluxo de telas do aplicativo.

3.1. Discussão

Por meio da avaliação realizada por um painel de envolvidos, foi possível notar que o aplicativo recebeu uma avaliação positiva nos critérios completude, ergonomia, utilidade e usabilidade, que foram todos os critérios analisados. De maneira geral, os entrevistados deixaram respostas positivas sobre o aplicativo.

Estas pequenas mudanças foram comentadas pela maioria dos entrevistados ao decorrer das perguntas, sendo as maiores críticas em relação ao tamanho da fonte dos textos no aplicativo, a falta do botão voltar passo e a flexibilidade na edição de ata, poder adicionar passos e editar o conteúdo de um passo. Uma sugestão foi dada para melhorar a visibilidade do status da cerimônia no aplicativo que é marcar os passos já finalizados no contexto global (lista de passos).

Novas funcionalidades foram criadas para acatar as sugestões e críticas dos entrevistados, entre elas, poder voltar passo, a edição do conteúdo de um passo e a marcação dos passos já finalizados no contexto global.

A funcionalidade de adicionar passo já havia sido notada em uma das iterações feitas no processo de desenvolvimento do aplicativo juntamente com os clientes, porém resolvemos deixar fora do escopo. Para suprimir a necessidade da criação de um novo passo o usuário pode simplesmente anotar estas novas informações em um passo já existente, de qualquer maneira a falta desta funcionalidade deixa o aplicativo menos flexível para a gama de cerimônias possíveis de ser criadas e acompanhadas com o aplicativo.

Esta avaliação inicial do aplicativo é de longe uma avaliação completa, visto que o número de avaliadores foi baixa e a execução de somente uma cerimônia utilizando o aplicativo também não nos ajuda a concluir muitas coisas. Conforme comentado anteriormente a quantidade de entrevistados e de testes é limitada pela restrição do ambiente seguro.

Com isto, então obteve se uma primeira avaliação do aplicativo e da mudança do processo de execução de cerimônias ICPEdu em relação aos critérios escolhidos, mas pela amostra pequena de entrevistados não se pode chegar em conclusões generalizadas

sobre o aplicativo. Porém este pequeno grupo que avaliou o aplicativo é um grupo que participa de cerimônias no ambiente seguro, e como o resultado da pesquisa para essa pequena amostra de entrevistados foi positiva, pode-se considerar pelo menos que o modelo de leitura e edição de atas pelo aplicativo poderá substituir o modelo anterior de planilha eletrônica muito em breve. Também as sugestões e críticas apontadas pelos entrevistados ajudou na correção e desenvolvimento de novas funcionalidades para o aplicativo.

4. Conclusão

O objetivo geral deste projeto foi desenvolver um aplicativo para prover assistência na execução de cerimônias de ICP, com o intuito de substituir o modelo atual de leitura e edição de ata em planilha eletrônica utilizado pela ICPEdu, para isto inicialmente foi feito uma análise de vários modelos de ata de cerimônias ICPEdu do modelo atual. Com a análise do modelo atual foi possível desenvolver um aplicativo com melhorias em alguns aspectos e defeitos em outros, estas apontadas na avaliação dos entrevistados via questionário.

A avaliação dos usuários via questionário obteve um resultado positivo, porém por ser uma amostra muito pequena e não generalizada de avaliadores e também por ter sido feito somente um teste não se pode afirmar o desempenho do aplicativo sobre os critérios avaliados. Porém mesmo sem poder fazer afirmações sobre o aplicativo com as respostas obtidas, o fato do retorno das respostas serem positivas faz pensar que o aplicativo realmente será utilizado no ambiente seguro do CCD-UFSC e posteriormente em outros lugares que utilizam cerimônias.

Este trabalho entretanto nada mais é que a porta de entrada para outros trabalhos, ao utilizar uma aplicação de propósito específico para concluir uma tarefa ao invés de uma de propósito geral muitas coisas podem ser trabalhadas/implementadas para melhorar o aplicativo visando melhorias no processo de execução e operação de uma ICP, neste caso a ICPEdu.

Muitos trabalhos futuros podem ser feitos com a inserção do SACI como um novo componente no processo de execução de cerimônias ICPEdu, alguns deles estão descritos na próxima seção.

4.1. Trabalhos Futuros

O SACI tem como objetivo melhorar o ambiente no processo de execução de cerimônias ICP, algumas melhorias no processo de execução de cerimônias da ICPEdu já podem ser garantidas com as funcionalidades já desenvolvidas. Muitas outras áreas podem ser exploradas no desenvolvimento do SACI, entre elas:

1. Melhorar a usabilidade e a interface para o usuário;
2. Garantir segurança e armazenamento dos dados digitais;
3. Automatização dos dados gerados na execução de uma cerimônia;
4. Generalização dos aspectos de uma cerimônia;
5. Elaborar um design responsivo;
6. Melhorar o desempenho do aplicativo.

O item 1, a melhoria da interface para o usuário e da usabilidade do SACI pode garantir uma melhoria no processo de execução de cerimônias.

O item 2, os dados gerados por uma cerimônia devem ser armazenados de uma maneira segura garantindo integridade, não-repúdio e autenticidade. Com a utilização de assinatura digital estas propriedades podem ser garantidas, para isso deve-se estudar a viabilidade e os benefícios de desenvolver uma funcionalidade para o SACI assinar digitalmente os arquivos gerados em uma cerimônia.

Sobre o item 3, a utilização de entrada e saída para as cerimônias com a utilização de linguagem de marcação permite que seja desenvolvida diversas funcionalidades para analisar de maneira automatizada os dados da cerimônia, por exemplo, verificar se a cerimônia obteve sucesso através dos *logs* das aplicações e da saída do SACI, verificações de auditoria, análise de métricas de tempo, geração de métricas de tempo e sanitização do documento da ata.

Sobre o item 4, generalizar os aspectos de uma cerimônia, permite que o aplicativo seja utilizado para outros tipos de cerimônias como por exemplo as cerimônias do Sistema de Votação Helios e também pode tornar mais inclusivo a utilização do SACI para a utilização do mesmo em outras ICPs.

O item 5 permite com que o aplicativo seja utilizado em qualquer dispositivo móvel, porém como os dados que devem ser apresentados para executar uma cerimônia ocupam bastante espaço, não é qualquer dispositivo móvel que vai apresentar os dados necessários de uma maneira agradável.

Sobre o item 6, o aplicativo foi desenvolvido por um amador, em aprendizado, o SACI pode ter seu desempenho melhorado, com melhorias nos padrões de desenvolvimento, buscando melhorar a utilização de memória e melhorar o tratamento de erro. Além disso para o desenvolvimento de qualquer funcionalidade nova no SACI seria interessante fazer uma refatoração no código, recomeçando com algum padrão de desenvolvimento e utilizando um padrão de arquitetura de software, como o Modelo-Visão-Controle (MVC).

References

ELLISON, C. Ceremony design and analysis. Cryptology ePrint Archive, 2007.

LUZ, C. P. d. Centro de Certificação Digital – Construção, Administração e Manutenção. [S.I]: Editora Ciência Moderna Ltda, 2008. 338 p.

ANEXO F – Código Fonte do SACI

Segue algumas das classes do código do SACI em anexo, para detalhes do projeto completo acessar: https://github.com/jr-/cerimony-assistant-0.1.

anexos/code/AbortedCeremony.java

```
package tcc.cerimony_assistant_v01;

/**
 * Created by zr on 27/03/18.
 */

public class AbortedCeremony {
    private String reason = "";
    private int step_number = -1;

    public String getReason() {
        return reason;
    }

    public void setReason(String reason) {
        this.reason = reason;
    }

    public int getStep_number() {
        return step_number;
    }

    public void setStep_number(int step_number) {
        this.step_number = step_number;
    }

}
```

anexos/code/CCerimonies.java

```
package tcc.cerimony_assistant_v01;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * Created by zr on 03/06/17.
 */

public class CCerimonies {
    private static CCerimonies cc = null;
```

```
    private Map<String, Cerimony> lc;
    private String selectedCerimonyName;

    private CCerimonies() {}

    public void setCerimonies(List<String>
        cerimoniesFileName){
        lc = new HashMap<>();
        for(int i = 0; i < cerimoniesFileName.size();
            i++){
            Cerimony c = new Cerimony();
            lc.put(cerimoniesFileName.get(i), c);
        }
    }

    public void setSelectedCerimony(String
        cerimonyName) {
        selectedCerimonyName = cerimonyName;
    }

    public Cerimony getSelectedCerimony() {
        return lc.get(selectedCerimonyName);
    }

    public static CCerimonies getInstance(){
        if(cc == null)
            cc = new CCerimonies();

        return cc;
    };

    public Map<String, Cerimony> getCerimonies() {
        return lc;
    }
}
```

anexos/code/CeremonyDetailsPageAdapter.java

```
package tcc.cerimony_assistant_v01;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import java.util.ArrayList;
import java.util.List;

/**
```

```
 * Created by zr on 04/01/18.
 */

public class CeremonyDetailsPageAdapter extends
    FragmentPagerAdapter {

    private final List<Fragment> mFragmentList = new
        ArrayList<>();
    private final List<String> mFragmentTitleList =
        new ArrayList<>();

    public CeremonyDetailsPageAdapter(FragmentManager
        fm) {
        super(fm);
    }

    public void addFragment(Fragment fragment, String
        title) {
        mFragmentList.add(fragment);
        mFragmentTitleList.add(title);
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return mFragmentTitleList.get(position);
    }

    @Override
    public Fragment getItem(int position) {
        return mFragmentList.get(position);
    }

    @Override
    public int getCount() {
        return mFragmentList.size();
    }
}
```

anexos/code/Cerimony.java

```
package tcc.cerimony_assistant_v01;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by zr on 31/01/17.
 */
```

```
public class Cerimony {
    private String name;
    private String shortName;
    private String creationDate;
    private String initialDate;
    private String finalDate;
    private String initialTime;
    private String finalTime;
    private String local;
    private boolean confirmRequirements = false;
    private boolean isAborted = false;
    private List<Participant> participants = new
        ArrayList<Participant>();
    private List<Step> steps = new ArrayList<Step>();
    private List<String> requirements = new
        ArrayList<String>();
    private String folderName;
    private AbortedCeremony abortedCeremony;
    private int currentStepNumber = 0;
    private String fileName;

    public String getFileName() {
        return fileName;
    }

    public void setFileName(String fileName) {
        this.fileName = fileName;
    }

    public int getCurrentStepNumber() {
        return currentStepNumber;
    }

    public void setCurrentStepNumber(int
        currentStepNumber) {
        this.currentStepNumber = currentStepNumber;
    }

    public Cerimony() {}

    public AbortedCeremony getAbortedCeremony() {
        return abortedCeremony;
    }

    public void setAbortedCeremony(AbortedCeremony
        abortedCeremony) {
        this.abortedCeremony = abortedCeremony;
    }
```

```
public boolean isAborted() {return isAborted;}

public void setAborted(boolean isAborted)
    {this.isAborted = isAborted;}

public boolean isConfirmRequirements() {return
    confirmRequirements;}

public void setConfirmRequirements(boolean
    confirmRequirements)
    {this.confirmRequirements =
    confirmRequirements;}

public String getShortName() {
    return shortName;
}

public void setShortName(String shortName) {
    this.shortName = shortName;
}

public String getFolderName() {
    return folderName;
}

public void setFolderName(String folderName) {
    this.folderName = folderName;
}

public String getInitialTime() {
    return initialTime;
}

public void setInitialTime(String initialTime) {
    this.initialTime = initialTime;
}

public String getFinalTime() {
    return finalTime;
}

public void setFinalTime(String finalTime) {
    this.finalTime = finalTime;
}

public String getCName() {
    return name;
}
```

```
public void setCName(String name) {
    this.name = name;
}

public String getCreationDate() {
    return creationDate;
}

public void setCreationDate(String creationDate) {
    this.creationDate = creationDate;
}

public String getInitialDate() {
    return initialDate;
}

public void setInitialDate(String initialDate) {
    this.initialDate = initialDate;
}

public String getFinalDate() {
    return finalDate;
}

public void setFinalDate(String finalDate) {
    this.finalDate = finalDate;
}

public String getLocal() {
    return local;
}

public void setLocal(String local) {
    this.local = local;
}

public List<Participant> getParticipants() {
    return participants;
}

public void setParticipants(List<Participant>
    participants) {
    this.participants = participants;
}

public List<Step> getSteps() {
    return steps;
}
```

```
public void setSteps(List<Step> steps) {
    this.steps = steps;
}

public List<String> getRequirements() { return
    requirements; }

public void setRequirements(List<String>
    requirements) { this.requirements =
    requirements; }

public String toXML() {

    String cerimony = "<?xml version=\"1.0\"
        encoding=\"UTF-8\"?>\r\n"
             + "<cerimony>\r\n"
             + "\t<name>" + name + "</name>\r\n"
             + "\t<short_name>" + shortName +
                 "</short_name>\r\n"
             + "\t<local>" + local +
                 "</local>\r\n"
             + "\t<requirements>\r\n";

    for(int i = 0; i < requirements.size(); i++) {
        cerimony += "\t\t<item>" +
            requirements.get(i) + "</item>\r\n";
    }


    cerimony += "\t</requirements>\r\n"
             + "\t<steps>\r\n";
    for(int i = 0; i < steps.size(); i++) {
        cerimony += "\t\t<step " + "name=\"" +
            steps.get(i).getSName() + "\">\r\n"
                 + "\t\t\t<description>" +
                     steps.get(i).getDescription()
                     + "</description>\r\n"
                 + "\t\t\t<input>" +
                     steps.get(i).getInput() +
                     "</input>\r\n"
                 + "\t\t\t<output>" +
                     steps.get(i).getOutput() +
                     "</output>\r\n"
                 + "\t\t\t<observation>" +
                     steps.get(i).getObservation()
                     + "</observation>\r\n"
                 + "\t\t</step>\r\n";
    }
```

```
        cerimony += "\t</steps>\r\n"
                + "</cerimony>\r\n";
        return cerimony;
    }

    public String toTXT() {

        String ceremony = name + "\r\n"
               + "Local: " + local + "\r\n"
               + "Resultado: ";
        if(isAborted) {
            ceremony += "A cerimônia foi abortada pelo
                motivo " + abortedCeremony.getReason()
                + " no passo " +
                abortedCeremony.getStep_number()
                     + " às " + finalTime + " do dia "
                         + finalDate + "." + "\r\n";
        } else {
            ceremony += "A cerimônia terminou com
                sucesso" + "\r\n";
        }

        ceremony += "Data de início: " + initialDate
            + "\r\n"
                  + "Horário de início: " +
                      initialTime + "\r\n"
                  + "Data de término: " + finalDate +
                      "\r\n"
                  + "Horário de término: " + finalTime
                      + "\r\n"
                  + "Requisitos confirmados:";

        if(requirements.size() > 0) {
            if(requirements.size() > 2) {
                for(int i = 0; i <
                    requirements.size()-2; i++) {
                    ceremony += " " +
                        requirements.get(i) + ",";
                }
            }
            if(requirements.size() != 1) {
                ceremony += " " +
                    requirements.get(requirements.size()-2)
                    + " e";
            }
            ceremony += " " +
                requirements.get(requirements.size()-1)
                + ".";
        }
```

```
            ceremony += "\r\n";
            if(participants.size() > 0) {
                ceremony += "Participantes
                    confirmados:\r\n";
                for(int i = 0; i < participants.size();
                    i++) {
                    Participant cp = participants.get(i);
                    ceremony += cp.getPName() + " - " +
                        cp.getCargo() + " - " +
                        cp.getUnidade() + " - " +
                        cp.getEmail() + "\r\n";
                }
            }
            ceremony +=
                "-----------------------------------------------------
            if(steps.size() > 0) {
                int loop_size = steps.size();
                if(isAborted) {
                    loop_size =
                        abortedCeremony.getStep_number() +
                        1;
                }
                for(int i = 0; i < loop_size; i++) {
                    Step cs = steps.get(i);
                    ceremony += "Passo " + i + ": " +
                        cs.getSName() + " - " +
                        cs.getTime() + "\r\n"
                            + "Descrição: " +
                                cs.getDescription() + "\r\n"
                            + "Entrada: " + cs.getInput() +
                                "\r\n"
                            + "Saída: " + cs.getOutput() +
                                "\r\n"
                            + "Observações: " +
                                cs.getObservation() + "\r\n"
                            + "Anotações: " + cs.getNotes()
                                + "\r\n"
                            + "Evidências: " +
                                cs.getEvidence() + "\r\n";
                    ceremony +=
                        "-----------------------------------------------
                }

            }
            return ceremony;
        }
    }
```

anexos/code/CerimonyDetails.java

```
package tcc.cerimony_assistant_v01;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

public class CerimonyDetails extends
    AppCompatActivity {

    private static final String TAG =
        "CerimonyDetails";

    private CeremonyDetailsPageAdapter mCMPageAdapter;

    public ViewPager mViewPager;
    public Toolbar mToolbar;

    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cerimony_details);

        mToolbar = (Toolbar)
            findViewById(R.id.toolbar);
        setSupportActionBar(mToolbar);

        mCMPageAdapter = new
            CeremonyDetailsPageAdapter(getSupportFragmentManager());

        mViewPager = (ViewPager)
            findViewById(R.id.vp);
        setupViewPager(mViewPager);

        TabLayout tabLayout = (TabLayout)
            findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(mViewPager);
    }

    private void setupViewPager(ViewPager viewPager) {
        CeremonyDetailsPageAdapter adapter = new
            CeremonyDetailsPageAdapter(getSupportFragmentManager());
        adapter.addFragment(new
            CerimonyDetailsFragment(), "Detalhes");
        adapter.addFragment(new
            RequirementsFragment(), "Requisitos");
```

```
        adapter.addFragment(new
            ParticipantsFragment(), "Participantes");

        viewPager.setAdapter(adapter);
    }

}
```

anexos/code/CerimonyDetailsFragment.java

```
package tcc.cerimony_assistant_v01;

import android.content.Context;
import android.content.Intent;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.ActionBar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.text.Html;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import java.util.List;

/**
 * A placeholder fragment containing a simple view.
 */
public class CerimonyDetailsFragment extends
    Fragment {
    private static final String TAG =
        "CerimonyDetailsFragment";
    private boolean first_time = true;
    private View mRootView;
    public CerimonyDetailsFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
                             Bundle savedInstanceState)
```

```
                                {
        mRootView =
            inflater.inflate(R.layout.fragment_cerimony_details,
            container, false);

        // The detail Activity called via intent.
            Inspect the intent for forecast data.
        Intent intent = getActivity().getIntent();
        String cerimonyPath = "";
        String cerimonyFileName = "";
        Cerimony cerimony;
        if (intent != null &&
            intent.hasExtra(Intent.EXTRA_TEXT)) {
            cerimonyFileName =
                intent.getStringExtra(Intent.EXTRA_TEXT);
            cerimonyPath = "new/"+cerimonyFileName;
            CCerimonies.getInstance().setSelectedCerimony(cerimonyFileN
            cerimony =
                CCerimonies.getInstance().getCerimonies().get(cerimonyF
            cerimony =
                CerimonyXmlPullParser.getCerimonyFromFile(getActivity()
                "new/"+cerimonyFileName, cerimony);
            cerimony.setFileName(cerimonyFileName);

            //dinamically modify GUI
            String format_title = "Abertura da Ata" +
                " - " + cerimony.getShortName();
            ((CerimonyDetails)
                getActivity()).mToolbar.setTitle(format_title);


            TextView tv_name = ((TextView)
                mRootView.findViewById(R.id.c_details_name));
            tv_name.setText(Html.fromHtml("<b>Nome da
                cerimônia:</b> " +
                cerimony.getCName()));

            TextView tv_participants = ((TextView)
                mRootView.findViewById(R.id.c_details_participants));
            String participants_text;
            List<Participant> participants =
                cerimony.getParticipants();
            if(participants.size() == 0) {
                participants_text = "<i>Nenhum
                    participante confirmado!</i>";
            } else {
                participants_text = "<b>Participantes
                    confirmados:</b><br />";
                for(int i = 0; i <
```

```
                participants.size(); i++) {
                participants_text += "\u2022 " +
                    participants.get(i).getPName()
                    + "<i> como </i>" +
                    participants.get(i).getCargo()
                    + "<br />";
            }
        }

        tv_participants.setText(Html.fromHtml(participants_tex

        TextView tv_requirements = ((TextView)
            mRootView.findViewById(R.id.c_details_requirements
        String requirements_text = "";
        boolean isConfirmRequirements =
            CCerimonies.getInstance().getSelectedCerimony().is
        if(!isConfirmRequirements) {
            requirements_text = "<i>Os requisitos
                não foram confirmados!</i>";
        } else {
            requirements_text = "<i>Os requisitos
                foram confirmados!</i>";
        }
        tv_requirements.setText(Html.fromHtml(requirements_tex

    }



    Button btn = (Button)
        mRootView.findViewById(R.id.start_cerimony_button);
    final String finalCerimonyNewPath =
        cerimonyPath;
    final String finalCerimonyLoadedPath =
        "load/"+cerimonyFileName;
    final String finalCerimonyFileName =
        cerimonyFileName;
    btn.setOnClickListener(new
        View.OnClickListener() {
        @Override
        public void onClick(View v) {
            List<Participant> participantList =
                CCerimonies.getInstance().getSelectedCerimony()
            boolean isConfirmRequirements =
                CCerimonies.getInstance().getSelectedCerimony()
            if(participantList.size() == 0 ||
                !isConfirmRequirements) {
                //display feedback message
                Context context = getContext();
```

```
                    CharSequence text = "É necessário
                        confirmar todos os requisitos e
                        confirmar pelo menos 1
                        participante para iniciar a
                        cerimônia";
                    int duration = Toast.LENGTH_SHORT;

                    Toast toast =
                        Toast.makeText(context, text,
                        duration);
                    toast.show();
                } else {
                    Intent intent = new
                        Intent(getActivity(),
                        ExecuteSteps.class);
                    intent.putExtra("NEW_PATH",
                        finalCerimonyNewPath);
                    intent.putExtra("LOAD_PATH",
                        finalCerimonyLoadedPath);
                    intent.putExtra("NAME",
                        finalCerimonyFileName);
                    startActivity(intent);
                }
            }
        });

        return mRootView;
    }

    public void onStart(){
        super.onStart();
        if(first_time == true) {
            ViewPager viewPager = (ViewPager)
                getActivity().findViewById(R.id.vp);
            viewPager.setCurrentItem(2);
            first_time = false;
        }
    }

    public void update(){
        TextView tv_requirements = ((TextView)
            mRootView.findViewById(R.id.c_details_requirements));
        String requirements_text = "";
        boolean isConfirmRequirements =
            CCerimonies.getInstance().getSelectedCerimony().isConfirmRe
        if(!isConfirmRequirements) {
            requirements_text = "<i>Os requisitos não
                foram confirmados!</i>";
        } else {
```

```
            requirements_text = "<i>Os requisitos
                foram confirmados!</i>";
        }
        tv_requirements.setText(Html.fromHtml(requirements_text))
    }


}
```

anexos/code/CerimonyXmlPullParser.java

```
package tcc.cerimony_assistant_v01;

import android.content.Context;
import android.content.res.AssetFileDescriptor;
import android.os.Environment;
import android.util.Log;

import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;

import tcc.cerimony_assistant_v01.Cerimony;

/**
 * Created by zr on 31/01/17.
 */

public class CerimonyXmlPullParser {

    public static Cerimony
        getCerimonyFromFile(Context ctx, String
        filePath, Cerimony cerimony) {
        String creator;
        String curText = "";
        try {
            XmlPullParserFactory factory =
                XmlPullParserFactory.newInstance();
            XmlPullParser xpp =
                factory.newPullParser();
```

```
String root_sd =
    Environment.getExternalStorageDirectory().toString();
BufferedReader reader = new
    BufferedReader(new
    FileReader(root_sd+"/ceremony-assistant/"+filePath));
xpp.setInput(reader);

int eventType = xpp.getEventType();
String tagname = xpp.getName();

while (
    !"requirements".equalsIgnoreCase(tagname)
    && eventType !=
    XmlPullParser.END_DOCUMENT) {

    //TODO refactoring mais elegante fazer
        loop até END_TAG
    switch (eventType) {
        case XmlPullParser.START_TAG:
            break;
        case XmlPullParser.TEXT:
            curText = xpp.getText();
            break;
        case XmlPullParser.END_TAG:
            if
                (tagname.equalsIgnoreCase("name"))
                {
                cerimony.setCName(curText);
            } else if
                (tagname.equalsIgnoreCase("local"))
                {
                cerimony.setLocal(curText);
            } else if
                (tagname.equalsIgnoreCase("short_name"))
                {
                cerimony.setShortName(curText);
            }
            break;
        default:
            break;
    }

    eventType = xpp.next();
    tagname = xpp.getName();
}

List<String> requirements = new
    ArrayList<String>();
while ( !"steps".equalsIgnoreCase(tagname)
```

```
        && eventType !=
        XmlPullParser.END_DOCUMENT) {

        switch (eventType) {
            case XmlPullParser.START_TAG:
                break;
            case XmlPullParser.TEXT:
                curText = xpp.getText();
                break;
            case XmlPullParser.END_TAG:
                if
                    (tagname.equalsIgnoreCase("item"))
                    requirements.add(curText);
                    break;
            default:
                break;
        }
        eventType = xpp.next();
        tagname = xpp.getName();
    }
    cerimony.setRequirements(requirements);
    eventType = xpp.next();
    tagname = xpp.getName();

    List<Step> steps = new ArrayList<Step>();
    while(!("steps".equalsIgnoreCase(tagname)
        && eventType ==
        XmlPullParser.END_TAG)) {

        if ("step".equalsIgnoreCase(tagname)
            && eventType ==
            XmlPullParser.START_TAG) {
            Step curStep = new Step();
            String stepName =
                xpp.getAttributeValue(null,
                "name");
            curStep.setSName(stepName);

            while
                (!("step".equalsIgnoreCase(tagname)
                && eventType ==
                XmlPullParser.END_TAG)) {

                switch (eventType) {
                    case
                        XmlPullParser.START_TAG:
                        break;
                    case XmlPullParser.TEXT:
                        curText = xpp.getText();
```

```
                            break;
                        case XmlPullParser.END_TAG:
                            if
                                (tagname.equalsIgnoreCase("input"))
                                {
                                curStep.setInput(curText);
                            } else if
                                (tagname.equalsIgnoreCase("output"))
                                {
                                curStep.setOutput(curText);
                            } else if
                                (tagname.equalsIgnoreCase("description
                                {
                                curStep.setDescription(curText);
                            } else if
                                (tagname.equalsIgnoreCase("observation
                                {
                                curStep.setObservation(curText);
                            }
                            break;
                        default:
                            break;

                    }
                    eventType = xpp.next();
                    tagname = xpp.getName();
                }

                steps.add(curStep);
            }
            eventType = xpp.next();
            tagname = xpp.getName();
        }
        cerimony.setSteps(steps);

    } catch (Exception e) {
        e.printStackTrace();
    }
    return cerimony;
  }
}
```

anexos/code/EndCeremonyActivity.java

```
package tcc.cerimony_assistant_v01;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
```

```
import android.support.v7.widget.Toolbar;

public class EndCeremonyActivity extends
    AppCompatActivity {

    private static final String TAG =
        "EndCeremonyActivity";


    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_end_ceremony);
        Toolbar toolbar = (Toolbar)
            findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
    }

}
```

anexos/code/EndCeremonyFragment.java

```
package tcc.cerimony_assistant_v01;

import android.content.Intent;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;


/**
 * A placeholder fragment containing a simple view.
 */
public class EndCeremonyFragment extends Fragment {
    private static final String TAG =
        "EndCeremonyFragment";

    public EndCeremonyFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
```

```
                                Bundle savedInstanceState)
                                    {
        View rootView =
            inflater.inflate(R.layout.fragment_end_ceremony,
            container, false);
        Cerimony ceremony =
            CCerimonies.getInstance().getSelectedCerimony();

        //dinamically modify GUI
        getActivity().setTitle("Fechamento da Ata - "
            + ceremony.getShortName());
        TextView status_tv = (TextView)
            rootView.findViewById(R.id.status);

        if(ceremony.isAborted()) {
            AbortedCeremony ac =
                ceremony.getAbortedCeremony();
            String reason = ac.getReason();
            int step_number = ac.getStep_number();
            status_tv.setText("A cerimônia foi
                abortada pelo motivo: \"" + reason
                +"\" no passo " + step_number);
        } else {
            status_tv.setText("A Cerimônia terminou
                com sucesso!");
        }


        Button btn = (Button)
            rootView.findViewById(R.id.return_main);
        btn.setOnClickListener(new
            View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new
                    Intent(getActivity(),
                    ListNewCerimonies.class);
                startActivity(intent);
            }
        });
        return rootView;
    }
}
```

anexos/code/ExecuteSteps.java

```
package tcc.cerimony_assistant_v01;
```

```
import android.content.DialogInterface;
import android.content.Intent;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import
    android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.TextView;

import java.io.File;

public class ExecuteSteps extends AppCompatActivity {

    public TextView title_toolbar;
    public TextView status_toolbar;
    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_execute_steps);
        Toolbar toolbar = (Toolbar)
            findViewById(R.id.toolbar);
        title_toolbar = (TextView)
            toolbar.findViewById(R.id.title_toolbar);
        status_toolbar = (TextView)
            toolbar.findViewById(R.id.status_toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(false);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the
            action bar if it is present.
```

```
        getMenuInflater().inflate(R.menu.menu_execute_steps,
            menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem
        item) {
        // Handle action bar item clicks here. The
            action bar will
        // automatically handle clicks on the Home/Up
            button, so long
        // as you specify a parent activity in
            AndroidManifest.xml.
        int id = item.getItemId();
        final int TAKE_PICTURE = 1;
        Uri imageUri;


        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            Intent intent = new
                Intent(MediaStore.ACTION_IMAGE_CAPTURE);
            Cerimony selectedCeremony =
                CCerimonies.getInstance().getSelectedCerimony();
            String folderName =
                selectedCeremony.getFolderName();
            int currentStepNumber =
                selectedCeremony.getCurrentStepNumber();
            String evidenceName = "evidencia-passo" +
                currentStepNumber + ".jpg";
            File photo = new
                File(Environment.getExternalStorageDirectory(),
                "ceremony-assistant/final/" +
                folderName + "/" + evidenceName);
            intent.putExtra(MediaStore.EXTRA_OUTPUT,
                    Uri.fromFile(photo));
            imageUri = Uri.fromFile(photo);
            startActivityForResult(intent,
                TAKE_PICTURE);

        }

        if (id == R.id.notes) {
            AlertDialog.Builder builder = new
                AlertDialog.Builder(this);
            builder.setTitle("Escreva suas
                anotações:");
            ViewGroup view = (ViewGroup)
```

```
                findViewById(android.R.id.content);
            View viewInflated =
                LayoutInflater.from(this).inflate(R.layout.abort_d
                view, false);
            final EditText input = (EditText)
                viewInflated.findViewById(R.id.input);
            builder.setView(viewInflated);

            builder.setPositiveButton(android.R.string.ok,
                new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface
                    dialog, int which) {
                    Cerimony selectedCeremony =
                        CCerimonies.getInstance().getSelectedCerimo
                    int currentStepNumber =
                        selectedCeremony.getCurrentStepNumber();
                    String note =
                        input.getText().toString();
                    selectedCeremony.getSteps().get(currentStepNumb

                }
            });
            builder.setNegativeButton(android.R.string.cancel,
                new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface
                    dialog, int which) {
                    dialog.cancel();
                }
            });

            builder.show();
        }

        return super.onOptionsItemSelected(item);
    }


    public void onActivityResult(int requestCode, int
        resultCode, Intent data) {
        if(requestCode == 1 && resultCode ==
            RESULT_OK) {
            Cerimony selectedCeremony =
                CCerimonies.getInstance().getSelectedCerimony();
            int currentStepNumber =
                selectedCeremony.getCurrentStepNumber();
            String evidenceName = "evidencia-passo" +
                currentStepNumber + ".jpg";
```

```
            selectedCeremony.getSteps().get(currentStepNumber).setEvide
        }
    }

}
```

anexos/code/ExecuteStepsFragment.java

```
package tcc.cerimony_assistant_v01;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.Color;
import android.net.Uri;
import android.os.Environment;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import org.w3c.dom.Text;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

/**
```

```
 * A placeholder fragment containing a simple view.
 */
public class ExecuteStepsFragment extends Fragment {


//   private ArrayAdapter<String> mListStepsAdapter;
    private StepsListAdapter mStepsListAdapter;
    private ListView mListStepsView;

    private boolean first_time = true;
    public ExecuteStepsFragment() {
    }
    int stepNumber = 0;
    Step curStep;
    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
                             Bundle savedInstanceState)
                                {
        final View rootView =
            inflater.inflate(R.layout.fragment_execute_steps,
            container, false);
//
//       // The detail Activity called via intent.
    Inspect the intent for forecast data.
        Intent intent = getActivity().getIntent();
        if (intent != null &&
            intent.hasExtra("LOAD_PATH")) {
            final String cerimonyName =
                intent.getStringExtra("NAME");

            //getting timestamp to start_ceremony_date
                and hour
            long msTime = System.currentTimeMillis();
            Date curDateTime = new Date(msTime);


            final Cerimony selectedCerimony =
                CCerimonies.getInstance().getCerimonies().get(ceri

            //set date to start_ceremony_date and hour
            Calendar cal = Calendar.getInstance();
            cal.setTime(curDateTime);
            String curTime =
                cal.get(Calendar.HOUR_OF_DAY) + ":" +
                cal.get(Calendar.MINUTE);
            String curDate =
                cal.get(Calendar.DAY_OF_MONTH) + "/" +
                (cal.get(Calendar.MONTH) + 1) + "/" +
```

```
    cal.get(Calendar.YEAR);
String rgDate = cal.get(Calendar.YEAR) +
    "." + (cal.get(Calendar.MONTH) + 1) +
    "." + cal.get(Calendar.DAY_OF_MONTH);

selectedCerimony.setInitialDate(curDate);
selectedCerimony.setInitialTime(curTime);

//create folder for started ceremony
final String folderName = rgDate + " - " +
    curTime + " - " +
    selectedCerimony.getShortName();
selectedCerimony.setFolderName(folderName);
File folder = new
    File(Environment.getExternalStorageDirectory(),
    "ceremony-assistant/final/" +
    folderName);
folder.mkdirs();
try {
    this.copy(new
        File(Environment.getExternalStorageDirectory(),"cere
        + cerimonyName), new
        File(Environment.getExternalStorageDirectory(),
        "ceremony-assistant/final/" +
        folderName + "/original-" +
        cerimonyName));
} catch (IOException e) {
    e.printStackTrace();
}

//get steps and curStep = steps(0)
final List<Step> steps =
    selectedCerimony.getSteps();
curStep = steps.get(stepNumber);

//dinamically modify GUI to step(0)

final EditText description_et =
    ((EditText)
    rootView.findViewById(R.id.description_text));
description_et.setText(curStep.getDescription());

final EditText input_et = ((EditText)
    rootView.findViewById(R.id.text_input));
input_et.setText(curStep.getInput());

final EditText output_et = ((EditText)
    rootView.findViewById(R.id.text_output));
output_et.setText(curStep.getOutput());
```

```
                final EditText observation_et =
                    ((EditText)
                    rootView.findViewById(R.id.text_observation));
                String observation_text =
                    curStep.getObservation();

                //TODO para todos tratar observation_text
                    colocar bullet depois de cada nova
                    linha e no começo do arquivo
                observation_et.setText(observation_text);


                final List<StepVisual> stepsVisualList =
                    new ArrayList<>();
                for ( int i=0; i < steps.size(); i++) {
                    String stepName =
                        steps.get(i).getSName();
                    StepVisual cur_step_visual = new
                        StepVisual(stepName, false);
                    stepsVisualList.add(cur_step_visual);
                }

//            mListStepsAdapter =
//                    new ArrayAdapter<String>(
//                            getActivity(),
//                            R.layout.list_item_steps,
//
    R.id.list_item_steps_textview,
//                            stepsList
//                    );
                mStepsListAdapter = new
                    StepsListAdapter(stepsVisualList,
                    getActivity());

                mListStepsView = (ListView)
                    rootView.findViewById(R.id.listview_steps);
                mListStepsView.setAdapter(mStepsListAdapter);


                final Button nextBtn = (Button)
                    rootView.findViewById(R.id.next_step_button);
                Log.v("stepsize", "" + steps.size());
                nextBtn.setOnClickListener(new
                    View.OnClickListener() {
                    //           @Override
                    public void onClick(View v) {
                        //logic to save in the xml
```

```
long msTime =
    System.currentTimeMillis();
Date curDateTime = new
    Date(msTime);
Calendar cal =
    Calendar.getInstance();
cal.setTime(curDateTime);
String curTime =
    cal.get(Calendar.HOUR_OF_DAY) +
    ":" + cal.get(Calendar.MINUTE);
String description =
    description_et.getText().toString();
String input =
    input_et.getText().toString();
String output =
    output_et.getText().toString();
String observation =
    observation_et.getText().toString();

//modify the step content
curStep.setTime(curTime);
curStep.setDescription(description);
curStep.setInput(input);
curStep.setOutput(output);
curStep.setObservation(observation);

//dinamically modify GUI to
    step(1) at step(size-1)
stepsVisualList.get(stepNumber).setExecuted(true);
mStepsListAdapter.notifyDataSetChanged();
stepNumber++;
selectedCerimony.setCurrentStepNumber(stepNumber);
if (stepNumber < steps.size()) {
    curStep = steps.get(stepNumber);

    String step_name =
        curStep.getSName();
    ((ExecuteSteps)
        getActivity()).title_toolbar.setText(selected
        + " - " + step_name);
    int steps_size = steps.size();
    int display_step_number =
        stepNumber + 1;
    ((ExecuteSteps)
        getActivity()).status_toolbar.setText("Passo
        " + display_step_number + "
        de " + steps_size);

    description_et.setText(curStep.getDescription())
```

```
                input_et.setText(curStep.getInput());
                output_et.setText(curStep.getOutput());
                String observation_text =
                    curStep.getObservation();
                observation_et.setText(observation_text);

                if (stepNumber == steps.size()
                    - 1) {
                    nextBtn.setBackgroundColor(Color.GREEN)
                    nextBtn.setText("Finalizar");
                }
            }

            if (stepNumber == steps.size()) {
                //end of ceremony, save,
                    feedbackmessage in the
                    initial activity?
//                  File file;
                File file2;

                cal = Calendar.getInstance();
                cal.setTime(curDateTime);
                curTime =
                    cal.get(Calendar.HOUR_OF_DAY)
                    + ":" +
                    cal.get(Calendar.MINUTE);
                String curDate =
                    cal.get(Calendar.DAY_OF_MONTH)
                    + "/" +
                    (cal.get(Calendar.MONTH) +
                    1) + "/" +
                    cal.get(Calendar.YEAR);

                selectedCerimony.setFinalDate(curDate);
                selectedCerimony.setFinalTime(curTime);

                if
                    (isExternalStorageWritable())
                    {
                    String root_sd =
                        Environment.getExternalStorageDirect
                    File dir = new File(root_sd
                        +
                        "/ceremony-assistant/final/"
                        + folderName);
                    dir.mkdirs();
                    try {
                        File file = new
                            File(root_sd +
```

```
                                    "/ceremony-assistant/new/",
                                    selectedCerimony.getFileName());
                                file.createNewFile();
                                FileOutputStream f = new
                                    FileOutputStream(file,
                                    false);
                                String xmlData =
                                    selectedCerimony.toXML();
                                f.write(xmlData.getBytes());
                                f.close();

                                file2 = new File(dir,
                                    selectedCerimony.getShortName().repla
                                    "") + "-" +
                                    selectedCerimony.getFinalDate().repla
                                    "-") + ".txt");
                                file2.createNewFile();
                                FileOutputStream f2 =
                                    new
                                    FileOutputStream(file2);
                                String txtData =
                                    selectedCerimony.toTXT();
                                f2.write(txtData.getBytes());
                                f2.close();

                                //send the information
                                    about the new file
                                    and folders to
                                    scanner
                                Intent intent = new
                                    Intent(Intent.ACTION_MEDIA_SCANNER_SC
                                intent.setData(Uri.fromFile(file2));
                                getActivity().sendBroadcast(intent);

                                Intent intent2 = new
                                    Intent(getActivity(),
                                    EndCeremonyActivity.class);
                                startActivity(intent2);
                            } catch (IOException e) {
                                e.printStackTrace();
                                //Log.v("error",
                                    e.printStackTrace());
                            }
                        }

                    }
                }
            });
```

```
final Button abort_btn = (Button)
    rootView.findViewById(R.id.abort_button);
abort_btn.setOnClickListener(new
    View.OnClickListener() {
    //           @Override
    public void onClick(View v) {
        AlertDialog.Builder builder = new
            AlertDialog.Builder(getContext());
        builder.setTitle("Abortar
            Cerimônia");
        builder.setMessage("Deseja abortar
            a cerimônia por qual motivo?");

        View viewInflated =
            LayoutInflater.from(getContext()).inflate(R
            (ViewGroup) getView(), false);
        final EditText input = (EditText)
            viewInflated.findViewById(R.id.input);
        builder.setView(viewInflated);

        builder.setPositiveButton(android.R.string.ok,
            new
            DialogInterface.OnClickListener()
            {
            @Override
            public void
                onClick(DialogInterface
                dialog, int which) {
                Cerimony ceremony =
                    CCerimonies.getInstance().getSelecte

                AbortedCeremony ac = new
                    AbortedCeremony();
                ac.setReason(input.getText().toString())
                ac.setStep_number(stepNumber);

                ceremony.setAborted(true);
                ceremony.setAbortedCeremony(ac);

                File file2;

                long msTime2 =
                    System.currentTimeMillis();
                Date curDateTime2 = new
                    Date(msTime2);
                Calendar cal2 =
                    Calendar.getInstance();
                cal2.setTime(curDateTime2);
```

```
                                String curTime2 =
                                    cal2.get(Calendar.HOUR_OF_DAY)
                                    + ":" +
                                    cal2.get(Calendar.MINUTE);
                                String curDate2 =
                                    cal2.get(Calendar.DAY_OF_MONTH)
                                    + "/" +
                                    (cal2.get(Calendar.MONTH)
                                    + 1) + "/" +
                                    cal2.get(Calendar.YEAR);

                                selectedCerimony.setFinalDate(curDate2);
                                selectedCerimony.setFinalTime(curTime2);

                                if
                                    (isExternalStorageWritable())
                                    {
                                    String root_sd =
                                        Environment.getExternalStorageDirecto
                                    File dir = new
                                        File(root_sd +
                                        "/ceremony-assistant/final/"
                                        + folderName);
                                    dir.mkdirs();
                                    try {
//                                      file = new
    File(dir,
    selectedCerimony.getShortName().replaceAll("\\s",
    "") + "-" +
    selectedCerimony.getFinalDate().replaceAll("/",
    "-") + ".xml");
//
    file.createNewFile();
//                                      FileOutputStream f
    = new FileOutputStream(file);
//                                      String xmlData =
    selectedCerimony.toXML();
//
    f.write(xmlData.getBytes());
//                                      f.close();

                                    file2 = new
                                        File(dir,
                                        selectedCerimony.getShortName().re
                                        "") + "-" +
                                        selectedCerimony.getFinalDate().re
                                        "-") + ".txt");
                                    file2.createNewFile();
                                    FileOutputStream f2
```

```
                                    = new
                                    FileOutputStream(file2);
                                String txtData =
                                    selectedCerimony.toTXT();
                                f2.write(txtData.getBytes());
                                f2.close();

                                //send the
                                    information
                                    about the new
                                    file and folders
                                    to scanner
                                Intent intent = new
                                    Intent(Intent.ACTION_MEDIA_S
                                intent.setData(Uri.fromFile(file
                                getActivity().sendBroadcast(inte

                                Intent intent2 = new
                                    Intent(getActivity(),
                                    EndCeremonyActivity.class);
                                startActivity(intent2);
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                        }

                        Intent intent = new
                            Intent(getActivity(),
                            EndCeremonyActivity.class);
                        startActivity(intent);

                    }
                });
                builder.setNegativeButton(android.R.string.canc
                    new
                    DialogInterface.OnClickListener()
                    {
                    @Override
                    public void
                        onClick(DialogInterface
                        dialog, int which) {
                        dialog.cancel();
                    }
                });

                builder.show();
            }
        });
```

```
final Button prev_btn = (Button)
    rootView.findViewById(R.id.prev_step_button);
prev_btn.setOnClickListener(new
    View.OnClickListener() {
    //          @Override
    public void onClick(View v) {
        if(stepNumber > 0) {
            //modify the step content
            String description =
                description_et.getText().toString();
            String input =
                input_et.getText().toString();
            String output =
                output_et.getText().toString();
            String observation =
                observation_et.getText().toString();
            curStep.setDescription(description);
            curStep.setInput(input);
            curStep.setOutput(output);
            curStep.setObservation(observation);

            //dinamically modify GUI to
                step(1) at step(size-1)
            stepNumber--;
            stepsVisualList.get(stepNumber).setExecuted(false
            mStepsListAdapter.notifyDataSetChanged();
            mListStepsView.getChildAt(stepNumber).setBackgrou
            selectedCerimony.setCurrentStepNumber(stepNumber)
            if (stepNumber < steps.size()) {
                curStep =
                    steps.get(stepNumber);

                String step_name =
                    curStep.getSName();
                ((ExecuteSteps)
                    getActivity()).title_toolbar.setText(sele
                    + " - " + step_name);
                int steps_size =
                    steps.size();
                int display_step_number =
                    stepNumber + 1;
                ((ExecuteSteps)
                    getActivity()).status_toolbar.setText("Pa
                    " + display_step_number
                    + " de " + steps_size);

                //bullet code "\u2022 "
                description_et.setText(curStep.getDescription
                input_et.setText(curStep.getInput());
```

```
                        output_et.setText(curStep.getOutput());
                        String observation_text =
                            curStep.getObservation();
                        observation_et.setText(observation_text

                        if (stepNumber ==
                            steps.size() - 2) {
                            nextBtn.setBackgroundColor(Color.LTG
                            nextBtn.setText("Avançar");
                        }
                    }
                }

            }
        });
    }
    return rootView;
}

public void onStart() {
    super.onStart();
    if(first_time) {
        Cerimony ceremony =
            CCerimonies.getInstance().getSelectedCerimony();
        List<Step> steps = ceremony.getSteps();
        String step_name = steps.get(0).getSName();
        int steps_size = steps.size();
        ((ExecuteSteps)
            getActivity()).title_toolbar.setText(ceremony.getS
            + " - " + step_name);

        ((ExecuteSteps)
            getActivity()).status_toolbar.setText("Passo
            " + 1 + " de " + steps_size);

        first_time = false;
    }
}

/* Checks if external storage is available for
    read and write */
public boolean isExternalStorageWritable() {
    String state =
        Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}
```

```
    public static void copy(File src, File dst)
        throws IOException {
        InputStream in = new FileInputStream(src);
        try {
            OutputStream out = new
                FileOutputStream(dst);
            try {
                // Transfer bytes from in to out
                byte[] buf = new byte[1024];
                int len;
                while ((len = in.read(buf)) > 0) {
                    out.write(buf, 0, len);
                }
            } finally {
                out.close();
            }
        } finally {
            in.close();
        }
    }
}
```

anexos/code/ListNewCerimonies.java

```
package tcc.cerimony_assistant_v01;

import android.content.Intent;
import android.os.Bundle;
import
    android.support.design.widget.FloatingActionButton;
import android.support.design.widget.Snackbar;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;

public class ListNewCerimonies extends
    AppCompatActivity {

    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_new_cerimonies);
        Toolbar toolbar = (Toolbar)
            findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);
        this.setTitle(getResources().getString(R.string.app_name_

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the
            action bar if it is present.
        getMenuInflater().inflate(R.menu.menu_list_new_cerimonies
            menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem
        item) {
        // Handle action bar item clicks here. The
            action bar will
        // automatically handle clicks on the Home/Up
            button, so long
        // as you specify a parent activity in
            AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            Intent intent = new Intent(this,
                ManageParticipantsActivity.class);
            startActivity(intent);
        }

        return super.onOptionsItemSelected(item);
    }
}
```

anexos/code/ListNewCerimoniesFragment.java

```
package tcc.cerimony_assistant_v01;

import android.content.Intent;
import android.os.Environment;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
```

```
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * A placeholder fragment containing a simple view.
 */
public class ListNewCerimoniesFragment extends
    Fragment {

    private ArrayAdapter<String>
        mNewCerimoniesAdapter;

    public ListNewCerimoniesFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
                             Bundle savedInstanceState)
                                 {

//        String[] fileNames = new String[0];
//        try {
//            fileNames =
    getActivity().getAssets().list("new");
//        } catch (IOException e) {
//            e.printStackTrace();
//        }
//
//

        final List<String> cerimonies_file_names =
            new ArrayList<>();
        List<String> cerimonies_nice_names = new
            ArrayList<>();
        String root_sd =
            Environment.getExternalStorageDirectory().toString();
        File file = new File(root_sd +
            "/ceremony-assistant/new/");
        File list[] = file.listFiles();
        String ceremony_name;
        for( int i=0; i < list.length; i++) {
            ceremony_name = list[i].getName();
            cerimonies_file_names.add(ceremony_name);
```

```
        //manipulate file names to nice names
        //remove the extension file name
        ceremony_name = ceremony_name.substring(0,
            ceremony_name.lastIndexOf("."));

        ceremony_name =
            ceremony_name.replaceAll("-", " ");

        cerimonies_nice_names.add(ceremony_name);
    }
    //instancia o singleton
    CCerimonies.getInstance().setCerimonies(cerimonies_file_na

    // Now that we have some dummy forecast data,
        create an ArrayAdapter.
    // The ArrayAdapter will take data from a
        source (like our dummy forecast) and
    // use it to populate the ListView it's
        attached to.
    mNewCerimoniesAdapter =
            new ArrayAdapter<String>(
                    getActivity(), // The current
                        context (this activity)
                    R.layout.list_item_newcerimonies,
                        // The name of the layout
                        ID.
                    R.id.list_item_newcerimonies_textview,
                        // The ID of the textview
                        to populate.
                    cerimonies_nice_names);

    View rootView =
        inflater.inflate(R.layout.fragment_list_new_cerimonies
        container, false);

    // Get a reference to the ListView, and
        attach this adapter to it.
    ListView listView = (ListView)
        rootView.findViewById(R.id.listview_newcerimonies);
    listView.setAdapter(mNewCerimoniesAdapter);

    listView.setOnItemClickListener(new
        AdapterView.OnItemClickListener() {
        public void onItemClick(AdapterView<?>
            adapterView, View view, int position,
            long l) {
            //String cerimonyFileName =
                mNewCerimoniesAdapter.getItem(position);
            String cerimonyFileName =
```

```
                    cerimonies_file_names.get(position);
                Intent intent = new
                    Intent(getActivity(),
                    CerimonyDetails.class)
                        .putExtra(Intent.EXTRA_TEXT,
                            cerimonyFileName);
                startActivity(intent);
            }
        });

        return rootView;
    }
}
```

anexos/code/ListStepsFragment.java

```
package tcc.cerimony_assistant_v01;

import android.content.Context;
import android.net.Uri;
import android.os.Bundle;
import android.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;

import java.util.ArrayList;
import java.util.List;


/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must
     implement the
 * {@link
     ListStepsFragment.OnFragmentInteractionListener}
     interface
 * to handle interaction events.
 * Use the {@link ListStepsFragment#newInstance}
     factory method to
 * create an instance of this fragment.
 */
public class ListStepsFragment extends Fragment {


    private ArrayAdapter<String> mListStepsAdapter;
```

```
    public ListStepsFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
                             Bundle savedInstanceState)
                                 {
        // Inflate the layout for this fragment
        Cerimony cerimony =
            CCerimonies.getInstance().getSelectedCerimony();
        List<String> stepsList = new ArrayList<>();
        List<Step> steps = cerimony.getSteps();
        for ( int i=0; i < steps.size(); i++) {
            String stepName = steps.get(i).getSName();
            stepsList.add(stepName);
        }

        mListStepsAdapter =
                new ArrayAdapter<String>(
                        getActivity(),
                        R.layout.list_item_steps,
                        R.id.list_item_steps_textview,
                        stepsList
                );

        View rootView =
            inflater.inflate(R.layout.fragment_list_steps,
            container, false);

        ListView listView = (ListView)
            rootView.findViewById(R.id.listview_steps);
        listView.setAdapter(mListStepsAdapter);

        return rootView;
    }
}
```

anexos/code/ManageParticipantsActivity.java

```
package tcc.cerimony_assistant_v01;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
```

```
public class ManageParticipantsActivity extends
    AppCompatActivity {

    @Override
    protected void onCreate(Bundle
        savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_manage_participants);
        Toolbar toolbar = (Toolbar)
            findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setTitle("Gerenciar
            Participantes");
    }
}
```

anexos/code/ManageParticipantsFragment.java

```
package tcc.cerimony_assistant_v01;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;
```

```
public class ManageParticipantsFragment extends
    Fragment {

    public ManageParticipantsFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container,
                           Bundle savedInstanceState)
                               {
        // Inflate the layout for this fragment
        final View view =
            inflater.inflate(R.layout.fragment_manage_participants
            container, false);
        String jString =
            ParticipantsJSONParser.convertJSONFiletoString();
        if ("".equals(jString)){
            //cria a o jsonArray participants, faz com
                que getParticipants nunca seja null
            jString =
                ParticipantsJSONParser.instantiateJson();
        }
        final String json = jString;
        final List<Participant> participants =
            ParticipantsJSONParser.getParticipantsFromJSON(json);

        TableLayout table =
            (TableLayout)view.findViewById(R.id.tableLayout);

        for(int i = 0; i < participants.size(); i++) {
            TableRow row =
                (TableRow)LayoutInflater.from(getActivity()).infla
                null);
            ((TextView)row.findViewById(R.id.text_name)).setText(p
            ((TextView)row.findViewById(R.id.text_email)).setText(
            ((TextView)row.findViewById(R.id.text_unit)).setText(p
            final Button remove_btn =
                ((Button)row.findViewById(R.id.button_remove_parti
            int id = participants.get(i).getId();
            row.setId(id);

            remove_btn.setOnClickListener(new
                View.OnClickListener() {
                public void onClick(View v) {
                    final View row = (View)
                        v.getParent();
                    TableRow tr = (TableRow) row;
```

```
                    TextView tv = (TextView)
                        tr.getChildAt(0);
                    final String name =
                        tv.getText().toString();

                    new
                        AlertDialog.Builder(getContext())
                            .setTitle("Deletar
                                Participante")
                            .setMessage("Você deseja
                                deletar o participante
                                com o nome
                                \""+name+"\"?")
                            .setIcon(android.R.drawable.ic_dialog_alert)
                            .setPositiveButton(android.R.string.yes,
                                new
                                DialogInterface.OnClickListener()
                                {

                                public void
                                    onClick(DialogInterface
                                    dialog, int
                                    whichButton) {
                                    int rId =
                                        row.getId();
                                    ParticipantsJSONParser.removeJSONParti
                                    ViewGroup container
                                        =
                                        ((ViewGroup)row.getParent());
                                    container.removeView(row);
                                    container.invalidate();
                                    Toast.makeText(getActivity(),
                                        "Participante
                                        \""+name+"\"
                                        deletado com
                                        sucesso!",
                                        Toast.LENGTH_SHORT).show();
                                }})
                            .setNegativeButton(android.R.string.no,
                                null).show();
            }
        });

        table.addView(row, i+1);
    }
    table.requestLayout();

    Button btn = (Button)
        view.findViewById(R.id.button_add_participant);
```

```
btn.setOnClickListener(new
    View.OnClickListener() {
    @Override
    public void onClick(View v) {
        boolean validate = true;
        Participant nParticipant = new
            Participant();
        String name = ((EditText)
            view.findViewById(R.id.textName)).getText().to
        String email = ((EditText)
            view.findViewById(R.id.textEmail)).getText().t
        String unit = ((EditText)
            view.findViewById(R.id.textUnit)).getText().to
        //if name || email || unit == "" não
            permitir adicionar, mostrar toast
            com o campo que não possou da
            validação
        if ("".equals(name) ||
            "".equals(email) ||
            "".equals(unit)) {
            validate = false;
        }

        if (validate) {
            //add participant on json
            nParticipant.setUnidade(unit);
            nParticipant.setPName(name);
            nParticipant.setEmail(email);
            int pId =
                ParticipantsJSONParser.addParticipantToJson
            nParticipant.setId(pId);

            //clean the edit texts
            ((EditText)
                view.findViewById(R.id.textName)).setText("
            ((EditText)
                view.findViewById(R.id.textEmail)).setText(
            ((EditText)
                view.findViewById(R.id.textUnit)).setText("

            TableLayout table =
                (TableLayout)view.findViewById(R.id.tableLa

            TableRow row =
                (TableRow)LayoutInflater.from(getActivity()
                null);
            ((TextView)row.findViewById(R.id.text_name)).se
            ((TextView)row.findViewById(R.id.text_email)).s
```

```
((TextView)row.findViewById(R.id.text_unit)).setText
Button remove_btn =
    ((Button)row.findViewById(R.id.button_remove_par
row.setId(pId);

remove_btn.setOnClickListener(new
    View.OnClickListener() {
    public void onClick(View v) {
        final View row = (View)
            v.getParent();
        TableRow tr = (TableRow)
            row;
        TextView tv = (TextView)
            tr.getChildAt(0);
        final String name =
            tv.getText().toString();

        new
            AlertDialog.Builder(getContext())
                .setTitle("Deletar
                    Participante")
                .setMessage("Você
                    deseja deletar o
                    participante com
                    o nome
                    \""+name+"\"?")
                .setIcon(android.R.drawable.ic_dialog_
                .setPositiveButton(android.R.string.ye
                    new
                    DialogInterface.OnClickListener()
                    {

                    public void
                        onClick(DialogInterface
                        dialog, int
                        whichButton) {
                        int rId =
                            row.getId();
                        ParticipantsJSONParser.removeJS
                        ViewGroup
                            container
                            =
                            ((ViewGroup)row.getParent()
                        container.removeView(row);
                        container.invalidate();
                        Toast.makeText(getActivity(),
                            "Participante
                            \""+name+"\"
                            deletado
```

```
                                                    com
                                                    sucesso!",
                                                    Toast.LENGTH_SHORT).s
                                            }})
                                        .setNegativeButton(android.R.str
                                            null).show();
                                }
                            });


                            table.addView(row,
                                table.getChildCount()-1);
                            table.requestLayout();
                        } else {
                            Context context = getContext();
                            int duration = Toast.LENGTH_SHORT;
                            CharSequence text = "Nenhum campo
                                pode estar vazio";
                            Toast toast =
                                Toast.makeText(context, text,
                                duration);
                            toast.show();
                        }
                    }
                });
                return view;
            }
        }
```

anexos/code/Participant.java

```
package tcc.cerimony_assistant_v01;

/**
 * Created by zr on 31/01/17.
 */

public class Participant {
    private int id;
    private boolean isEditor;
    private boolean isCreator;
    private String name;
    private String cargo;
    private String unidade;
    private String email;

    public int getId() {
        return id;
```

```
    }

    public void setId(int id) {
        this.id = id;
    }

    public boolean isEditor() {
        return isEditor;
    }

    public void setEditor(boolean isEditor) {
        this.isEditor = isEditor;
    }

    public boolean isCreator() {
        return isCreator;
    }

    public void setCreator(boolean isCreator) {
        this.isCreator = isCreator;
    }

    public String getPName() {
        return name;
    }

    public void setPName(String name) {
        this.name = name;
    }

    public String getCargo() {
        return cargo;
    }

    public void setCargo(String cargo) {
        this.cargo = cargo;
    }

    public String getUnidade() {
        return unidade;
    }

    public void setUnidade(String unidade) {
        this.unidade = unidade;
    }

    public String getEmail() {
        return email;
    }
```

```
    public void setEmail(String email) {
        this.email = email;
    }
}
```

anexos/code/ParticipantsFragment.java

```
package tcc.cerimony_assistant_v01;


import android.content.Context;
import android.os.Bundle;
import android.app.Fragment;
import android.os.Environment;
import android.support.v4.view.ViewPager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Checkable;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;


/**
 * A simple {@link Fragment} subclass.
 */
public class ParticipantsFragment extends
    android.support.v4.app.Fragment {
    private static final String TAG =
```

```
        "ParticipantsFragment";

    public ParticipantsFragment() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle
        savedInstanceState) {
        final View view =
            inflater.inflate(R.layout.fragment_participants,
            container, false);

        //convert participants.json in a string
        File file = new
            File(Environment.getExternalStorageDirectory(),
            "ceremony-assistant/participants.json");
        FileInputStream stream = null;
        try {
            stream = new FileInputStream(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        String jString = "";
        try {
            FileChannel fc = stream.getChannel();
            MappedByteBuffer bb = null;
            try {
                bb =
                    fc.map(FileChannel.MapMode.READ_ONLY,
                    0, fc.size());
            } catch (IOException e) {
                e.printStackTrace();
            }
            /* Instead of using default, pass in a
                decoder. */
            jString =
                Charset.defaultCharset().decode(bb).toString();
        }
        finally {
            try {
                stream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
```

```
//parse participants.json string to
    participants object
final ArrayList<Participant> participants =
    ParticipantsJSONParser.getParticipantsFromJSON(jString

//display dinamycally the participants in a
    tablelayout
TableLayout table =
    (TableLayout)view.findViewById(R.id.tableLayout);
//CheckBox cb;
Spinner spinner;
//final ArrayList<CheckBox> cbData = new
    ArrayList<CheckBox>();
final ArrayList<Spinner> spinnerData = new
    ArrayList<Spinner>();
for(int i = 0; i < participants.size(); i++) {
    TableRow row =
        (TableRow)LayoutInflater.from(getActivity()).infla
        null);
    ((TextView)row.findViewById(R.id.text_participant)).se
    //cb =
        (CheckBox)row.findViewById(R.id.check_box_particip
    spinner =
        (Spinner)row.findViewById(R.id.spinner_participant
    spinnerData.add(spinner);
  //  cbData.add(cb);
    table.addView(row);
}
table.requestLayout();

Button btn = (Button)
    view.findViewById(R.id.confirm_participants_button);

btn.setOnClickListener(new
    View.OnClickListener() {
    @Override
    public void onClick(View v) {
    Participant participant;
    List<Participant> cer_participants = new
        ArrayList<Participant>();

    for(int i = 0; i < participants.size();
        i++) {
        String pRole =
            spinnerData.get(i).getSelectedItem().toString()
        if
            (!"Nenhuma".equalsIgnoreCase(pRole))
            {
            participant = participants.get(i);
```

```
                    participant.setCargo(pRole);
                    cer_participants.add(participant);
                }
            }
            CharSequence feedback_text;
            if(cer_participants.size() == 0) {
                feedback_text = "Nenhum participante
                    foi confirmado";
                //display feedback message
                Context context = getContext();
                int duration = Toast.LENGTH_SHORT;

                Toast toast = Toast.makeText(context,
                    feedback_text, duration);
                toast.show();
            } else {
                feedback_text = "Participantes
                    confirmados com sucesso";
                CCerimonies.getInstance().getSelectedCerimony().setParti
                //display feedback message
                Context context = getContext();
                int duration = Toast.LENGTH_SHORT;

                Toast toast = Toast.makeText(context,
                    feedback_text, duration);
                toast.show();
                ViewPager viewPager = (ViewPager)
                    getActivity().findViewById(R.id.vp);
                viewPager.setCurrentItem(1);
            }

            }
        });
        return view;
    }

}
```

anexos/code/ParticipantsJSONParser.java

```
package tcc.cerimony_assistant_v01;

import android.os.Environment;
import android.util.Log;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
```

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.charset.Charset;
import java.util.ArrayList;

/**
 * Created by zr on 15/02/18.
 */

public class ParticipantsJSONParser {
    private static final String TAG_PARTICIPANTS =
        "participants";
    private static final String TAG_ID = "id";
    private static final String TAG_SNAME = "sname";
    private static final String TAG_SEMAIL = "semail";
    private static final String TAG_SORGANIZATION =
        "sorganization";
    public static String convertJSONFiletoString() {
        //convert participants.json in a string
        //se o arquivo não existe, cria o arquivo
        File file = new
            File(Environment.getExternalStorageDirectory(),
            "ceremony-assistant/participants.json");
        try {
            file.createNewFile();
        } catch (IOException e) {
            e.printStackTrace();
        }
        FileInputStream stream = null;
        try {
            stream = new FileInputStream(file);
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        String jString = "";
        try {
            FileChannel fc = stream.getChannel();
            MappedByteBuffer bb = null;
            try {
                bb =
                    fc.map(FileChannel.MapMode.READ_ONLY,
                    0, fc.size());
            } catch (IOException e) {
```

```
                e.printStackTrace();
            }
            /* Instead of using default, pass in a
                decoder. */
            jString =
                Charset.defaultCharset().decode(bb).toString();
        }
        finally {
            try {
                stream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return jString;
    }

    public static String instantiateJson() {
        JSONObject jsonObj = new JSONObject();
        JSONArray jsonArray = new JSONArray();
        try {
            jsonObj.put("participants", jsonArray);
        } catch (JSONException e) {
            e.printStackTrace();
        }
        String jParticipants = jsonObj.toString();
        writeJson(jParticipants);

        return jParticipants;
    }

    public static boolean writeJson(String json) {
        String path =
            Environment.getExternalStorageDirectory()
            + "/ceremony-assistant/participants.json";
        try (FileWriter file = new FileWriter(path)) {
            file.write(json);
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }

        return true;
    }
    public static ArrayList<Participant>
        getParticipantsFromJSON(String json) {
        if (json != null) {
            try {
```

```
                ArrayList<Participant> participantList
                    = new ArrayList<Participant>();
                JSONObject jsonObj = new
                    JSONObject(json);

// Getting JSON Array node
                JSONArray participants =
                    jsonObj.getJSONArray(TAG_PARTICIPANTS);

// looping through All Participants
                for (int i = 0; i <
                    participants.length(); i++) {
                    JSONObject c =
                        participants.getJSONObject(i);
                    int id = c.getInt(TAG_ID);
                    String sname =
                        c.getString(TAG_SNAME);
                    String email =
                        c.getString(TAG_SEMAIL);
                    String sorganization =
                        c.getString(TAG_SORGANIZATION);


// tmp hashmap for single student
                    Participant participant = new
                        Participant();
                    participant.setId(id);
                    participant.setUnidade(sorganization);
                    participant.setPName(sname);
                    participant.setEmail(email);


// adding student to students list
                    participantList.add(participant);
                }
                return participantList;
            } catch (JSONException e) {
                e.printStackTrace();
                return null;
            }
        } else {
            Log.e("JSONNULL", "No participants added
                in participants.JSON");
            return null;
        }
    }

    public static String
        ParticipantsToJson(ArrayList<Participant>
```

```
        pList) {
        return "";
    }

    public static int
        addParticipantToJson(Participant participant)
        {
        String json = convertJSONFiletoString();

        if (json != null) {
            try {

                ArrayList<Participant> participantList
                    = new ArrayList<Participant>();
                JSONObject jsonObj = new
                    JSONObject(json);

// Getting JSON Array node
                JSONArray jParticipants =
                    jsonObj.getJSONArray(TAG_PARTICIPANTS);
                int id = 0;
                //get last id
                if(jParticipants.length() > 0) {
                    JSONObject c =
                        jParticipants.getJSONObject(jParticipants.length
                    id = c.getInt(TAG_ID) + 1;
                }

                JSONObject jParticipant = new
                    JSONObject();
                jParticipant.put("id", id);
                jParticipant.put("sname",
                    participant.getPName());
                jParticipant.put("semail",
                    participant.getEmail());
                jParticipant.put("sorganization",
                    participant.getUnidade());
                jParticipants.put(jParticipant);
                writeJson(jsonObj.toString());
                return id;
            } catch (JSONException e) {
                e.printStackTrace();
                return -1;
            }
        } else {
            Log.e("JSONNULL", "No participants added
                in participants.JSON");
            return -1;
        }
```

```
    }

    public static boolean
        removeJSONParticipantById(int id) {
        try{
            String json = convertJSONFiletoString();
            if(!"".equals(json)) {
                JSONObject jsonObj = new
                    JSONObject(json);
                JSONArray jParticipants =
                    jsonObj.getJSONArray(TAG_PARTICIPANTS);
                jParticipants.remove(id);

//              int len = jParticipants.length();
//              if (jParticipants != null) {
//                  for (int i=0;i<len;i++)
//                  {
//                      //Excluding the item at
    position
//                      JSONObject c =
    jParticipants.getJSONObject(i);
//                      int c_id = c.getInt(TAG_ID);
//                      if (c_id != id)
//                      {
//
    list.put(jParticipants.get(i));
//                      }
//                  }
//              }
                JSONObject jsonFinal = new
                    JSONObject();
                jsonFinal.put("participants",
                    jParticipants);

                writeJson(jsonFinal.toString());
            } else {
                return false;
            }
        } catch (JSONException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
}
```

anexos/code/RequirementsFragment.java

```
package tcc.cerimony_assistant_v01;


import android.content.Context;
import android.os.Bundle;
import android.app.Fragment;
import android.os.Environment;
import android.support.v4.view.ViewPager;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.Checkable;
import android.widget.Spinner;
import android.widget.TableLayout;
import android.widget.TableRow;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;


/**
 * A simple {@link Fragment} subclass.
 */
public class RequirementsFragment extends
    android.support.v4.app.Fragment {
    private static final String TAG =
        "RequirementsFragment";

    public RequirementsFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle
        savedInstanceState) {
```

```
final View view =
    inflater.inflate(R.layout.fragment_requirements,
    container, false);

//display dinamycally the requirements in a
    tablelayout
Cerimony ceremony =
    CCerimonies.getInstance().getSelectedCerimony();
List<String> requirements =
    ceremony.getRequirements();
TableLayout table =
    (TableLayout)view.findViewById(R.id.tableLayout);
CheckBox cb;

final ArrayList<CheckBox> cbData = new
    ArrayList<CheckBox>();
for(int i = 0; i < requirements.size(); i++) {
    TableRow row =
        (TableRow)LayoutInflater.from(getActivity()).infla
        null);
    ((TextView)row.findViewById(R.id.text_requirement)).se
    cb =
        (CheckBox)row.findViewById(R.id.check_box_requirem
    cbData.add(cb);
    table.addView(row);
}
table.requestLayout();

Button btn = (Button)
    view.findViewById(R.id.confirm_requirements_button);

btn.setOnClickListener(new
    View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int count = 0;
        for(int i = 0; i < cbData.size(); i++)
            {
            if (cbData.get(i).isChecked()) {
                count++;
            }
        }
        if(cbData.size() == count) {
            CCerimonies.getInstance().getSelectedCerimony()
            //display feedback message
            Context context = getContext();
            CharSequence text = "Requisitos
                confirmados com sucesso!";
            int duration = Toast.LENGTH_SHORT;
```

```
                    Toast toast =
                        Toast.makeText(context, text,
                        duration);
                    toast.show();
                    ViewPager viewPager = (ViewPager)
                        getActivity().findViewById(R.id.vp);
                    //update the content of
                        ceremonydetailsfragment #pog
                    android.support.v4.app.Fragment
                        fragment =
                        ((CeremonyDetailsPageAdapter)
                        viewPager.getAdapter()).getItem(0);
                    ((CerimonyDetailsFragment)fragment).update();
                    viewPager.setCurrentItem(0);
                } else {
                    CCerimonies.getInstance().getSelectedCerimony().setC
                    Context context = getContext();
                    CharSequence text = "Falta
                        confirmar alguns requisitos!";
                    int duration = Toast.LENGTH_SHORT;

                    Toast toast =
                        Toast.makeText(context, text,
                        duration);
                    toast.show();
                }
            }
        });
        return view;
    }

}
```

anexos/code/Step.java

```
package tcc.cerimony_assistant_v01;

import java.util.ArrayList;
import java.util.List;

/**
 * Created by zr on 31/01/17.
 */

public class Step {
    private String id;
    private String name;
```

```
private String description;
private String inputType;
private String outputType;
private String observation = "";
private String time = "";
private String input;
private String output;
private String notes = "";
private String evidence = "";

public String getEvidence() {
    return evidence;
}

public void setEvidence(String evidence) {
    this.evidence = evidence;
}

public String getNotes() {
    return notes;
}

public void setNotes(String notes) {
    this.notes = notes;
}

public String getTime() {
    return time;
}

public void setTime(String time) {
    this.time = time;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getSName() {
    return name;
}

public void setSName(String name) {
    this.name = name;
}
```

```
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public String getInputType() {
        return inputType;
    }

    public void setInputType(String inputType) {
        this.inputType = inputType;
    }

    public String getOutputType() {
        return outputType;
    }

    public void setOutputType(String outputType) {
        this.outputType = outputType;
    }

    public String getObservation() {
        return observation;
    }

    public void setObservation(String observation) {
        this.observation = observation;
    }

    public String getInput() {
        return input;
    }

    public void setInput(String input) {
        this.input = input;
    }

    public String getOutput() {
        return output;
    }

    public void setOutput(String output) {
        this.output = output;
    }
}
```

anexos/code/StepsListAdapter.java

```
package tcc.cerimony_assistant_v01;

import android.app.Activity;
import android.graphics.Color;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.EditText;
import android.widget.TextView;

import java.util.List;

/**
 * Created by zr on 14/05/18.
 */

public class StepsListAdapter extends BaseAdapter {
    private final List<StepVisual> steps;
    private final Activity activity;

    public StepsListAdapter(List<StepVisual> steps,
        Activity activity) {
        this.steps = steps;
        this.activity = activity;
    }

    @Override
    public int getCount() {
        return steps.size();
    }

    @Override
    public Object getItem(int position) {
        return steps.get(position);
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View
        convertView, ViewGroup parent) {
        StepVisual cur_step_visual =
            steps.get(position);
        String stepName = cur_step_visual.getName();
        View view = activity.getLayoutInflater()
```

```
                .inflate(R.layout.list_item_steps,
                    parent, false);

        TextView stepName_tv = (TextView)
            view.findViewById(R.id.list_item_steps_textview);
        stepName_tv.setText(stepName);
        if(cur_step_visual.isExecuted()){
            stepName_tv.setBackgroundColor(Color.LTGRAY);
        } else {
            stepName_tv.setBackgroundColor(android.R.drawable.btn_defau
        }

        return view;
    }
}
```

anexos/code/StepVisual.java

```
package tcc.cerimony_assistant_v01;

/**
 * Created by zr on 14/05/18.
 */

public class StepVisual {
    private String name;
    private boolean executed;

    public StepVisual(String name, boolean executed) {
        this.name = name;
        this.executed = executed;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isExecuted() {
        return executed;
    }

    public void setExecuted(boolean executed) {
        this.executed = executed;
    }
```

```
}
```