UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE CIÊNCIAS E TECNOLOGIA
COORDENAÇÃO DE ESTÁGIOS DO DEE

Estágio em Análise de Testes

ESTAGIÁRIA: VANESSA BATISTA DE SOUSA SILVA
ORIENTADORA: FERNANDA CECÍLIA LOUREIRO

Campina Grande – Paraíba

Biblioteca Setorial do CDSA. Março de 2021.

Sumé - PB

Agradecimentos

Agradeço a minha família e Deus por ter me dado toda a estrutura necessária à minha formação

Agradeço a todos os professores e funcionários do Departamento de Engenharia Elétrica, em especial ao Professor Edson Guedes e toda a equipe do Laboratório de Alta Tensão que contribuíram para o enriquecimento da minha formação mostrando sempre muito profissionalismo e amizade.

Aos meus grandes amigos que foram determinantes durante a dura jornada do curso. Em especial a Danyelle Mousinho e João Marcelo que me acompanharam durante todo o curso e que junto comigo formavam "O trio". Agradeço às companheiras do Centro Acadêmico de Engenharia Elétrica (CAEE).

Aos meus companheiros de trabalho do CIn/Motorola, em especial aos amigos da equipe de testes de TDMA, Luckerson Cruz, Waleska Dantas, Juliana Loureiro, Luciano Lopes e Alessandra Araújo. Ao meu "Team Leader" Edson Fontes, que sempre agiu como um verdadeiro Líder.

Gostaria de agradecer a Motorola e ao Centro de Informática da UFPE pela criação do Programa de Imersão Tecnológica que promove a formação de profissionais habilitados na área de teste de software e ainda incentiva o desenvolvimento de pesquisa na área de tecnologia da informação, ao mesmo tempo em que oferece toda a estrutura necessária ao desenvolvimento de um programa de sucesso.

Gostaria de agradecer aos professores do curso de Análise de Teste pela paciência e dedicação durante a orientação que recebemos para desenvolvermos este projeto. E a todos os profissionais do CIn/Motorola pelo companheirismo e pela amizade.

Resumo

O estágio foi realizado no laboratório montado para atender ao Programa de Imersão Tecnológica (PIT). O PIT é resultado de uma parceria entre a Motorola e o Centro de Informática da Universidade Federal de Pernambuco (CIn-UFPE).

O objetivo do PIT é "Fomentar a formação de recursos humanos com alto grau de especialização na área de software embutido para testes de aplicações em computação móvel, de forma a viabilizar a atração de empresas e a criação de um centro de excelência em software básico no Recife, usando os incentivos e benefícios previstos na Lei 10.176".

O programa é dividido em duas partes: teoria e laboratório. Na parte teórica, são oferecidos, aos participantes do programa, cursos na área de Engenharia de Software com ênfase em Análise de Testes ministrados pela Qualiti, empresa pernambucana de consultoria. A avaliação dos participantes é feita através de um exame, cuja nota deve ser maior ou igual a 7,0 (sete). A reprovação ou desistência do aluno de cursar disciplinas implica na sua exclusão imediata do PIT.

Na parte de laboratório, são realizados os testes nos celulares Motorola. Nas duas primeiras semanas do estágio foi realizado um treinamento cujo objetivo foi habilitar os participantes a realizarem os testes de acordo com o padrão Motorola. Os testes são feitos em parceria com diversos centros Motorola espalhados pelo mundo.

Também faz parte do programa o desenvolvimento de uma monografia. O tema escolhido para a monografia deve estar de acordo com os interesses da Motorola. A orientação ao desenvolvimento é feita por professores do Centro de Informática da UFPE.

Em respeito à filosofia da empresa, não vai ser possível detalhar as atividades de testes.

A Motorola

A Motorola é líder mundial em soluções integradas de comunicação e de eletrônica. As vendas mundiais da empresa em 2001 foram de 30 bilhões de dólares.

Desde 1995 a Motorola vem investindo cerca de US$ 210 milhões principalmente na instalação de um complexo industrial, localizado em Jaguariúna, interior de São Paulo e ampliação de sua capacidade produtiva e tecnológica. Com fábricas de celulares, equipamentos iDEN, rádios bidirecionais e infra-estrutura celular (estações rádio base), o Campus ainda conta com um Centro de Tecnologia de Semicondutores, um Centro de Pesquisa e Desenvolvimento de Terminais Celulares em hardware, software, mecânica e desenho industrial, além de uma base da Motorola University para treinamento de funcionários e à consultoria do programa Seis Sigma para o mercado como um todo. A empresa também está fortemente no país comercializando semicondutores e desenvolvendo diversas soluções para os mercados de banda larga, TV a cabo e automotivo.

O CIn

Inovação, excelência e criatividade aliados com a competência e experiência de um conceituado corpo docente fazem do Centro de Informática da UFPE um dos melhores centros acadêmicos de informática da América Latina. Apesar de ser um dos mais novos da UFPE (criado oficialmente em 1999), o CIn conta com a expertise de mais de 25 anos de funcionamento, já que é a evolução natural do antigo Departamento de Informática.

Atualmente, o CIn oferece curso de bacharelado em Ciência da Computação. O reconhecimento nacional e internacional vem, principalmente, do alto nível acadêmico do seu corpo docente.

Hoje atuam no corpo docente do CIn 46 doutores. Estão matriculados no Centro cerca de 600 alunos de graduação, 123 de mestrado, 65 alunos de doutorado e 165 de especialização. A Infra-Estrutura do CIn conta com mais de 500 pontos de trabalho interligados em Rede, distribuídos nos 15 laboratórios, nas salas de aulas e dos professores.

Índice

1. Introdução

Será apresentada no relatório uma descrição simplificada das atividades desenvolvidas no estágio. O detalhamento das atividades não poderá ser feito em respeito à filosofia da empresa de preservar às informações pertinentes ao seu processo.

Será também apresentada uma descrição das disciplinas do Curso Seqüencial de Formação Complementar que foi realizado paralelamente ao estágio. O curso aborda conhecimentos em Engenharia de Software com ênfase em Análise de Testes. Ao término do curso de J2ME (Java to Micro Edition) foi proposto o desenvolvimento de uma aplicação para celular utilizando a lingugem J2ME. O detalhamento do projeto desenvolvido será apresentado no relatório.

A principal atividade desenvolvida no estágio foi testes de software para celulares. O Fluxo de Teste faz parte de um conjunto de atividades realizadas no desenvolvimento dos produtos Motorola. O objetivo principal do Fluxo de Teste é assegurar a total satisfação do cliente com o produto desenvolvido.

No Relatório, será suposto que o desenvolvimento do processo Motorola é feito de acordo com a "Metodologia" RUP (Rational Unified Process). Esta suposição é necessária para que possamos descrever o estágio sem detalhar informações pertinentes exclusivamente à Motorola.

De acordo com o RUP, além do Fluxo de Teste, existem também os fluxos de Requisitos, Análise e Projeto, Implementação, Implantação, Gerência de Configuração e Planejamento e Gerenciamento. Estas atividades são desenvolvidas com maior ou menor ênfase em cada uma das fases do processo de desenvolvimento dos produtos: Concepção, Elaboração, Construção e Transição.

O planejamento inicial dos Testes ocorre na fase da Concepção, durante o planejamento do projeto.

Na fase de Elaboração, o foco principal do Fluxo de Teste é o projeto e execução de testes de integração, de forma a validar e estabelecer uma arquitetura estável para o processo.

Na fase de Construção, o foco principal do Fluxo de Teste é o projeto e execução dos testes de sistema dos diversos requisitos implementados no produto.

Na fase de Transição, o foco do Fluxo de Teste muda para homologação e avaliação das correções que foram feitas após os testes aplicados nas fases anteriores. Os testes realizados nessa

fase são realizados fora do ambiente de teste. Normalmente são realizados pelos usuários finais. O controle desta atividade é feito através da avaliação repassada dos usuários finais para a Motorola.

2. Teste de Sistema

O Documento de Especificação é um documento de registro de todas os requisitos associados a um produto. Requisito é algo que o produto tem que ter ou fazer para atender às necessidades intrínsecas ao produto e às solicitações do cliente. Os requisitos são classificados em funcionais e não funcionais.

Requisitos funcionais são aqueles que expressam o que o produto deve fazer. Eles são derivados dos casos de uso e são diretamente relacionados às funcionalidades do produto.

Requistos não funcionais são aqueles que expressam propriedades ou qualidades que o produto deve ter. Eles são relacionados ao comportamento das funcionalidades.

Além dos requisitos funcionais e não funcionais, existem também as restricões associadas ao produto. As restricões são problemas que modelam os requisitos, ou seja, para um requisito ser validado, ele deve atender a todas as restricões do projeto.

Todos os requisitos e restrições devem estar registrados e descritos precisamente em um documento de especificação de requisitos. O Documento de Especificação é elaborado na fase de Concepção e serve como base para fluxos do processo de desenvolvimento, Análise e Projeto, Implementação, Testes, Implantação. No fluxo de Testes, o Documento de Especificação é utilizado para elaborar planos, casos e procedimentos de teste.

O Fluxo de Teste deve verificar se todos os requisitos do sistema foram corretamente implantados e se houve a correta integração entre todos os componentes do software. Os teste também detectam falhas humanas injetadas no processo. Os resultados dos testes devem ser comparados com os resultados esperados, descritos no Documento de Especificação, a fim de produzir uma indicação da qualidade e da confiabilidade software.

Um Fluxo de Teste bem planejado garante a qualidade do software, redução nos custos de manutenção corretiva e, principalmente, a satisfação do cliente.

O Fluxo de Teste é dividido em quatro estágios de teste: Teste de Unidade, Teste de Integração, Teste de Sistema e Teste de Aceitação (homologação). No estágio, foram realizados os

Testes de Sistema. Os Testes de Sistema são realizados com maior ênfase na fase de Construção, como mostrado na Figura 1.

No Teste de Sistema é verificado se a aplicação está funcionando como um todo, considerando a integração dos componentes de software com o ambiente operacional similar ao de produção, ou seja, o software é integrado ao hardware do celular e então é testado.

Os Testes de Sistema são realizados a partir de uma análise entre os dados de entrada e saída com base nos requisitos do sistema que estão registrados no Documento de Especificação.

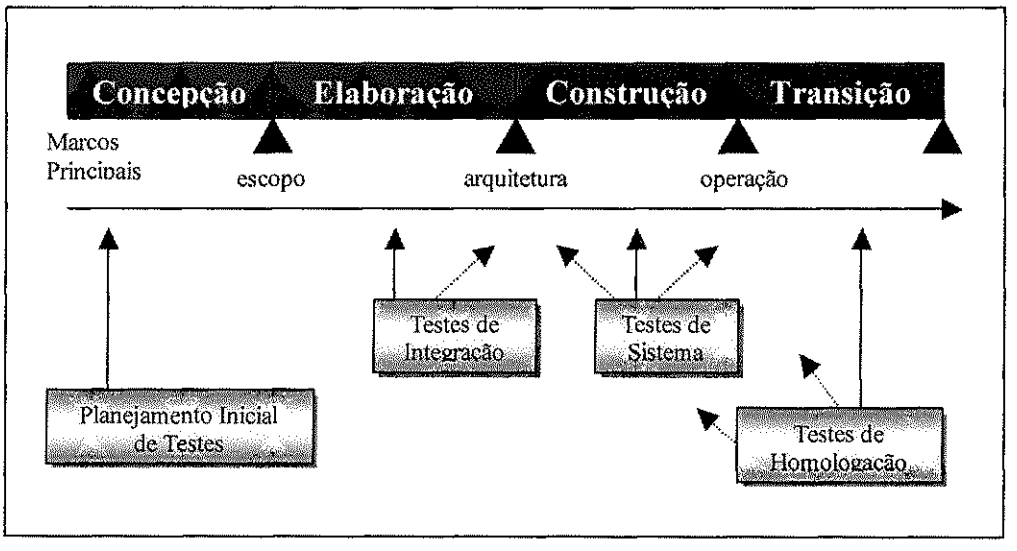

Fig. 1: Testes e Relação com as Fases de Desenvolvimento

2.1. Etapas do Teste de Sistema

Os Testes de Sistema são realizados em paralelo ao desenvolvimento do produto. Trata-se de um processo contínuo que é iniciado nas fases iniciais de desenvolvimento. Durante o ciclo de vida de desenvolvimento, varia atividades da etapa de Teste de Sistema são realizadas.

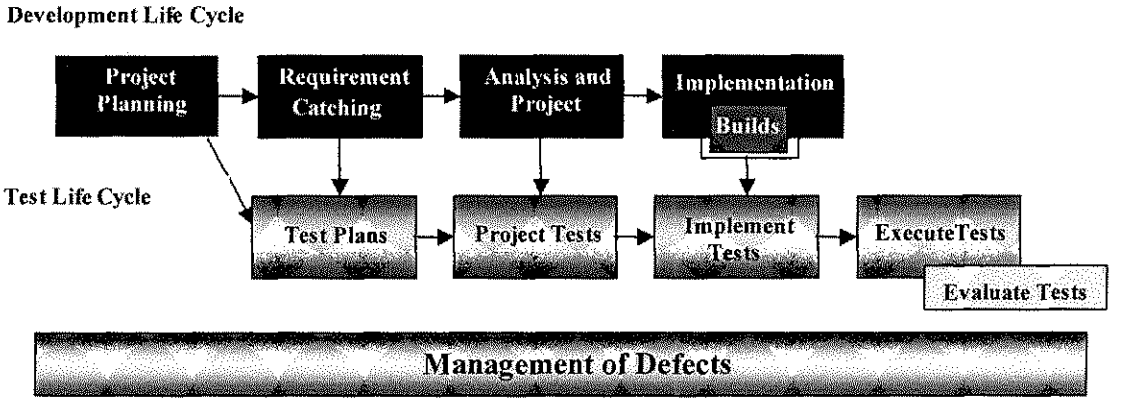

Fig.2: Ciclo de vida do Teste de Sistema

2.1.1. Planejar Testes

Na etapa de Planejar Testes, a equipe de desenvolvimento fornece à equipe de teste o documento de especificação dos requisitos do sistema. De posse do documento de especificação, a equipe de teste identifica os requisitos a serem testados, define as prioridades de cada requisito, define as estratégias de teste, define os recursos e elabora cronograma da atividade de teste. Ao final desta etapa é gerado o documento Plano de Teste.

Na identificação, cada requisito é identificado no documento de especificação através de um código criado com base no nome da função ao qual pertence o requisito, por exemplo, o primeiro requisito identificado na descrição da função Phonebook é identificado por PB_0001. Os requisitos identificados ficam registrados no documento de especificação.

Na definição de prioridades, inicialmente é feita a avaliação dos riscos associados a uma falha na satisfação dos requisitos, sobre as perspectivas de Impacto de falha do requisito e de Probabilidade de ocorrência de algum problema relacionado ao requisito. O Risco avaliado é determinado através da média aritmética dos fatores de Impacto e Probabilidade arredondando para o número inteiro maior e mais próximo. A avaliação de prioridades dos testes baseia-se em três fatores, Risco avaliado, Freqüência e Importância. Sendo determinada através da média aritmética destes fatores arredondando o resultado para o número inteiro maior e mais próximo. A Freqüência é relaciona a quantidade de vezes que o requisito é executado durante um período de utilização do sistema. A Importância é determinada com base na comparação de requisitos.

Os fatores utilizados na avaliação são:

Nível	Prioridade
4	Muito Alto
3	Alto
2	Médio
1	Baixo
0	Muito Baixo

Na definição da estratégia de teste, são previstos os testes que serão necessários com seus respectivos objetivos. Para cada teste previsto, é feita uma verificação dos requisitos que serão testados do em cada teste. Ao final da verificação, é realizado um mapeamento dos requisitos. O objetivo do mapeamento é detectar requisitos que ainda não foram associados a nenhum dos testes previstos. Em seguida, novos testes são previstos de forma que todos os requisitos possam ser testados.

A definição de recursos e elaboração de cronograma é uma tarefa realizada pelo Team Leader. São definidos recursos humanos (principais e de apoio), o ambiente de teste (hardware e software), ferramentas de teste. É calculado o esforço associado às atividades de teste, com base em estimativas e históricos. São definidos os marcos de referência.

2.1.2. Projetar Testes

Na etapa de Projetar Testes, são criados os casos de teste que foram previstos na etapa de Planejar Testes e as planilhas de teste. Os casos de teste são identificados de acordo com sua funcionalidade. Para cada caso de teste, são apresentados objetivos do teste, lista dos requisitos que estão sendo testados, condições iniciais e finais de execução, configurações necessárias à realização do teste, procedimento de teste e resultados esperados. Os casos de teste devem ser criados de modo a serem executados de forma simples. É importante que cada caso de teste cumpra realmente o que está proposto no seu objetivo. Devem ser criados casos de teste para mostrar que o software não faz o que realmente não é suposto fazer, por exemplo, testar que as entradas de Phonebook não são apagadas quando o celular é desligado.

Um bom caso de teste deve satisfazer aos seguintes critérios:

a) Possuir probabilidade razoável de detectar erro: Ao elaborar o caso de teste, o testador deve prever como o software pode falhar e como a falha pode ser capturada. Ao elaborar um caso de teste, é importante frisar que o objetivo é encontrar falhas e que estas falhas devem ser tornadas notáveis.

b) Não deve ser redundante: Se dois casos de teste se propõem a detectar não é necessário executar os dois.

c) Não deve ser nem tão simples nem tão complexo: É interessante economizar tempo de execução de teste combinando dois ou mais testes em um único caso de teste. Porém deve-se ter o cuidado de não criar um caso de teste muito complexo, difícil de executar e entender, isto pode onerar muito tempo para a criação do caso de teste.

Para cada funcionalidade do celular é criado um documento chamado de Feature Test Design (FTD). Os FTD são identificados por um código e pela sua versão. Os FTD contém uma tabela de *Revision History* para controle de versão do documento. A tabela contém o número da versão, o local onde o respectivo documento está armazenado na rede Motorola, a data de modificação, o nome do responsável pela modificação, e a razão da modificação, ou seja, o número da CR (Change Request) que tratou a solicitação da modificação. Na introdução do FTD são apresentadas visão geral, escopo, finalidade, audiência, aplicabilidade, fontes e referências, tabela de termos, abreviações e acrônimos.

Na Figura 3 é mostrado um exemplo de um caso de teste da funcionalidade *Speed Dial*.

SPDL-0001: Store & Access Valid Phonebook Entries

Objective:

Setup:

Initial Conditions:

Final Conditions:

Cleanup: None.

Note: None.

STEP	PROCEDURE	EXPECTED RESULTS

Fig.3: Modelo de Caso de Teste

Para cada FTD é gerada uma planilha de teste. Nas planilhas devem estar contidos dados empregados na execução do caso de teste e os valores resultantes da execução.

2.1.3. Implementar Testes

Na etapa de Implementação são criados códigos para automatizar a execução dos casos de teste. Para saber se um teste falhou ou não, o testador deve comparar o resulatado de cada passo do caso de teste com o resultado esperado que deve ser lido cuidadosamente. A automação de casos de teste contribui para a redução de falhas humanas inseridas durante a execução de testes manuais, com isso, obtém-se o aumento da qualidade, da produtividade da atividade de teste e da confiabilidade do software.

Uma das ferramentas de automação de testes desenvolvidas pela Motorola e utilizadas no estágio para realizar testes automáticos nos celulares Motorola é o PTF (Phone Test Framework). O PTF é rodado num microcomputador com o Java instalado e se comunica com o aparelho celular através de uma interface serial (RS-232) ou de uma interface USB. Em alguns casos de testes são necessários dois aparelhos, usando duas interfaces de comunicação do microcomputador.

2.1.4. Executar Testes

Projetado os casos de teste, inicia-se a etapa de Execução que é realizada até que todas as falhas detectadas pelos testes sejam resolvidas e o celular esteja em condições de ser liberado para as operadoras de telefonia móvel.

Nesta etapa, os casos de teste são executados e são registrados na planilha os estados *pass*, *fail* e *block*. Se na execução do teste, o celular se comporta de acordo com o que está descrito no caso de teste, diz-se que o teste está *pass*. Se o teste não pode ser executado o estado é *block*. Se o celular não se comporta de acordo com o que está descrito no caso de teste, o teste está *fail*. Para cada teste que falha é criado um registro de solicitação de mudança chamado de CR (Change Request) cujo objetivo é solicitar à equipe de desenvolvimento a mudança necessária ao correto funcionamento do celular. A CR deve conter as possíveis causas ou ponto onde se originou o problema e devem ser apresentadas sugestões de correção.

As mudanças solicitadas podem ser mudanças na Build do celular, no documento de especificação ou no FTD. A Motorola possui um processo de controle de mudanças bastante rígido. Abaixo é apresentado a estrutur de um documento de solicitação de mudança (CR).

Estrutura de uma CR:

1. Identificador da solicitação: Código que identifica unicamente a CR
2. Identificação do solicitante
3. Identificação do Sistema: Versão da Build, Hardware, Funcionalidade, Caso de Teste.
4. Classificação: Normalmente os tipos de mudança realizados são adição de nova funcionalidade, melhoria de funcionalidade já existente e correção de defeitos.
5. Descrição: A descrição deve ser bem clara e incluir toda a informação necessária para implementar a mudança.
6. Status: A situação atual da mudança, por exemplo, aprovada, rejeitada, em implantação, postergada, etc.
7. Observações Gerais

2.1.5. Avaliar Testes

Na etapa de Avaliar Testes são realizadas as seguintes atividades: Registrar os resultados gerais de teste; Avaliar procedimentos de teste; Avaliar cobertura dos testes; Analisar resultados gerais; Atualizar solicitações de mudanças. Todas as atividades são realizadas pelo Team Leader da equipe.

3. O Curso

O Curso Seqüencial de Formação Complementar teve duração de 252 horas, divididas entre 12 mini-cursos ministrados por profissionais da Qualiti. As aulas aconteceram em um período do expediente. Uma semana após o encerramento de um mini-curso, iniciava-se outro. Cada mini-curso possuia uma avaliação, podendo ser uma prova tradicional realizada no final do mini-curso ou um projeto realizado em paralelo ao mini-curso. A nota da avaliação devia ser no mínimo 7,0 (sete). No caso de prova tradicional, se a nota obtida fosse inferior a 7,0 (sete), era realizada uma segunda avaliação. Se a nota fosse inferior a 7,0 (sete), o aluno perderia o Curso Seqüencial de Formação Complementar.

Paralelo aos mini-cursos, foi desenvolvido um projeto sobre a orientação de professores do Centro de Informática da UFPE. Foram dedicados 240 horas para o desenvolvimento do projeto que deveria atender às necessidades da Motorola. No meu caso, foi proposto, sobre a orientação do professor Sérgio Cavalcante, o desenvolvimento de uma ferramenta que produz o documento de especificação de requisitos de software estruturado em janelas que simulam todas as telas do celular, com os respectivos requisitos, softkeys e telas transientes associados a cada uma das telas do celular. A justificativa do meu projeto é otimizar a atividade de criação de casos de teste, resultando num processo mais rápido, mais barato e eficiente.

Abaixo é apresentada uma tabela com uma breve descrição dos mini-cursos.

Nome
Fundamentos de um Processo Iterativo e Incremental
Objetivos
Introduzir problemas atuais em engenharia de software; Apresentar modelos de ciclo de vida; Apresentar os conceitos básicos do RUP; Apresentar boas práticas de desenvolvimento.
Resumo
Apresentou alguns dos principais problemas de engenharia de software, que justificam a criação do RUP (Rational Unified Process). RUP é um conjunto de processos que reúne as melhores práticas em engenharia de software para garantir que um produto de software tenha qualidade. O RUP é uma plataforma de processos que fornece atividades, artefatos e guias ligados às ferramentas Rational e à linguagem UML.
Nome
Qualidade de Software
Objetivos
Discutir os principais problemas que desencadearam a busca por qualidade, tanto dos produtos de SW, como no processo; Apresentar o que é exigido de um produto de SW de qualidade; Apresentar os principais modelos de qualidade para a produção de software; Detalhar a norma ISO 9000-3, o CMM e o CMMI.
Resumo
Foi mostrada toda a evolução de um processo de qualidade e toda a política de custos que envolvem este processo. Foram detalhadas as características de um produto de SW de qualidade como funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade. Foi mostrada a evolução do processo de criação de normas e certificação para os produtos e empresas de SW. Foram apresentadas a documentação e as normas exigidas pelos atuais modelos de qualidade. Também foram feitas comparações entra cada um dos modelos de qualidade apresentados.
Nome
Planejamento e Gerenciamento de Projetos
Objetivos
Discutir uma metodologia de desenvolvimento iterativa e incremental; Apresentar as atividades de planejamento de projetos de software na óptica de um processo iterativo e incremental; Elucidar as dúvidas mais comuns relacionadas ao planejamento de projetos iterativos e incrementais;
Resumo

Apresentou as formas de planejamento e gerenciamento de um projeto a partir da simulação de um projeto. Para o projeto, foram realizadas as seguintes tarefas: análise dos requisitos, análise dos riscos e prioridades, seleção de profissionais necessários ao desenvolvimento do projeto, definição dos treinamentos necessários, cálculo dos custos do projeto.
Nome
Análise e Projeto de Software com UML
Objetivos
Apresentar os conceitos básicos do fluxo de análise e projeto; Descrever e exercitar técnicas para modelar um sistema orientado a objetos; Conhecer os principais diagramas de UML; Apresentar o padrão de arquitetura em camadas da Qualiti-CESAR.
Resumo
Foi feita uma introdução a Orientação a Objetos com UML apresentando. Foram apresentadas atividades, artefatos e responsáveis pelo fluxo de Análise e Projeto. Foi realizado um estudo detalhado sobre casos de uso, incluindo análise e construção de diagramas de casos de uso, de classes, de seqüência e de colaboração. A ferramenta utilizada para a construção desses diagramas foi o Rose Real Time da Rational. Foi realizado um estudo sobre projeto de arquitetura ressaltando os conceitos de herança, cápsulas e concorrência, protocolos, subsistemas e fachadas. Foram analisadas oportunidades de reuso, a estruturação em camadas e o agrupamento das classes em pacotes.
Nome
Orientação a Objetos com Java
Objetivos
Obter uma visão geral dos aspectos básicos de implementação: conceitos básicos, responsáveis e documentos, fluxo de atividades e guias; Implementar as classes do modelo de projeto em termos de componentes (código fonte ou executável, arquivo de documentação, etc); Definir a organização do código utilizando subsistemas de implementação; Testar os componentes como unidades; Integrar os componentes desenvolvidos em uma versão executável do sistema.
Resumo
Inicialmente, foi mostrada um pouco da história sobre Java e conceitos de Orientação a Objetos, destacando suas vantagens. Em seguida, foi apresentado o JSDK (Java Software Development Kit) conjunto de ferramentas, bibliotecas e exemplos para desenvolvimento em Java. Foi ensinado ainda como compilar e executar arquivos Java. Foram abordados a sintaxe básica Java, suas classes, objetos, atributos e métodos, referências, strings, arrays, conceitos de herança e polimorfismo, classes abstratas e interfaces, exceções, pacotes e componentes AWT.
Nome
Testes de Software
Objetivos

Apresentar e discutir os conceitos básicos sobre o processo de testes; Entender como criar e utilizar os artefatos gerados ao longo deste processo; Exercitar os conceitos apresentados; Apresentar os diferentes tipos de ferramentas que podem ser utilizadas no processo de testes.
Resumo
Foram apresentadas as etapas do processo de teste: planejamento, projeto, implementação, execução e avaliação dos testes. Em seguida foram analisados os dois tipos de abordagem de testes, a abordagem funcional ("caixa preta") e a estrutural ("caixa branca"). Foram estudados os estágios de testes: testes de unidade, de integração, de sistema e de aceitação (homologação).
Nome
Gerência de Configuração
Objetivos
Fornecer os principais conceitos relacionados à gerência de configuração; Criar uma visão geral de como a gerência de configuração pode ser aplicada ao seu projeto.
Resumo
Foram apresentados conceitos básicos de gerência de configuração, como configuração, baseline, repositório, lock, check-out, check-in, tags, branch, merge, build e release. Foram mostrados alguns padrões usados em gerência de configuração e o CVS (Concurret Versions System). Foi também mostrado o Ant, um software para a construção automática de builds, e o Bugzilla, uma ferramenta de controle de defeitos e melhorias em sistemas e que pode ser usada para gerência de mudanças. O curso foi finalizado com uma discussão sobre o planejamento de configurações de mudanças, os passos que são recomendados, o ambiente que tem que ser organizado, as atividades e responsáveis, além das ferramentas mais utilizadas.
Nome
Arquitetura de Software para Celulares
Objetivos
Apresentar uma introdução aos sistemas embarcados; Apresentar arquiteturas de software para celulares; Estudar a arquitetura Platform 2000 da Motorola; Estudar o Gerenciamento de memória limitada.
Resumo
Foi estudado sistemas embarcados, destacando áreas de aplicação, características, arquitetura básica, mercado. Foi estudado também projeto e arquitetura desses sistemas. Foi estudada a plataforma de desenvolvimento de softwares para celulares Motorola, a Platform 2000 (P2K).

4. Agenda Cultural

Ao final do curso de J2ME (Java to Micro Edition) foi proposto o desenvolvimento de uma aplicação para celular utilizando a linguagem J2ME. A minha equipe desenvolveu um projeto entitulado "Agenda Cultural". A descrição e especificação da aplicaçõa Agenda Cultural será apresentada a seguir. O programa está disponível no Apêndice.

4.1. Descrição

A Agenda Cultural é uma aplicação que fornece informações, via celular, sobre a programação cultural semanal de uma determinada cidade. A busca por informações pode ser obtida por categoria de eventos. As categorias são: Bar, Cinema, Teatro e Show. Para cada tipo de evento, a aplicação também permite ao usuário efetuar a busca de acordo com alguns critérios de seleção ou efetuar uma busca geral.

Para a categoria Bar, existem as opções de busca: Pub, Boteco e Todos. Para a categoria Cinema, as opções são: Aventura, Comédia Suspense, terror e Todos. Para a categoria Teatro, as opções são: Adulto Infantil e Musical. Para a categoria Show, as opções são: Forró, MPB, Rock, Samba e Todos.

Selecionando uma das opções de cada categoria, a aplicação retorna uma lista de eventos. Para cada evento selecionado são fornecidas informações associadas ao evento, como, por exemplo, Local, Data, Horário, Preço, etc.

As informações são obtidas de um servidor. Para que as informações possam ser divulgadas, os estabelecimentos precisam cadastrar seus eventos neste servidor.

Além de poder acessar as informações, o usuário também pode efetuar a compra de ingressos, no caso de teatros e shows, e reservar mesas, no caso de bares.

A compra de ingressos é feita através de cartão de crédito. O usuário envia o número do cartão. Como comprovante, ele recebe do servidor uma mensagem no celular que deve ser apresentada na entrada do evento. No caso de reserva de mesas, o usuário solicita a reserva e recebe uma mensagem que também deve ser mostrada na entrada do estabelecimento.

4.2. Especificação

1. As informações da Agenda Cultural são transmitidas via rede.

2. Acessando a aplicação Agenda Cultural, irá aparecer uma tela transiente com a mensagem "Bem Vindo a Agenda Cultural". A tela transiente é mostrada durante 1 segundo.

3. Após a tela transiente, uma segunda tela é mostrada com as quatro categorias de eventos disponíveis: Bar, Cinema, Teatro e Show. O título desta tela é "Eventos Culturais". As *softkeys* da esquerda e da direita são, respectivamente, "Sair" e "Selecionar".

4. É possível navegar sobre as opções da tela "Eventos Culturais" através das *arrow keys*.

5. Pressionando a softkey "Sair" o celular sai da aplicação.

6. Destacando a opção "Bar" e pressionando a *softkey* "Selecionar", a tela "Bar" é acessada. As opções disponíveis nesta tela são: Pub, Boteco e Todos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

7. É possível navegar sobre as opções da tela "Bar" através das *arrow keys*.

8. Destacando uma das opções de "Bar" e pressionando a *softkey* "Selecionar", a tela "Lista de Eventos" é acessada. As opções de eventos mostradas na tela estão de acordo com a opção escolhida referem-se aos nomes dos eventos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

9. É possível navegar sobre as opções da tela "Lista de Eventos" através das *arrow keys*.

10. Destacando uma das opções de "Lista de Eventos" e pressionando a *softkey* "Selecionar", a tela "Detalhes" é acessada. Nesta tela são mostradas informações associadas ao evento como Nome, Local, Data e Hora e o campo para digitar o número do cartão de crédito. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Comprar".

11. Pressionando a softkey "Voltar" o celular retorna a tela anterior.

12. Destacando a opção "Cinema" e pressionando a *softkey* "Selecionar", a tela "Cinema" é acessada. As opções disponíveis nesta tela são: Aventura, Comédia Suspense, terror e Todos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

13. É possível navegar sobre as opções da tela "Cinema" através das *arrow keys*.

14. Destacando uma das opções de "Cinema" e pressionando a *softkey* "Selecionar", a tela "Lista de Eventos" é acessada. As opções de eventos mostradas na tela estão de acordo com a opção escolhida referem-se aos nomes dos eventos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

15. É possível navegar sobre as opções da tela "Lista de Eventos" através das *arrow keys*.

16. Destacando uma das opções de "Lista de Eventos" e pressionando a *softkey* "Selecionar", a tela "Detalhes" é acessada. Nesta tela são mostradas informações associadas ao evento como Nome, Local, Data e Hora e o campo para digitar o número do cartão de crédito. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Comprar".

17. Pressionando a softkey "Voltar" o celular retorna a tela anterior.

18. Destacando a opção "Teatro" e pressionando a *softkey* "Selecionar", a tela "Teatro" é acessada. As opções disponíveis nesta tela são: Adulto Infantil e Musical. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

19. É possível navegar sobre as opções da tela "Teatro" através das *arrow keys*.

20. Destacando uma das opções de "Teatro" e pressionando a *softkey* "Selecionar", a tela "Lista de Eventos" é acessada. As opções de eventos mostradas na tela estão de acordo com a opção escolhida referem-se aos nomes dos eventos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

21. É possível navegar sobre as opções da tela "Lista de Eventos" através das *arrow keys*.

22. Destacando uma das opções de "Lista de Eventos" e pressionando a *softkey* "Selecionar", a tela "Detalhes" é acessada. Nesta tela são mostradas informações associadas ao evento como

Nome, Local, Data e Hora e o campo para digitar o número do cartão de crédito. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Comprar".

23. Pressionando a softkey "Voltar" o celular retorna a tela anterior.

24. Destacando a opção "Shows" e pressionando a *softkey* "Selecionar", a tela "Shows" é acessada. As opções disponíveis nesta tela são: Forró, MPB, Rock, Samba e Todos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

25. É possível navegar sobre as opções da tela "Shows" através das *arrow keys*.

26. Destacando uma das opções de "Shows" e pressionando a *softkey* "Selecionar", a tela "Lista de Eventos" é acessada. As opções de eventos mostradas na tela estão de acordo com a opção escolhida e referem-se aos nomes dos eventos. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Selecionar".

27. É possível navegar sobre as opções da tela "Lista de Eventos" através das *arrow keys*.

28. Destacando uma das opções de "Lista de Eventos" e pressionando a *softkey* "Selecionar", a tela "Detalhes" é acessada. Nesta tela são mostradas informações associadas ao evento como Nome, Local, Data e Hora e o campo para digitar o número do cartão de crédito. As *softkeys* da esquerda e da direita são, respectivamente, "Voltar" e "Comprar".

29. Pressionando a softkey "Voltar" o celular retorna a tela anterior.

30. O formato pra informação de hora é hh:mm.

31. O formato pra informação de Data é dd/mm/yy.

32. Pressionando a softkey "Comprar", a compra é efetuada e o servidor envia para o celular uma mensagem com o numero do ingresso.

4.3. Casos de Uso

Será apresentada a descrição de três casos de uso de prioridades essenciais.

O primeiro caso de uso descreve a tentativa do usuário consultar as opções de um determinado evento. No caso, o evento escolhido é do tipo "Show". O comportamento para eventos dos tipos Bar, Cinema e Teatro é semelhante.

O segundo caso de uso descreve a tentativa do usuário consultar um evento. No caso, a categoria escolhida para ser descrita no caso de uso é "Show" e a busca será pela opção "Samba". O comportamento para as opções Forró, MPB, Samba e Todos é semelhante, descrever os casos de uso para todas as opções seria, portanto, uma tarefa redundante. Pela mesma razão, não serão descritos os casos de uso que descrevem a tentativa do usuário consultar os eventos Bar, Cinema e Teatro.

O terceiro caso de uso descreve a tentativa do usuário comprar um ingresso. No caso, a categoria escolhida para ser descrita no caso de uso é "Show" e a busca será pela opção "Samba" e o evento escolhido é o show de "Alcione".

Caso de Uso 1: Consultar os tipos de Show

Descrição: Realiza consulta aos tipos de Show.

Prioridade: Essencial.

Condição Inicial: Informar o tipo de evento (Show).

Condição Final: Retornar uma lista de tipos de Show.

Fluxo Principal:

1. usuário inicia a aplicação;
2. usuário informa a categoria do evento (Show);
3. usuário recebe uma lista com tipos de Show.

Caso de Uso 2: Consultar a lista de Shows do tipo Samba

Descrição: Realiza consulta a lista de Shows do tipo Samba.

Prioridade: Essencial.

Condição Inicial: Informar o tipo de evento.

Condição Final: Retornar uma lista de eventos.

Fluxo Principal:

1. usuário inicia a aplicação;

2. usuário informa a categoria do evento (Show);
3. usuário informa o tipo de busca (Samba);
4. usuário recebe uma lista com os shows
5. usuário escolhe um evento;
6. usuário recebe informações associadas ao evento.

Fluxo Secundário: No passo 5, caso não haja evento, o usuário recebe uma mensagem informando que não há evento.

Caso de Uso 3: Comprar ingresso para um dos Shows do tipo Samba

Descrição: Realiza a comprar ingresso para um dos Shows do tipo Samba.

Prioridade: Essencial.

Condição Inicial: Informar o tipo de evento.

Condição Final: Retornar uma lista de eventos.

Fluxo Principal:

1. usuário inicia a aplicação;
2. usuário informa a categoria do evento (Show);
3. usuário informa o tipo de busca (Samba);
4. usuário recebe uma lista com os shows;
5. usuário escolhe um evento;
6. usuário recebe informações associadas ao evento;
7. usuário digita o número do cartão de crédito;
8. usuário recebe uma mensagem de confirmação da compra.

Fluxo Secundário: No passo 5, caso não haja evento, o usuário recebe uma mensagem informando que não há evento.

5. Conclusão

A atividade de teste é uma das principais atividades de garantia de qualidade de um produto. Trata-se, portanto, de uma atividade essencial no processo de desenvolvimento de qualquer produto, seja produto de software ou não.

Apesar da grande importância, a atividade de testes de software é recente, carente de profissionais especializados e necessita de muita pesquisa. Por essa razão, a Motorola aliada ao CIn, agregou ao estágio uma especialização na área de testes de software o que tornou o estágio ainda mais enriquecedor, por juntar o conhecimento científico por parte do Cin ao conhecimento técnico da Motorola.

Além do aprendizado obtido nas atividades de teste, também foi bastante enriquecedor o participar de toda a estrutura organizacional da Motorola, que se destaca através de certificação CMM nível 2, pela sua estrutura organizacional. Aliando ainda o fato de ser uma empresa de grande porte, multinacional e uma das líderes em seu segmento.

6. Referências

DEITEL, 2001 H.M; DEITEL, P.J; Java, Como Programar, Bookman Companhia Editora.

INTHURN, 2001 Cândida; Qualidade & Teste de Software, Visual Books Ltda.

Apostila "Introdução a Testes de Software";

Apostila "Análise e Projeto de Software com UML", volumes 1 and 2.

http://java.sun.com/j2se/1.3/docs/api/ on 2004-08-10

7. Apêndice – Programa

AgendaMidlet

```
package agenda;

import javax.microedition.lcdui.Alert;
import javax.microedition.lcdui.AlertType;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.CommandListener;
import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.Displayable;
import javax.microedition.midlet.MIDlet;
import javax.microedition.midlet.MIDletStateChangeException;
import Interfaces.*;
import Servlet.*;

public class AgendaMidlet extends MIDlet implements CommandListener{

     private Display display;
     private Alert alerta;
     private Eventos eventOp;
     private Filme filme;
     private Show show;
     private Teatro teatro;
     private ConexaoServlet server;
     private Bar bar;
     public ListEventos list;
     private EventoDetalhes detalhe;
     private int ind=0;
     private ConexaoOpCartao cartao;

      public AgendaMidlet() {
          alerta     = new Alert("Agenda Cultural", "Bem Vindo a Agenda
Cultural", null, AlertType.INFO);
           alerta.setTimeout(1500);

           eventOp  = new Eventos(this);
           eventOp.setCommandListener(this);
```

```
                filme    = new Filme(this);
                filme.setCommandListener(this);

                show     = new Show(this);
                show.setCommandListener(this);

                teatro   = new Teatro(this);
                teatro.setCommandListener(this);

                bar      = new Bar(this);
                bar.setCommandListener(this);

                list = new ListEventos();
                list.setCommandListener(this);

                display  = Display.getDisplay(this);
        }

        protected void startApp() throws MIDletStateChangeException {
                display.setCurrent(alerta,eventOp);

        }

        protected void pauseApp() {
        }

        protected      void      destroyApp(boolean      arg0)      throws
MIDletStateChangeException {

        private void setTelaAnt(int ind){
                switch(ind){
                        case 0: display.setCurrent(bar);
                                break;
                        case 1: display.setCurrent(filme);
                                break;
                        case 2: display.setCurrent(teatro);
                                break;
                        case 3: display.setCurrent(show);
                                break;
                }
```

```
        }

        public void commandAction(Command c, Displayable a) {

              int opcao=0;

              if(a == eventOp){
                    if(c == eventOp.getOkCommand()){
                          ind = eventOp.getSelectedIndex();
                          //Abre a tela relacionada ao evento selecionado
(filme, bar, teatro,...)
                          setTelaAnt(ind);
                    }
                    else if(c == eventOp.getSairCommand()){
                          try{
                                destroyApp(false);
                            notifyDestroyed();
                           }catch (MIDletStateChangeException sce) {
                            System.out.println(sce.getMessage());
                           }
                    }
              }

              //Opções caso esteja na tela de filmes
              if(a == filme){
                    if(c == filme.getBuscarCommand()){
                          String tipo=null;

                          switch(filme.getSelectedIndex()){
                                case 0: tipo="Aventura";
                                      break;
                                case 1: tipo="Comedia";
                                      break;
                                case 2: tipo="Suspense";
                                      break;
                                case 3: tipo="Terror";
                                      break;
                                case 4: tipo="Todos";
                                            break;
                          }
                          server = new ConexaoServlet(this,"FILME",tipo);
```

```
                Thread t = new Thread(server);
                t.start();
                display.setCurrent(server);
                System.out.println("Ind="+ind);

        }
        else if(c == filme.getVoltarCommand()){
                display.setCurrent(eventOp);
        }
        ind=1;
}

//Opções caso esteja na tela de teatro
if(a == teatro){
        if(c == teatro.getBuscarCommand()){
                String tipo=null;

                switch(teatro.getSelectedIndex()){
                        case 0: tipo="Adulto";
                                break;
                        case 1: tipo="Infantil";
                                break;
                        case 2: tipo="Musical";
                                break;
                        case 3: tipo="Todos";
                                        break;
                }
                server = new ConexaoServlet(this,"TEATRO",tipo);
                Thread t= new Thread(server);
                t.start();
                display.setCurrent(server);

        }
        else if(c == teatro.getVoltarCommand()){
                display.setCurrent(eventOp);
        }
}

//Opções caso esteja na tela de show
if(a == show){
        if(c == show.getBuscarCommand()){
```

```
                String tipo=null;

                switch(show.getSelectedIndex()){
                      case 0: tipo="Forro";
                            break;
                      case 1: tipo="MPB";
                            break;
                      case 2: tipo="Rock";
                            break;
                      case 3: tipo="Samba";
                            break;
                      case 4: tipo="Todos";
                            break;
                }
                server = new ConexaoServlet(this,"SHOW",tipo);
                Thread t= new Thread(server);
                t.start();
                display.setCurrent(server);

          }
          else if(c == show.getVoltarCommand()){
                display.setCurrent(eventOp);
          }
    }

    //Opcoes da tela de bar
    if(a == bar){
          if(c == bar.getBuscarCommand()){
                String tipo=null;
                switch(bar.getSelectedIndex()){
                      case 0: tipo="Pub";
                            break;
                      case 1: tipo="Buteco";
                            break;
                      case 2: tipo="Todos";
                            break;
                }
                server = new ConexaoServlet(this,"BAR",tipo);
                Thread t= new Thread(server);
                t.start();
                display.setCurrent(server);
```

```
                }
                else if(c == bar.getVoltarCommand()){
                        display.setCurrent(eventOp);
                }
        }


        //Lista com o nome dos eventos retornados do servidor
        if(a == server){
                opcao = server.getSelectedIndex();
                if(c == server.getSelecionar()){
                  detalhe = new EventoDetalhes(this);
                  detalhe.setCommandListener(this);
                 detalhe.append("Nome: " + server.nome[opcao]+ "\n");
                  detalhe.append("Local: "+ server.local[opcao]+ "\n");
                  detalhe.append("Data: " + server.data[opcao]+ "\n");
                  detalhe.append("Hora: " + server.hora[opcao]+ "\n");
                  detalhe.append("Preço: " + server.preco[opcao]);
                  display.setCurrent(detalhe);
                }else if(c == server.getVoltar()){
                        setTelaAnt(ind);
                }
        }

        //Tela de detalhes dos eventos
        if(a == detalhe){
                if(c == detalhe.getComprar()){
                        cartao=          new          ConexaoOpCartao(this,
server.nome[opcao], detalhe.numCard.getString());
                        Thread t1 = new Thread(cartao);
                        t1.start();
                        display.setCurrent(cartao);
                }else if(c == detalhe.getVoltar()){
                        display.setCurrent(server);
                }
        }

        //tela de confirmacao de compra
        if(a == cartao){
                if(c == cartao.getOk()){
```

```
                    display.setCurrent(eventOp);
                }
            }

        } //fim commandAction

    }
```

ListEventos (Seleciona os tipos de eventos)

```
package agenda;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDletStateChangeException;

public class ListEventos extends List {

	private Command voltar;
	private Command selecionar;

	public ListEventos() {
		super("Lista de Eventos",List.IMPLICIT);

		selecionar = new Command("SELEC",  Command.OK, 1);
		voltar     = new Command("VOLTAR", Command.BACK, 1);
		this.addCommand(selecionar);
		this.addCommand(voltar);
	}

	protected Command getSelecionar(){
		return selecionar;
	}

	protected Command getVoltar(){
		return voltar;
	}

	protected void startApp() throws MIDletStateChangeException {
	}

	protected void pauseApp() {
	}

	protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
	}
```

```
}
```

Eventos (Lista dos tipos de eventos)

```
package Interfaces;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDletStateChangeException;
import agenda.AgendaMidlet;

public class Eventos extends List {

        private static String [] vEventos={"Bar", "Cinema", "Teatro",
"Show"};
        private Command ok, sair;
        Image[] imagem=null;

        public Eventos(AgendaMidlet x){

              super("Eventos Culturais", List.IMPLICIT);
              ok     = new Command("OK", Command.OK, 1);
              sair = new Command("SAIR", Command.EXIT, 1);

              this.addCommand(ok);
              this.addCommand(sair);
              this.setCommandListener(x);

              try{
                    Image imagem = Image.createImage("/App.png");
                    for(int i=0; i<vEventos.length; i++){
                          this.append(vEventos[i],imagem);
                    }
              }catch(Exception ioe){}

        }

        protected void pauseApp() {
        }
```

```
        protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
        }

        public Command getOkCommand(){
              return ok;
        }

        public Command getSairCommand(){
              return sair;
        }
  }
```

Bar

```
  package Interfaces;

  import javax.microedition.lcdui.Command;
  import javax.microedition.lcdui.Image;
  import javax.microedition.lcdui.List;
  import javax.microedition.midlet.MIDletStateChangeException;
  import agenda.*;

  public class Bar extends List {

        private static String []tipos = {"Pub", "Buteco", "Todos"};
        private Command voltar, buscar;

        public Bar(AgendaMidlet x) {
              super("Bares",List.IMPLICIT);
              buscar = new Command("BUSCAR", Command.OK, 1);
              voltar = new Command("VOLTAR", Command.BACK, 1);

              this.addCommand(buscar);
              this.addCommand(voltar);
              this.setCommandListener(x);

              try{
                    Image imagem = Image.createImage("/App.png");
```

```
                        for(int i=0; i<tipos.length; i++){
                              this.append(tipos[i],imagem);
                        }
                  }catch(Exception ioe){}
            }

            protected void startApp() throws MIDletStateChangeException {
            }

            protected void pauseApp() {
            }

            protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
            }

            public Command getBuscarCommand(){
                  return buscar;
            }

            public Command getVoltarCommand(){
                  return voltar;
            }

      }
```

Filme

```
ackage Interfaces;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDletStateChangeException;
import agenda.AgendaMidlet;

public class Filme extends List {

        private static String[] tipos = {"Aventura", "Comédia",
                                                "Suspense","Terror",
"Todos"};
        private Command voltar, buscar;

        public Filme(AgendaMidlet x) {
              super("Filmes",List.IMPLICIT);
              buscar = new Command("BUSCAR", Command.OK, 1);
              voltar = new Command("VOLTAR", Command.BACK, 1);

              try{
                    Image imagem = Image.createImage("/App.png");
                    for(int i=0; i<tipos.length; i++){
                          this.append(tipos[i],imagem);
                    }
              }catch(Exception ioe){}



              this.addCommand(buscar);
              this.addCommand(voltar);
              this.setCommandListener(x);

        }

        protected void startApp() throws MIDletStateChangeException {
        }
```

```
            protected void pauseApp() {
            }

            protected      void      destroyApp(boolean      arg0)      throws
MIDletStateChangeException {
            }

            public Command getBuscarCommand(){
                  return buscar;
            }

            public Command getVoltarCommand(){
                  return voltar;
            }

      }
```

Show

```
      package Interfaces;

      import javax.microedition.lcdui.Command;
      import javax.microedition.lcdui.Image;
      import javax.microedition.lcdui.List;
      import javax.microedition.midlet.MIDletStateChangeException;
      import agenda.AgendaMidlet;


      public class Show extends List {

            private static String []tipos = {"Forro", "MPB", "Rock", "Samba",
                                                "Todos"};
            private Command voltar, buscar;

            public Show(AgendaMidlet x) {
                  super("Shows",List.IMPLICIT);
                  buscar = new Command("BUSCAR", Command.OK, 1);
                  voltar = new Command("VOLTAR", Command.BACK, 1);

                  this.addCommand(buscar);
```

```
                this.addCommand(voltar);
                this.setCommandListener(x);

                try{
                      Image imagem = Image.createImage("/App.png");
                      for(int i=0; i<tipos.length; i++){
                            this.append(tipos[i],imagem);
                      }
                }catch(Exception ioe){}
          }

          protected void startApp() throws MIDletStateChangeException {
          }

          protected void pauseApp() {
          }

          protected      void      destroyApp(boolean      arg0)      throws
MIDletStateChangeException {
          }

          public Command getBuscarCommand(){
                return buscar;
          }

          public Command getVoltarCommand(){
                return voltar;
          }

    }
```

Teatro

```
package Interfaces;

import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.Image;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDletStateChangeException;
import agenda.AgendaMidlet;

public class Teatro extends List {

        private static String []tipos = {"Adulto", "Infantil", "Musical",
"Todos"};
        private Command buscar, voltar;

        public Teatro(AgendaMidlet x) {
              super("Peças", List.IMPLICIT);
              buscar = new Command("BUSCAR", Command.OK, 1);
              voltar = new Command("VOLTAR", Command.BACK, 1);

              this.addCommand(buscar);
              this.addCommand(voltar);
              this.setCommandListener(x);

              try{
                    Image imagem = Image.createImage("/App.png");
                    for(int i=0; i<tipos.length; i++){
                          this.append(tipos[i],imagem);
                    }
              }catch(Exception ioe){}
        }

        protected void startApp() throws MIDletStateChangeException {
        }

        protected void pauseApp() {
        }
```

```
            protected       void       destroyApp(boolean       arg0)       throws
MIDletStateChangeException {
            }

            public Command getBuscarCommand(){
                  return buscar;
            }

            public Command getVoltarCommand(){
                  return voltar;
            }
      }
```

EventosDetalhes

```
      package Interfaces;

      import javax.microedition.lcdui.Command;
      import javax.microedition.lcdui.Form;
      import javax.microedition.lcdui.TextField;
      import javax.microedition.midlet.MIDletStateChangeException;

      import agenda.AgendaMidlet;


      public class EventoDetalhes extends Form {

            private Command voltar;
            private Command comprar;
            public TextField numCard;

            public EventoDetalhes(AgendaMidlet x) {
                  super("Detalhes");

                  numCard  =  new  TextField("Entre  número  do  cartão","",16,
TextField.NUMERIC);
                  comprar = new Command("COMPRAR", Command.OK, 1);
                  voltar  = new Command("VOLTAR", Command.BACK, 1);
                  this.append(numCard);
                  this.addCommand(comprar);
```

```
            this.addCommand(voltar);
            this.setCommandListener(x);
        }

        public Command getComprar(){
            return comprar;
        }

        public Command getVoltar(){
            return voltar;
        }

        protected void startApp() throws MIDletStateChangeException {
        }

        protected void pauseApp() {
        }

        protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
        }

    }
```

ArrayEventos

```
package agenda;

import javax.microedition.midlet.MIDletStateChangeException;


public class ArrayEventos {

	private Recorder[] rec;
	public static final int INDEX = 10;

	public ArrayEventos() {
		//super();
		rec = new Recorder[INDEX];
	}

	public Recorder getRecord (int i){
		return rec[i];
	}

	public void setRecord(Recorder r, int i){
		rec[i] = r;
	}

	public void addRecord(Recorder r){
		for (int i=0; i<INDEX;i++){
			if (rec[i] == null){
				rec[i] = r;
				break;
			}
		}
	}

	public void deleteRecorder(int ind){
		rec[ind] = null;
		for(int i=ind;i<(INDEX-1); i++){
			rec[i]=rec[i+1];
		}
	}
```

```
            protected void startApp() throws MIDletStateChangeException {
            }

            protected void pauseApp() {
            }

            protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
            }

    }
```

AgendaServlet

```
    package com.agenda;

    import java.io.*;
    import java.net.*;

    import javax.servlet.*;
    import javax.servlet.http.*;

    public class agendaServlet extends HttpServlet {

        /** Initializes the servlet.
         */
        public void init(ServletConfig config) throws ServletException {
            super.init(config);

        }

        /** Destroys the servlet.
         */
        public void destroy() {

        }

        /** Processes requests for both HTTP <code>GET</code> and
<code>POST</code> methods.
```

```
            * @param request servlet request
            * @param response servlet response
            */
           protected      void      processRequest(HttpServletRequest      request,
HttpServletResponse response)
           throws ServletException, IOException {
               response.setContentType("text/html");
               PrintWriter out = response.getWriter();

               String evento = request.getParameter("EVENTO");
               String tipo   = request.getParameter("TIPO");

               if(evento.equals("BAR")){
                   if(tipo.equals("Pub")){
                       out.write("NOME=Downtown&LOCAL=Recife
Antigo&DATA=10/08/2004&HORA=22:00&PRECO=15.00&"+

"NOME=Costureira&LOCAL=Derby&DATA=07/09/2004&HORA=20:00&PRECO=0.00&");
                   }else if(tipo.equals("Buteco")){

out.write("NOME=Buteco&LOCAL=Parnamirim&DATA=05/05/2004&HORA=22:00&PRECO=0.00&");
                   }else if(tipo.equals("Todos")){
                       out.write("NOME=Downtown&LOCAL=Recife
Antigo&DATA=10/08/2004&HORA=22:00&PRECO=15.00&"+

"NOME=Costureira&LOCAL=Derby&DATA=07/09/2004&HORA=20:00&PRECO=0.00&"+

"NOME=Buteco&LOCAL=Parnamirim&DATA=05/05/2004&HORA=22:00&PRECO=0.00&");
                   }
               }

               if(evento.equals("FILME")){
                   if(tipo.equals("Aventura")){
                       out.write("NOME=Indiana
Jones&LOCAL=Recife&DATA=02/09/2004&HORA=14:00&PRECO=14.00&"+
                                 "NOME=Planeta                                    dos
Macacos&LOCAL=Guararapes&DATA=02/09/2004&HORA=14:00&PRECO=10.00&");
                   }else if(tipo.equals("Comedia")){
                       out.write("NOME=Debi                                        e
Loide&LOCAL=Tacaruna&DATA=04/09/2004&HORA=16:00&PRECO=14.00&"+
```

```
                                "NOME=Amor                                  Segunda
Vista&LOCAL=Tacaruna&DATA=04/09/2004&HORA=14:00&PRECO=14.00&");
                }else if(tipo.equals("Suspense")){
                    out.write("NOME=Colecionaor                             de
Ossos&LOCAL=Recife&DATA=01/08/2004&HORA=18:00&PRECO=14.00&"+
                                "NOME=Efeito
Borboleta&LOCAL=Tacaruna&DATA=04/09/2004&HORA=20:00&PRECO=14.00&"+

"NOME=Copycat&LOCAL=Guararapes&DATA=04/09/2004&HORA=18:00&PRECO=10.00&");
                }else if(tipo.equals("Terror")){
                    out.write("NOME=&LOCAL=&DATA=&HORA=&PRECO=&");
                }else if(tipo.equals("Todos")){
                    out.write("NOME=Indiana
Jones&LOCAL=Recife&DATA=02/09/2004&HORA=14:00&PRECO=14.00&"+
                                "NOME=Planeta                               dos
Macacos&LOCAL=Guararapes&DATA=02/09/2004&HORA=14:00&PRECO=10.00&"+
                                "NOME=Debi                                  e
Loide&LOCAL=Tacaruna&DATA=04/09/2004&HORA=16:00&PRECO=14.00&"+
                                "NOME=Amor                                  Segunda
Vista&LOCAL=Tacaruna&DATA=04/09/2004&HORA=14:00&PRECO=14.00&"+
                                "NOME=Colecionaor                           de
Ossos&LOCAL=Recife&DATA=01/08/2004&HORA=18:00&PRECO=14.00&"+
                                "NOME=Efeito
Borboleta&LOCAL=Tacaruna&DATA=04/09/2004&HORA=20:00&PRECO=14.00&"+

"NOME=Copycat&LOCAL=Guararapes&DATA=04/09/2004&HORA=18:00&PRECO=10.00&");
                }
            }

             if(evento.equals("TEATRO")){
                if(tipo.equals("Adulto")){
                    out.write("NOME=Cinderela&LOCAL=Santa
Isabel&DATA=11/10/2004&HORA=24:00&PRECO=12.00&"+
                                "NOME=Dois                                  na
Gangorra&LOCAL=Artplex&DATA=04/09/2004&HORA=20:00&PRECO=10.00&");
                }else if(tipo.equals("Infantil")){
                    out.write("NOME=Alice&LOCAL=Teatro                      do
Parque&DATA=25/09/2004&HORA=10:00&PRECO=6.00&"+
                                "NOME=Tres
Porquinhos&LOCAL=Artplex&DATA=25/09/2004&HORA=14:00&PRECO=6.00&");
                }else if(tipo.equals("Musical")){
```

```
                    out.write("NOME=Chicago&LOCAL=Santa
Isabel&DATA=25/09/2004&HORA=10:00&PRECO=6.00&");
                }else if(tipo.equals("Todos")){
                    out.write("NOME=Cinderela&LOCAL=Santa
Isabel&DATA=11/10/2004&HORA=24:00&PRECO=12.00&"+
                                "NOME=Dois                                    na
Gangorra&LOCAL=Artplex&DATA=04/09/2004&HORA=20:00&PRECO=10.00&"+
                                "NOME=Alice&LOCAL=Teatro                      do
Parque&DATA=25/09/2004&HORA=10:00&PRECO=6.00&"+
                                "NOME=Tres
Porquinhos&LOCAL=Artplex&DATA=25/09/2004&HORA=14:00&PRECO=6.00&"+
                                "NOME=Chicago&LOCAL=Santa
Isabel&DATA=25/09/2004&HORA=10:00&PRECO=6.00&");
                }
            }

            if(evento.equals("SHOW")){
                if(tipo.equals("Forro")){
                    out.write("NOME=Limao          com          Mel&LOCAL=Clube
Portugues&DATA=20/09/2004&HORA=22:00&PRECO=10.00&"+
                                "NOME=Calcinha                Preta&LOCAL=Clube
Portugues&DATA=29/10/2004&HORA=22:00&PRECO=10.00&"+
                                "NOME=Magnificos&LOCAL=Classic
Hall&DATA=08/09/2004&HORA=22:00&PRECO=20.00&");
                }else if(tipo.equals("MPB")){
                    out.write("NOME=Djavan&LOCAL=Classic
Hall&DATA=01/09/2004&HORA=22:00&PRECO=30.00&"+
                                "NOME=Marina                  Lima&LOCAL=Teatro
UFPE&DATA=30/10/2004&HORA=20:00&PRECO=40.00&");
                }else if(tipo.equals("Rock")){
                    out.write("NOME=&LOCAL=&DATA=&HORA=&PRECO=&");
                }else if(tipo.equals("Samba")){
                    out.write("NOME=Alcione&LOCAL=Classic
Hall&DATA=20/09/2004&HORA=10:00&PRECO=15.00&"+
                                "NOME=Martinho       da       Vila&LOCAL=Armazem
14&DATA=29/10/2004&HORA=10:00&PRECO=15.00&");
                }else if(tipo.equals("Todos")){
                    out.write("NOME=Limao          com          Mel&LOCAL=Clube
Portugues&DATA=20/09/2004&HORA=22:00&PRECO=10.00&"+
                                "NOME=Calcinha                Preta&LOCAL=Clube
Portugues&DATA=29/10/2004&HORA=22:00&PRECO=10.00&"+
```

```
                                "NOME=Magnificos&LOCAL=Classic
Hall&DATA=08/09/2004&HORA=22:00&PRECO=20.00&"+
                                "NOME=Djavan\n&LOCAL=Classic
Hall&DATA=01/09/2004&HORA=22:00&PRECO=30.00&"+
                                "NOME=Marina                    Lima&LOCAL=Teatro
UFPE&DATA=30/10/2004&HORA=20:00&PRECO=40.00&"+
                                "NOME=Alcione&LOCAL=Classic
Hall&DATA=20/09/2004&HORA=10:00&PRECO=15.00&"+
                                "NOME=Martinho       da       Vila&LOCAL=Armazem
14&DATA=29/10/2004&HORA=10:00&PRECO=15.00&");
                    }
                }

                out.close();
            }

            /** Handles the HTTP <code>GET</code> method.
             * @param request servlet request
             * @param response servlet response
             */
            protected void doGet(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
                processRequest(request, response);
            }

            /** Handles the HTTP <code>POST</code> method.
             * @param request servlet request
             * @param response servlet response
             */
            protected void doPost(HttpServletRequest request, HttpServletResponse
response)
            throws ServletException, IOException {
                processRequest(request, response);
            }

            /** Returns a short description of the servlet.
             */
            public String getServletInfo() {
                return "Short description";
            }
```

```
}
```

ConexaoServlet

```
package Servlet;

import java.io.IOException;
import java.io.InputStream;
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import javax.microedition.lcdui.Command;
import javax.microedition.lcdui.List;
import javax.microedition.midlet.MIDletStateChangeException;
import agenda.AgendaMidlet;

public class ConexaoServlet extends List implements Runnable {

      private String baseURL= "http://localhost:8084/agenda/agendaServlet";
      String evento, tipo;
      private Command voltar;
      private Command selecionar;
      public String[] detalheEvento;
      public int ind=0;
      private static final int MAX=10;
      private String line=null;
      public   String[]   nome=null,   local=null,   data=null,   hora=null,
preco=null;

      public ConexaoServlet(AgendaMidlet x, String e, String t) {
            super("Lista",List.IMPLICIT);
            evento = e;
            tipo   = t;

            //detalheEvento = new String[MAX];
            nome  = new String[MAX];
            local = new String[MAX];
            data  = new String[MAX];
            hora  = new String[MAX];
            preco = new String[MAX];
```

```
                selecionar = new Command("SELEC",  Command.OK, 1);
                voltar     = new Command("VOLTAR", Command.BACK, 1);
                this.addCommand(selecionar);
                this.addCommand(voltar);
                this.setCommandListener(x);

        }

        protected void startApp() throws MIDletStateChangeException {
        }

        protected void pauseApp() {
        }

        protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
        }

        public Command getSelecionar(){
                return selecionar;
        }

        public Command getVoltar(){
                return voltar;
        }

        private void obtemSubStr(){
                String s, n;

                n  = line.substring(line.indexOf("=")+1, line.indexOf("&"));

                if(n.equals("")){
                        this.append("Evento não encontrado",null);

                }else{
                        //nome
                        nome[ind] = n;
                        line = line.substring(line.indexOf("&")+1,
line.length());
```

```
                        //local
                        local[ind]        =    line.substring(line.indexOf("=")+1,
line.indexOf("&"));
                        line        =        line.substring(line.indexOf("&")+1,
line.length());
                        //list.setlist(aux);

                        //data
                        data[ind]         =    line.substring(line.indexOf("=")+1,
line.indexOf("&"));
                        line        =        line.substring(line.indexOf("&")+1,
line.length());

                        //hora
                        hora[ind]         =    line.substring(line.indexOf("=")+1,
line.indexOf("&"));
                        line        =        line.substring(line.indexOf("&")+1,
line.length());

                        //preço
                        preco[ind]        =    line.substring(line.indexOf("=")+1,
line.indexOf("&"));
                        line        =        line.substring(line.indexOf("&")+1,
line.length());

                        this.append(nome[ind],null);
                        ind++;
                }



        }

        public void run() {
              HttpConnection connection = null;
              InputStream is = null;
              byte[] buffer  = null;

              try{
                    connection              =              (HttpConnection)
Connector.open(baseURL+"?EVENTO="+evento+ "&TIPO="+tipo+"&");
```

```
                int responseCode = connection.getResponseCode();

                if(responseCode != HttpConnection.HTTP_OK){
                      System.out.println("ERROR" + responseCode);
                }

                is = connection.openInputStream();
                int ch = 0, i = 0;

                buffer = new byte[5000];

                while ((ch=is.read()) != -1) {
                      buffer[i] = (byte)ch;
                      i++;
                }

                line = new String(buffer,0,i);

                while(line.indexOf("&")!= -1){
                      obtemSubStr();
                }

          }catch(IOException ioe){
                System.out.println(ioe);

          }

          try{
                connection.close();
          }catch(Exception e){}
     }

}
```

OpCartaoServlet

```
package com.agenda;

import java.io.*;
import java.net.*;
import java.util.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 *
 * @author  llaf
 * @version
 */
public class OpCartaoServlet extends HttpServlet {



    /** Initializes the servlet.
     */
    public void init(ServletConfig config) throws ServletException {
        super.init(config);

    }

    /** Destroys the servlet.
     */
    public void destroy() {

    }

    /**  Processes  requests  for  both  HTTP  <code>GET</code>  and
<code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected     void     processRequest(HttpServletRequest     request,
HttpServletResponse response)
```

```
        throws ServletException, IOException {

            int i;

            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            String evento = request.getParameter("Nome");
            String cpf    = request.getParameter("Cpf");

            Random gen = new Random();
            i = gen.nextInt(100);

            out.write("INGRESSO=" + i + "&");
            System.out.println(i);
            out.close();
        }

        /** Handles the HTTP <code>GET</code> method.
         * @param request servlet request
         * @param response servlet response
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
            processRequest(request, response);
        }

        /** Handles the HTTP <code>POST</code> method.
         * @param request servlet request
         * @param response servlet response
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse
response)
        throws ServletException, IOException {
            processRequest(request, response);
        }

        /** Returns a short description of the servlet.
         */
        public String getServletInfo() {
```

```
            return "Short description";
        }

    }
```

ConexaoOpCartao

```
    package Servlet;

    import java.io.IOException;
    import java.io.InputStream;

    import javax.microedition.io.Connector;
    import javax.microedition.io.HttpConnection;
    import javax.microedition.lcdui.Command;
    import javax.microedition.lcdui.Form;
    import javax.microedition.midlet.MIDletStateChangeException;

    import agenda.AgendaMidlet;


    public class ConexaoOpCartao extends Form implements Runnable  {

        private String baseURL =
"http://localhost:8084/agenda/OpCartaoServlet";
        private Command ok;
        private String line = null, nome, cpf;
        public String ingresso;

        public ConexaoOpCartao(AgendaMidlet x, String n, String cpf) {
            super("Ingresso");
            nome = n;
            this.cpf = cpf;
            ok = new Command("OK", Command.OK, 1);
            this.addCommand(ok);
            this.setCommandListener(x);

        }

        public Command getOk(){
```

```
            return ok;
        }

        protected void startApp() throws MIDletStateChangeException {
        }

        protected void pauseApp() {
        }

        protected void destroyApp(boolean arg0) throws
MIDletStateChangeException {
        }

        public void run() {
            HttpConnection connection = null;
            InputStream is = null;
            byte[] buffer  = null;

            try{
                connection = (HttpConnection)
Connector.open(baseURL+"?Nome="+nome+"&"
                        +"Cpf="+cpf+"&");

                int responseCode = connection.getResponseCode();

                if(responseCode != HttpConnection.HTTP_OK){
                    System.out.println("ERROR" + responseCode);
                }

                is = connection.openInputStream();
                int ch = 0, i = 0;

                buffer = new byte[50];

                while ((ch=is.read()) != -1) {
                    buffer[i] = (byte)ch;
                    i++;
                }

                line = new String(buffer,0,i);
```

```
                        ingresso        =    line.substring(line.indexOf("=")+1,
line.indexOf("&"));

                        if(ingresso.equals("")){
                              this.append("Operação  não  foi  realizada!\nTente
novamente.");
                        }else{
                              this.append("Nome: "+ nome + "\n");
                              this.append("Número Ingresso: "+ ingresso + "\n");
                        }
                  }catch(IOException ioe){
                        System.out.println(ioe);
                  }

                  try{
                        connection.close();
                  }catch(Exception e){}

            }

      }
```