PEDRO PAULO SIMÕES FREITAS

Orientador: Ricardo Rabelo

CLOUDGUIDE - UMA PLATAFORMA DE SUPORTE AO TURISTA EM OURO PRETO COM PONTOS DE ACESSO SEM FIO

Ouro Preto

Novembro de 2010

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

CLOUDGUIDE - UMA PLATAFORMA DE SUPORTE AO TURISTA EM OURO PRETO COM PONTOS DE ACESSO SEM FIO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

PEDRO PAULO SIMÕES FREITAS

Ouro Preto
Novembro de 2010

UNIVERSIDADE FEDERAL DE OURO PRETO

FOLHA DE APROVAÇÃO

CloudGuide - Uma plataforma de suporte ao turista em Ouro Preto com pontos de acesso sem fio

PEDRO PAULO SIMÕES FREITAS

Monografia defendida e aprovada pela banca examinadora constituída por:

Dr. Ricardo Rabelo – Orientador
Universidade Federal de Ouro Preto

Dr. Daniel Fernandes Macedo
Universidade Federal de Ouro Preto

Me. Elton José da Silva
Universidade Federal de Ouro Preto

Ouro Preto, Novembro de 2010

Resumo

Uma das características da computação ubíqua vem do aumento da presença de dispositivos portáteis devido aos avanços na fabricação de componentes eletrônicos. Esses dispositivos possuem uma considerável capacidade de processamento, com recursos para comunicação sem fio e armazenamento de dados. Os dispositivos vêm se popularizando como handhelds, PDA's, e atualmente têm aparecido smartphones e celulares de grande capacidade computacional.

O trabalho que segue tem por base o desenvolvimento de um novo Padrão de Projeto para dispositivos móveis, que utilizam o sistema operacional Android. Este padrão consiste em baixar uma imagem, junta-la a um texto e mostrar o conjunto na tela. Como o padrão utiliza a internet para fazer o download da imagem, são feito testes em três situações. Também são apresentados alguns trabalhos executados na parte de usabilidade, onde foram feitos estudos de interação, e propostas algumas interfaces.

Abstract

One of the characteristics of ubiquitous computing comes from the increased presence portable devices due to advances in component manufacturing electronics. These devices have a considerable processing capacity, with features for wireless communication and data storage. The devices are becoming popular as handhelds, PDAs, and have now appeared as cellular phones and large computational capacity.

The work that follows is based on the development of a new Design Pattern for mobile devices that uses the Android operating system. This pattern is to download an image, add it to a text and show on screen. As the standard uses the internet to download the image test is done in three situations. We presented some work performed on usability, where interaction studies were done , and proposed a number of interfaces.

Dedico este trabalho a meus pais Pedro e Pilar, aos meus irmãos Joaquim e Marcelo, e a minha namorada Aline.

Agradecimentos

Agradeço primeiro a Deus.

Aos meus pais, Pedro e Pilar, meus irmãos Joaquim e Marcelo, que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa da minha vida.

Ao professor e orientador Ricardo Rabelo por seu apoio para a execução e conclusão desta monografia.

Aos colegas da graduação, que juntos chegamos a essa nova etapa.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Atualmente, os aparelhos celulares possuem uma nova variedade de funcionalidades, como GPS, múltiplas interfaces de rede sem fio para acesso à internet, sensores de movimento, câmeras, etc. Com estes novos recursos, uma nova classe de aplicações vem sendo desenvolvidas, aproveitando ao máximo estas novas funcionalidades. Com esse aumento de funções deve-se ter uma preocupação maior com a usabilidade desses dispositivos, pois existem algumas limitações para a interface, como o tamanho da tela, tempo de baterias e limitação da configuração de hardware.

Em dispositivos móveis a interface tem um importante papel, pois é por ela que o usuário recebe e envia informações, e quanto mais fácil for a utilização, melhor será essa troca de informação. De acordo com alguns estudo feitos por Ribeiro (2007), as telas pequenas, apenas algumas linhas devem ser mostradas, pra facilitar a compreensão do usuário .

Além disso, vale ressaltar que a interação com o usuário possui outras características, como sensores de movimento e GPS, que permitem que o usuário possa interagir com o dispositivo utilizando outras formas de entrada de dados. Por exemplo, um acelerômetro pode informar para o dispositivo que o usuário está se movendo e algum aplicativo pode utilizar desta informação para melhorar a interação com o usuário. Muitos aparelhos já utilizam essa interface para possibilitar o rápido atendimento de uma chamada telefônica pela detecção do movimento do aparelho.

Essa rápida evolução da computação móvel também tem enorme potencial para fornecer informações multimídia dinâmicas a pessoas em movimento. Uma área de pesquisa que está interessada em explorar as formas em que os dispositivos móveis podem ser usados para fornecer serviços mais sofisticados é o da computação ciente de contexto. Aplicações cientes de contexto utilizam informações contextuais tais como localização e perfil do usuário, a fim de fornecer funcionalidade e informações adaptadas Cheverst et al. (2000).

Neste trabalho será apresentado um estudo de algumas estratégias para realizar a coleta dos dados do usuário de forma confortável, permitindo a atualização de seus dados quando necessário. Essas estratégias incluem formulação e disposição das perguntas, projeto de design

da interface, entre outros. A aplicação alvo é um aplicativo de suporte ao turista, chamado de CloudGuide.

Pensado neste mesmo contexto citado acima é proposto um novo Padrão de Projeto, que tem como objetivo auxiliar em aplicações para dispositivos móveis. Também são mostrados alguns testes, pois este padrão envolve o uso de internet para baixar imagens, com isso são examinadas mais de perto três soluções, e seus desempenhos.

Capítulo 2

Justificativa

Ouro Preto é uma cidade famosa por sua magnífica arquitetura colonial, primeira cidade brasileira a ser declarado Patrimônio Histórico e Cultural da Humanidade, pela UNESCO em 1980, e por seus principais pontos turísticos, igrejas, museus e eventos culturais. Com todas essas características Ouro Preto tornou-se uma referência para o turismo mundial, fazendo com que vários turistas de outros países, viessem conhecer um pouco melhor esta cidade.

A avaliação dos serviços e sites para o turismo e as negociações entre empresas e consumidores são baseados nos critérios de conteúdo, qualidade e design. Entretanto, quando os serviços envolvem dispositivos móveis, a avaliação depende principalmente de fatores como usabilidade, eficiência e pouco tempo para respostas. Mesmo com as limitações dos dispositivos móveis, a migração das tecnologias desses sites tem sua funcionalidade maximizada ao serem mesclados com aplicações embutidas nesses dispositivos.

Para oferecer recursos cientes de contexto é preciso coletar as informações do usuário, da aplicação e do dispositivo, fazendo com que esse usuário melhore a integração com a aplicação e a qualidade do acesso a essa aplicação de acordo com as capacidades do dispositivo. Porém, coletar informações sobre o usuário pode não ser uma tarefa agradável e simples, sendo preciso desenvolver algumas estratégias para coletar de forma mais cômoda para o usuário todas as suas informações que são necessárias para a aplicação.

Para atender bem aos usuários, uma aplicação sensível ao contexto deve se adaptar às informações que serão exibidas de acordo com o conjunto de circunstâncias coletadas do usuário, dispositivo, ambiente, etc. De todas essas circunstâncias, talvez a mais importante seja o interesse do turista. Por isso, a aplicação precisa de informações do usuário sempre completa e atualizada para gerenciar o perfil, permitindo que sejam apresentadas informações que realmente despertam o interesse desse usuário.

Capítulo 3

Trabalhos Relacionados

Os estudos de Huang et al. (2003) apresentam um sistema de gerenciamento de perfil, chamado Agent and Profile Management System (APMS), onde os usuários fazem o download e instalação de um aplicativo para seu dispositivo móvel. Esse aplicativo conecta com seu provedor correspondente, faz uma requisição, espera o processamento e recebe uma resposta no dispositivo. A maioria dos procedimentos acontece no lado do servidor. Contudo, o usuário precisa conhecer os serviços que estão disponíveis e saber exatamente qual deles quer usar.

Berkenbrock (2009) propões uma categorização de requisitos para avaliação da usabilidade de sistemas groupware móvel e define algumas métricas para avaliação da usabilidade. A usabilidade é um fator fundamental para tornar agradável ao usuário à coleta de informações sobre seu perfil. Com a análise desses requisitos no trabalho, concluiu-se que é preferível oferecer opções de múltipla escolha para as respostas, especialmente para usuários estrangeiros com pouca habilidade na escrita do idioma da aplicação. E como foi dito na seção anterior, a cidade de Ouro Preto atrai turistas de toda parte do mundo. Um motivo forte para que a aplicação seja acessível a pessoas de vários países.

Estudos de Berkenbrock (2009) mostram que devido às limitações no espaço da tela de dispositivos móveis, muitas aplicações usam a barra de rolagem como mecanismo para acessar informações que não cabem na tela. Entretanto, esse recurso pode ser irritante para os usuários Hirata (2008). Para evitar o uso de barras de rolagem, o conteúdo pode ser dividido em duas ou mais telas. Entretanto, evitando a barra de rolagem e dividindo em mais telas aumenta-se o número de passos de navegação, pois o usuário precisa selecionar mais opções para visualizar todo o conteúdo. Percebemos assim que é importante identificar os requisitos do usuário para definir qual é a melhor opção, obtendo um equilíbrio entre o uso da barra de rolagem e o número de passos de navegação para completar uma tarefa.

Capítulo 4

Plataforma Android e Aplicações Móveis

Para o desenvolvimento da aplicação, foi utilizado o sistema operacional para dispositivos móvel chamado Android. O Android é um sistema operacional criado pelo Google com o objetivo de difusão dos serviços online do Google, através do uso de acessos remotos a web services. A plataforma tem uma característica peculiar não ser dependente do hardware, possibilitando sua instalação em praticamente qualquer modelo de aparelho celular. Normalmente, os celulares possuem sistemas operacionais proprietários, cujo perfil e utilização são na maioria das vezes definidos pela operadora de telefonia.

A plataforma Android tem como característica uma alta taxa de resposta as requisições do usuário, além de fornecer uma API completa para a criação de aplicações cientes do contexto. Essa característica é refletida inclusive no estilo de programação dos dispositivos, uma vez que os desenvolvedores devem criar aplicativos que sejam responsivos à interação com o usuário, caso contrário, o sistema operacional finaliza a execução do aplicativo.

4.1 Interface e interação

Na plataforma Android, você define uma atividade de interface utilizando uma hierarquia de nós View e ViewGroup, como mostrado no diagrama da figura 4.1. Esta árvore hierarquiqua pode ser simples ou complexo como o desenvolvedor precisar, e também pode ser construída usando o conjunto de widgets do Android, layouts pré-definidos, ou podem haver construções da View.

As aplicações no Android são divididas em módulos chamados Atividades. Cada Atividade é na verdade uma classe, desenvolvida na linguagem Java, e pode ser associada com um componente chamado View. O View é a representação visual de uma Atividade na tela do aparelho. A metodologia de desenvolvimento do Android prevê a máxima reutilização das atividades, através do compartilhamento destas entre as aplicações que estão executando. O

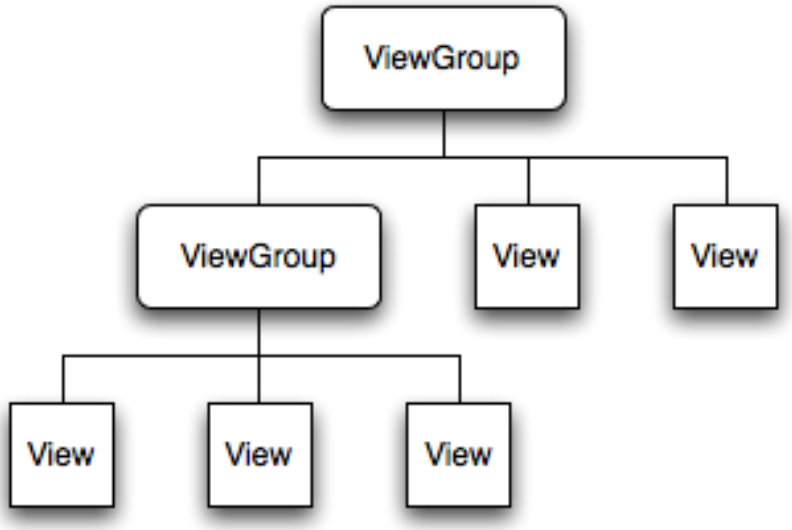

Figura 4.1: Árvore hierarquiqua

objetivo é diminuir o consumo de memória e bateria. Por exemplo, uma atividade que acessa a agenda de contatos pode ser compartilhada por quaisquer aplicações que a utilizem.

Para permitir esse compartilhamento, os componentes ficam em repositórios comuns e as aplicações devem gerenciar o armazenamento dos dados, controlando acessos e permissões de alterações. A configuração de uma interface composta pelas Views pode ser feita de duas maneiras:

- Declaração dos elementos de interface em XML. A vantagem da utilização do XML é que as descrições da interface do usuário são externas ao código do aplicativo, isso significa que é possível modificar a interface sem ter que modificar seu código-fonte. (listing 4.1);
- Instanciar elementos de layout em tempo de execução, utilizando a programação Java. As Atividades modificam a tela de acordo com o avanço do processamento. (listing 4.2).

O Android permite utilizar essas duas formas ao mesmo tempo. Um exemplo disso é a declaração dos layouts do aplicativo utilizando XML, incluindo os elementos da tela que irão aparecer e suas propriedades. É possível então adicionar código no aplicativo que iria alterar o estado dos Views que estão na tela, incluindo aqueles declarados em XML, em tempo de execução. Um botão na interface é um exemplo de utilização das duas maneiras: pode ser declarado no XML e sua ação acontece em tempo de execução.

A seguir um exemplo de código XML, que apresenta um texto (TextView) e um botão (Button). O próximo código é reponsavel por chamar o XML e mostra-lo na tela.

Uma aplicação de mensagens de texto, por exemplo, pode ter uma atividade que mostra uma lista de contatos para enviar mensagens, uma segunda atividade para escrever a men-

Listing 4.1: Interface em XML

Código XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content
            android:onClick="onBackClick
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Código JAVA

```
public class ImageListActivity extends ListActivity {
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
     }
      public void onBackClick(View v) {
          //ação do botão
     }
}
```

sagem para o contato escolhido, e outras atividades para ver as mensagens antigas ou alterar configurações. Embora essas atividades trabalhem juntas para formar uma interface coesa, cada atividade é independente das outras. Cada uma é implementada como uma subclasse da classe base de atividade Android (2010).

O Android implementa ainda o conceito de Intent (Intenção) para indicar que a aplicação tem a intenção de realizar uma tarefa qualquer. Cada ação representa uma Intent. Um bom exemplo é uma mudança de tela, onde a intenção é na verdade uma ação de finalizar uma atividade e iniciar outra.

Listing 4.2: Interface instanciando elementos em tempo de execução

Código XML

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content
            android:text="Hello, I am a Button" />
</LinearLayout>
```

Código JAVA

```
public class ImageListActivity extends ListActivity {
     @Override
     protected void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.main);
         // Elemento de id = button é instanciado
     final Button button = (Button) findViewById(R.id.button);
         button.setOnClickListener(new View.OnClickListener() {
               public void onClick(View v) {
                    //ação do botão
               }
        });
  }
}
```

Capítulo 5

Estudo de Interface

Para a aplicação foram feitos três estudos de interface. Esta seção apresentará cada um deles.

A partir de um questionário utilizado por pesquisadores do departamento de turismo (DETUR) da Universidade Federal de Ouro Preto, questionário que tem como objetivo avaliar os interesses do turista na cidade, foram desenvolvidas algumas estratégias de coleta de dados, com o objetivo de permitir uma coleta de dados mais personalizada e agradável ao usuário. A intenção é permitir que ao invés de um pesquisador entrevistar os turistas na rua, estes mesmos possam executar a aplicação de pesquisa em seus aparelhos e com isso, possam preencher os dados. Para uma coleta eficiente, vários estudos e estratégias de implementação da interface foram feitos.

5.1 Interface Simples

A primeira interface consistiu em reproduzir um questionário na tela, onde as questões da aplicação, sendo este estilo de desenvolvimento comum em aplicativos para desktops e mesmo em sites.

A interface de coleta dos dados do usuário foi projetada para somente uma tela com todas as demais perguntas, evitando que fossem necessários vários passos para a execução da tarefa. A maioria das opções de respostas foi de múltipla escolha, com exceção de algumas perguntas onde o turista deve informar o seu nome, CPF e nome de lugares que já tenha visitado. Como identificador único, o sistema utiliza a identificação do usuário do aparelho.

Essa interface é de fácil utilização para os turistas, pois expõe as perguntas de forma clara e fornece as opções de respostas na maioria das vezes, evitando que o usuário digite várias informações. Também possui um recurso de auto-completar que facilita quando o usuário esta digitando algo, na aplicação é utilizado para o campo cidade de origem (figura 5.2).

Um outro recurso de interface utilizado é chamado de Spinner, este recurso, assim que é acionado é gerado um menu de opções sobre a tela que está sendo utilizada e o usuário faz a escolha de uma opção (figura 5.3).

Figura 5.1: Interface da aplicação - Preferências do turista

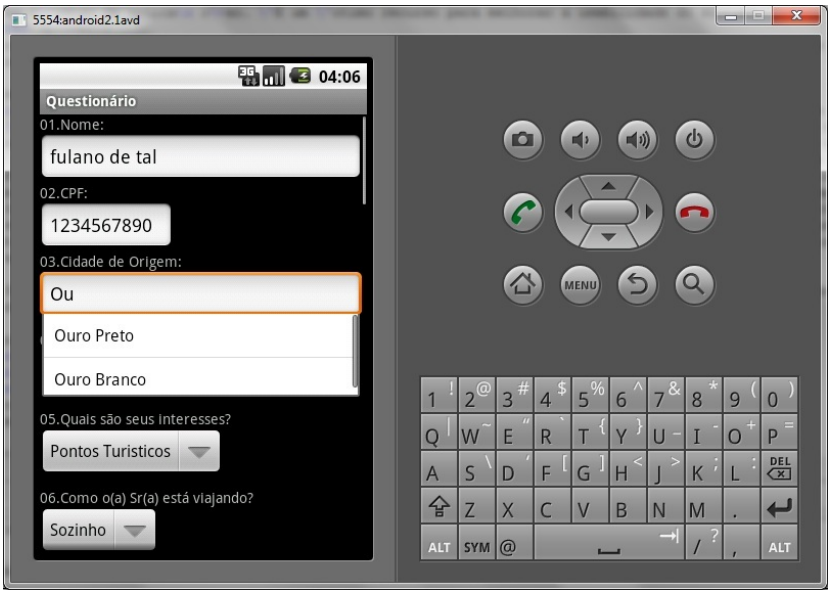

Figura 5.2: Interface da aplicação - Recurso auto completar

Quando o botão "Salvar", da figura 5.1, é acionado, a aplicação captura todos os dados informados pelo usuário e envia para um webservice através de uma conexão HTTP. Essa estratégia de persistência dos dados na web facilita a utilização das informações por aplicações sensíveis ao contexto, que podem acessá-las de qualquer local, a qualquer momento. Assim, serviços personalizados podem ser oferecidos aos turistas, com base nos dados do perfil que eles mesmos criam.

Figura 5.3: Interface da aplicação - Recurso spinner

5.2 Interface com Abas

A segunda interface agrupou por conteúdo as perguntas, criando uma interface com abas, como mostrado na figura 5.4. Este tipo de interface minimiza a rolagem da tela e estrutura funcionalmente as questões. Outra característica importante é o posicionamento das barras na parte superior, permitindo um controle mais claro do usuário a respeito da aba utilizada. Esta tela introduz mais conforto no preenchimento, pois o agrupamento permitiu um preenchimento parcial dos dados, assim como seu salvamento.

Para economizar o tráfego de dados externo, esta aplicação salva localmente os dados parciais, enviando para o webservice somente quando todas as informações estiverem preenchidas.

5.3 Interface Contextual

A terceira interface consistiu em agrupar de maneira mais confortável os questionários eliminando ao máximo informações textuais. Para isso, foi criada uma tela de entrada que permitisse o agrupamento dos itens do questionário, que se encontram sob um mesmo contexto.

A Figura 5.5 apresenta esta interface, na qual a informação do turista é agrupada de acordo com as visitas, a museus, restaurantes e um mapa guia. Além disso, o perfil permite a coleta de informações pessoais.

A plataforma Android possui uma API para coleta de informações de GPS e rede sem fio, que permitem o uso de um sistema de localização para identificar os locais onde o usuário esteve. Essa informação pode ser associada ao webservice do Google Maps e permite que as aplicações tenham uma noção exata da localização do usuário.

Figura 5.4: Exemplo interface com abas

Com o uso dessa API, é possível detectar automaticamente quais os pontos turísticos que o turista se encontra e também o momento em que ele se movimenta em direção a eles. Essa informação é utilizada para preencher as informações a respeito da localização do turista e com isso, reduzir as informações que ele preenche a questões de opinião a respeito do local visitado. Com isso, as informações a respeito dos museus, restaurantes, etc, podem ser consideradas já preenchidas.

Figura 5.5: Exemplo de interface contextual

Capítulo 6

Padrões de Projetos

6.1 Descrição

A técnica de reutilização de código pode acelerar o processo desenvolvimento do software. Quando é encontrada uma boa solução para um determinado problema, os projetistas a reutilizam várias vezes. Assim essas soluções são chamadas de padrões, esses padrões resolvem problemas específicos de projetos e tornam os projetos orientados a objetos mais flexíveis, e às vezes reutilizáveis. Os projetistas que já adquiriram um conhecimento sobre esses padrões podem utilizá-los com mais facilidade em diferentes problemas de projeto.

(Gamma et al. (2000)) Os padrões de projeto torna mais fácil reutilizar projetos e arquiteturas bem-sucedidas. Expressar técnicas testadas e aprovadas as torna mais acessíveis para os desenvolvedores de novos sistemas. Os padrões de projeto ajudam a escolher alternativas de projetos que tornam um sistema reutilizável e a evitar alternativas que comprometam a reutilização. Os padrões de projeto podem melhorar a documentação e a manutenção de sistemas ao fornecer uma especificação explícita de interações de classes e objetos e o seu objetivo subjacente. Em suma, ajudam um projetista a obter mais rapidamente um projeto adequado.

Os padrões de projetos existem para acelerar o desenvolvimento de uma aplicação, pois são soluções previamente prontas para os problemas mais comuns do cotidiano de um desenvolvedor. O importante dos padrões de projetos são as definições dos problemas e a melhor saída para ele. Através da análise de caso e de soluções de uma situação, e tendo um conhecimento dos padrões de projeto pode-se decidir qual usar, como usar e porque usar. (Zemel (2009)) Os padrões podem ser implementados em qualquer linguagem de programação. (Gamma et al. (2000)) Mas a utilização de padrões podem ser mais fácil em uma determinda linguagem que em outra.

Os padrões contêm quatro elementos essenciais:

- "nome do padrão", nomeia o padrão usando uma ou duas palavras que descrevem as características de um projeto;

- "problema", enumera algumas características para qual situação é melhor aplicar o padrão. Em algumas descrições do problema, é adicionada uma lista de condições que devem ser satisfeitas para que faça sentido utilizar o padrão;
- "solução", não é descrita uma implementação e sim os elementos que compõem o padrão de projeto, seus relacionamentos, suas responsabilidade e colaborações;
- "consequências", relata vantagens, desvantagens e resultados de uso do padrão. São críticas para a avaliação de alternativas de projetos e para a compreensão dos custos e benefícios na utilização do padrão.

Um exemplo particular de padrão de projeto é o MVC (Model-View-Controller), utilizado para construir interfaces com o usuário em Smalltalk. O MVC separa Visão e Modelo, a visão deve garantir que sua aparência reflita o estado do modelo, assim quando houver alguma mudança nos dados do modelo, este notifica as visões que dependem dele. Com a notificação as visões podem atualizar-se, esta abordagem permite ligar múltiplas visões a um modelo para oferecer diferentes apresentações, e também pode criar novas visões sem ter que refazer o modelo. Quando um modelo contiver alguns valores de dados, estes dados podem ser mostrados em visões diferentes, em uma planilha, um gráfico de barras ou em um gráfico de pizza.

Padrões de projeto variam em sua granularidade e no seu nível de abstração. Devido ao grande número de padrões de projeto e por alguns conterem propriedades semelhantes, foi necessário criar uma maneira para organizá-los, esta organização ajuda a compreender melhor e mais rápido os padrões. Segue a classificação na tabela 6.1, retirada de Duziani et al. (2004).

Escopo	**Criação**	**Estrutura**	**Comportamento**
Classe	Factory Method	Adapter (classe)	Interpreter Template Method
Objeto	Abstract Factory Buider Prototype Singleton	Adapter (objeto) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Mediator Observer State Strategy Visitor

Tabela 6.1: Organizações dos Padrões de Projeto.

O primeiro critério para a organização é chamado de finalidade, o que o padrão faz. Os padrões podem ter finalidade de criação, estrutural ou comportamental. O segundo critério é o escopo, especifica se o padrão se aplica primeiramente a classe ou a objeto.

(Gamma et al. (2000)) Os padrões de criação voltados para classes deixam alguma parte da criação de objetos para subclasses, enquanto que os padrões de criação voltados para objetos postergam esse processo para outro objeto. Os padrões estruturais voltados para classes utilizam a herança pra compor classes, enquanto que os padrões estruturais voltados para objetos descrevem maneiras de montar objetos. Os padrões comportamentais voltados para classes usam a herança para descrever algoritmos e fluxo de controle, enquanto coopera para executar uma tarefa que um único objeto não pode executar.

6.2 Aplicação para dispositivos móveis

Após os estudos relizados na seção 6.1 é proposto um padrão de projeto para a área da computação ubíqua. Este padrão servirá para uma aplicação ciente de contexto, esta aplicação consiste em dar apoio ao turista da cidade de Ouro Preto. Na tabela 6.1 o padrão proposto encaixaria com propósito de criação e escopo de classe. O padrão consiste em mostrar uma lista de imagens junto a um texto. Um exemplo da execução deste padrão na aplicação, seria a imagem de uma igreja e o seu nome, monstrando na tela para um usuário. Seguem as imagens, figura 6.1 e 6.2

Figura 6.1: Lista de Atividades

Figura 6.2: Atividade

Capítulo 7

Paradigmas de programação síncrona e assíncrona no Android

Em Debunne (2010), são criadas algumas situações em que é feito o download de imagem. Nas próximas seções é abordado um exemplo onde será preenchido um ListView com imagens em miniatura baixado da internet.

7.1 Um download de imagem

Baixar uma imagem da web é bastante simples, usando as classes HTTP fornecidas pelo framework.

No trecho de listing 7.1, primeiro é criado um cliente e um pedido de HTTP. Em seguida é feito um teste de conexão, caso seja bem-sucedido, a entidade que contém a imagem é decodificada para criar o bitmap resultante. Para ter acesso a Internet em aplicações que utilizam Android deve-se colocar uma permissão para acesso a internet no arquivo manifest.

Este método usado para fazer o downlaod da imagem pode oferecer resultados desagradáveis quando é feito algum movimento na barra de rolagem, pois este movimento faz acontece uma pausa. Acontece esta pausa porque a cada exibição de uma nova View deve esperar o download da imagem. Isso impede que o deslocamento seja suave.

7.2 Apresentando tarefas assíncronas

A classe AsyncTask (classe da API do Android) fornece uma das maneiras mais simples para disparar uma nova tarefa a partir do segmento. Primeiro é criada uma classe ImageDownloader, esta será responsável pela criação dessas tarefas. A ImageDownloader oferecerá um método de download que irá atribuir uma imagem baixada a um ImageView. A imagem é baixada utilizando sua URL.

Listing 7.1: Download Simples

```
static Bitmap downloadBitmap(String url) {
    final AndroidHttpClient client = AndroidHttpClient.newInstance("Android");
    final HttpGet getRequest = new HttpGet(url);

    try {
        HttpResponse response = client.execute(getRequest);
        final int statusCode = response.getStatusLine().getStatusCode();
        if (statusCode != HttpStatus.SC_OK) {
            Log.w("ImageDownloader", "Error " + statusCode +
      " while retrieving bitmap from " + url);
            return null;
        }

        final HttpEntity entity = response.getEntity();
        if (entity != null) {
            InputStream inputStream = null;
            try {
                inputStream = entity.getContent();
                final Bitmap bitmap = BitmapFactory.decodeStream(inputStream);
                return bitmap;
            } finally {
                if (inputStream != null) {
                    inputStream.close();
                }
                entity.consumeContent();
            }
        }
    } catch (Exception e) {
        getRequest.abort();
        Log.w("ImageDownloader",
    "Error while retrieving bitmap from " + url, e.toString());
    } finally {
        if (client != null) {
            client.close();
        }
    }
    return null;
}
```

O BitmapDownloaderTask é a classe que vai realmente fazer o download da imagem. Ela é iniciada usando o execute do método download, isso faz com que se tenha um retorno imediato, portanto, o método fica muito rápido. Em listing 7.2 está a apresentação das classes.

O método doInBackground simplesmente usa o método downloadBitmap que é mostrado na listing 7.2.

O método OnPostExecute é executado na chamada da thread, isso acontece quando a tarefa estiver concluída. Este método utiliza o bitmap resultante como um parâmetro, que é

Listing 7.2: Download assíncrono

```
public class ImageDownloader {

    public void download(String url, ImageView imageView) {
            BitmapDownloaderTask task = new BitmapDownloaderTask(imageView);
            task.execute(url);
        }
    }

  class BitmapDownloaderTask extends AsyncTask<String, Void, Bitmap> {
    private String url;
    private final WeakReference<ImageView> imageViewReference;

    public BitmapDownloaderTask(ImageView imageView) {
      imageViewReference = new WeakReference<ImageView>(imageView);
    }
    @Override
    protected Bitmap doInBackground(String... params) {
      return downloadBitmap(params[0]);
    }
    @Override
    protected void onPostExecute(Bitmap bitmap) {
      if (isCancelled()) {
        bitmap = null;
      }

      if (imageViewReference != null) {
        ImageView imageView = imageViewReference.get();
        if (imageView != null) {
          imageView.setImageBitmap(bitmap);
        }
      }
    }
  }
}
```

simplesmente associado ao ImageView fornecido. Note que o ImageView é armazenado como um WeakReference, de modo que um download em andamento não mata a atividade atual.

Um problema para esse código acontece quando ocorre alguma ação na barra de rolagem, a ListView recicla as imagens que foram exibidas. Assim enquanto baixa a nova imagem que vai aparecer, sempre existe uma imagem, e esta imagem é a que foi baixada por ultimo. Quando o download da imagem é finalizado a nova imagem aparece substituindo a que está no seu lugar. Isso acontece pois é utilizado apenas um ImageView para todas as imagens, assim uma vai subscrevendo a outra que é mostrada na tela.

Listing 7.3: Download assíncrono com feedback

```
static class DownloadedDrawable extends ColorDrawable {
    private final WeakReference<BitmapDownloaderTask> bitmapDownloaderTaskReference;
    public DownloadedDrawable(BitmapDownloaderTask bitmapDownloaderTask) {
        super(Color.BLACK);
        bitmapDownloaderTaskReference =
            new WeakReference<BitmapDownloaderTask>(bitmapDownloaderTask);
    }

    public BitmapDownloaderTask getBitmapDownloaderTask() {
        return bitmapDownloaderTaskReference.get();
    }
}
```

7.3 Manipulando Concorrência

Para resolver o problema citando na seção 7.2, e adicionado uma informação extra ao ImageView usando uma subclasse dedicada Drawable. Esta classe vincula uma imagem temporária ao ImageView enquanto o download está em andamento. Em listing 7.2 segue o código da classe DownloadedDrawable

Esta implementação resultará no ImageView exibindo um fundo preto, enquanto seu download está em andamento. Pode-se também colocar alguma mensagem do tipo "download em andamento", em vez do fundo preto. Mais uma vez, observe o uso de um WeakReference para limitar as dependências objeto.

Capítulo 8

Testes de desempenho

Pensando em um melhor desempenho para o padrão proposto, foram realizados alguns testes referentes ao tempo de resposta para o download da imagem.

Estes testes são necessários, pois na criação de aplicações que se espera uma resposta, devem ter uma atenção especial principalmente quando são operações de rede, onde podem ocorrer atrasos imprevisíveis. Alguns usuários irão tolerar algumas pausas, principalmente se estiverem sendo fornecidas informações sobre o que está acontecendo (feedback), mas se o tempo de resposta for muito demorado pode acontecer um abandono por parte do usuário. (Debunne (2010))

Os testes foram feitos em três situações com uma conexão HTTP, com velocidade de internet 1 MB e um arquivo de 39,3 KB.

A seguir uma breve descrição das três situações testadas:

- A situação 1 é sequencial, esta maneira é a mais simples de implementar, mas a mais custosa e que tem um pior desempenho como mostra no gráfico da figura 8.1. Nesta situação a tela só é exibida quando o download da imagem já terminou, assim caso tenha várias imagens em uma tela, só exibira as imagens quando terminar o download de todas. Se acontecer alguma ação na barra de rolagem, o conteúdo que estiver que aparecer, irá demorar, pois a tela só é exibida com a conclusão do download.

- A situação 2 foi assíncrona, seu desempenho foi bem superior ao da situação 1, pois a tela é gerada independente do download ter finalizado ou não. Assim à medida que os downloads das imagens estão finalizando, estas são mostradas na tela, independentes umas das outras. Nessa situação utiliza-se o recurso de threads, com isso a aplicação consegue fazer várias coisas ao mesmo tempo. Com as threads ou processos leves é possível que um processo tenha múltiplas linhas de controle, já um processo tradicional é permitido uma linha de controle. (Tanenbaum e Woodhull (2000))

- A situação 3 é bem parecida com a situação 2, a diferença está na hora em que a tela é mostrada e o download não acabou, na situação 2 não aparece nada neste instante, já na situação 3 tem-se a possibilidade de colocar um feedback para o usuário. Um exemplo é colocar a palavra "carregando", aonde a imagem irá aparecer.

A figura 8.1 é apresentado uma gráfio com o desempenho das três situações.

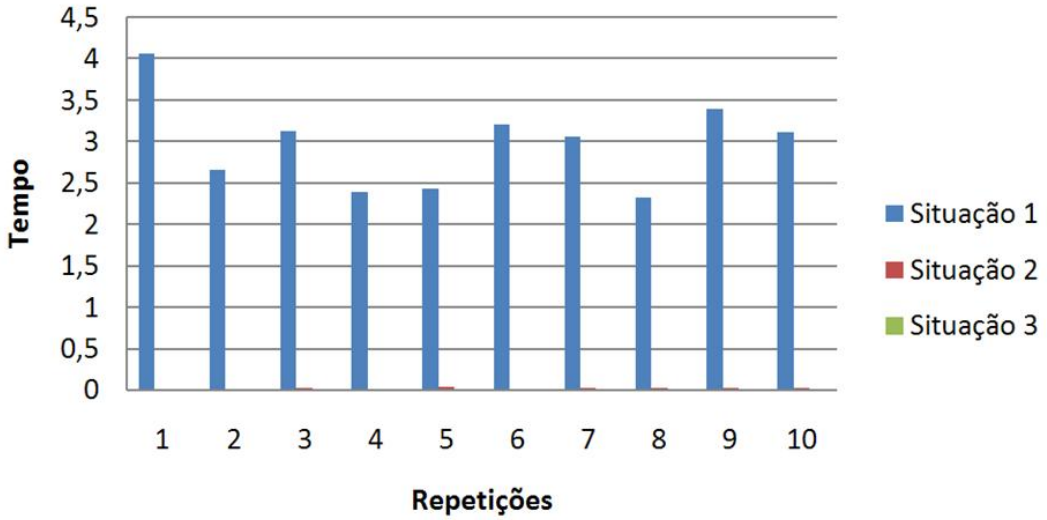

Figura 8.1: Gráfico dos teste

Nas três situações foram obtidos 10 tempos, e ao fim dos cálculos foi calculada a média e variância dos tempos. Segue uma tabela com os valores:

	Situação 1	**Situação 2**	**Situação 3**
Média	2,975	0,022	0,016
Variância	0,293	6,627	2,795

Tabela 8.1: Resultados dos Testes.

Ao fim dos testes e avaliando os resultados pode-se perceber que as situações 2 e 3 tiveram um desempenho bem superior ao da situação 1, isso pode ser justificado pois nas situações 2 e 3 foram usadas threas, com elas é permitido fazer várias coisas ao mesmo tempo, pois quando uma thread é executado, um retorno já é feito imediatamente, podendo assim chamar outra thread. Com isso uma thread não precisa esperar a outra treminar para começar.

Mas a melhor situação dentre as três é a situação 3, esta possibilita um feedback para o usuário, pois enquanto o download da imagem está sendo executado, é permitido colocar algumas mensagens no local onde a imagem vai aparecer.

Capítulo 9

Proposta de um novo Padrão de Projeto

9.1 Elementos

De acordo com Gamma et al. (2000) um padrão contêm quatro elementos essenciais, a seguir uma descrição dos mesmos para o novo padrão proposto.

9.1.1 Nome

O nome do novo padrão será CloudDownloader.

9.1.2 Problema

Este padrão tem como objetivo fazer download de uma imagem e vinculá-la a um texto. Podem ser colocado vários conjuntos (imagem + texto) destes na tela.

9.1.3 Solução

Apresentação da descrição UML (Unified Modeling Language) para o padrão, figura 9.1.

- **CloudImageText** é a classe principal, esta da origem ao novo Padrão de Projeto.
- **MyCustomAdapter** responsável por vincular o texto e a imagem a uma View, através do método getView, essa View é ligada diretamente a um layout que é mostrado na tela. Essa classe é implementada dentro de *CloudImageText* e herda de *ArrayAdapter*, classe da API do Android.
- **backgroudLoadListView** responsável por mostrar na tela do dispositivo o layout que é gerado em *MyCustomAdapter*. Essa classe também é implementada dentro de *CloudImageText* e herda de *AsyncTask*, classe da API do Android.

- **ImageDownloader** classe responsável por baixar uma imagem através de uma URL passada por parameto. É criado um objeto desta classe dentro de *MyCustomAdapter*, este objeto que realiza o download.

Além dessas classes implementadas dentro de **CloudImageText** existe seu construtor, este recebe como parâmetro dois vetores de string, alguns métodos de ação em cliques na tela, mas os métodos de clique podem variar.

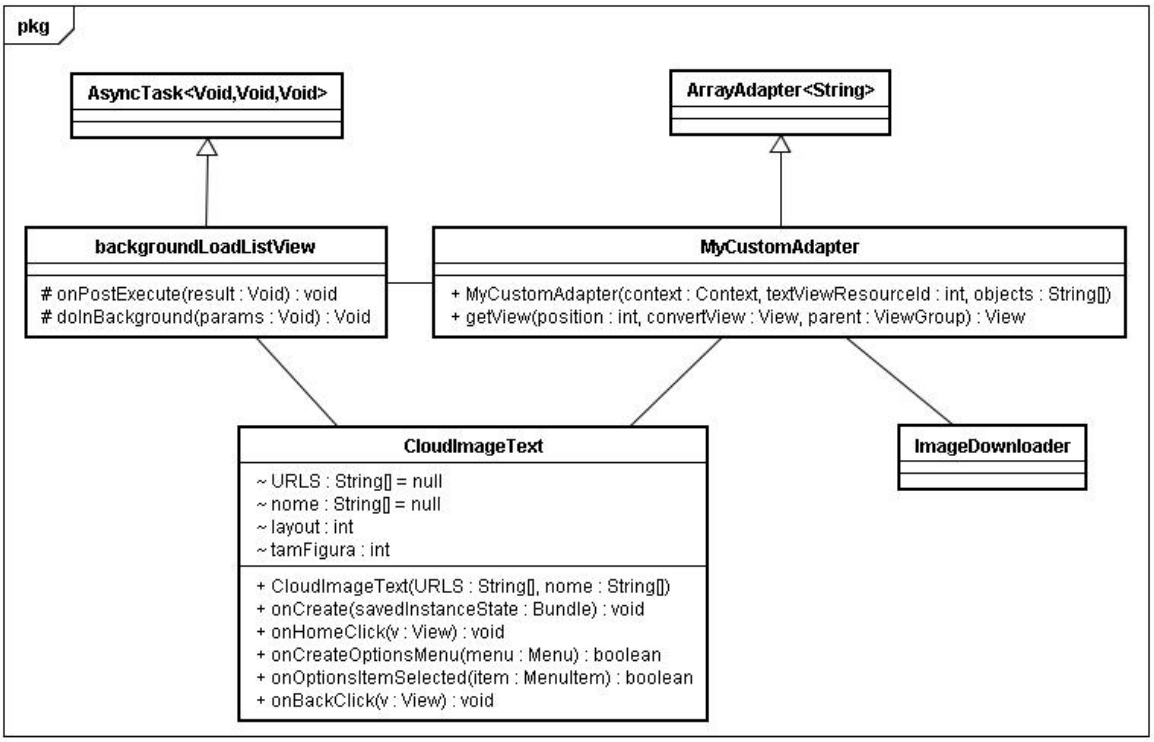

Figura 9.1: Digrama UML classe CloudImageText

9.1.4 Consequências

Uma vantagem é que se uma atividade demorar a responder, ela será eliminada. Esta é uma caracteristica particular do Android.

Uma desvantagem é que este padrão é fechado, ou seja so funciona para o sistema operacional Android, não funcionando nos outros sistemas tradicionais.

9.2 Aplicação utilizando o padrão

A aplicação consiste em um conjunto de classes que são responsáveis por inserir vários conjuntos de imagens e textos em um layout. A imagem é baixada pela classe **ImageDownloader**

e o texto é passado em um vetor "nome". Essa associação do texto, imagem e layout é feita no método getView, dentro da classe **MyCustomAdapter**.

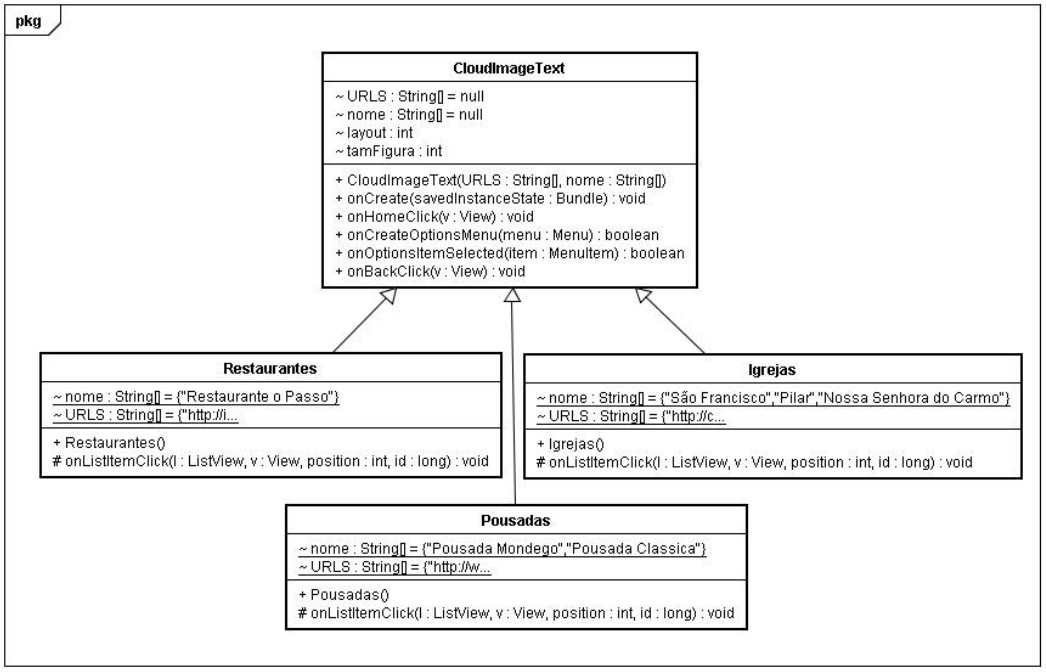

Figura 9.2: Digrama UML subclasses e superclasse

Na figura 9.2 é mostrado o digrama UML da aplicação que utiliza o padrão. As classes Restaurantes, Igrejas e Pousadas herdam de **CloudImageText**, pois estas têm as mesmas funcionalidades, com isso pode-se fazer reutilização de código. A diferença entre essas classes é somente nos parâmetros que são passados para a classe **CloudImageText**. Nessas classes foram acrescentadas o método onListItemClick, este método captura a origem do clique na tela do dispositivo, para este caso o método captura a posição do conjunto imagem e texto na tela.

A aplicação que está utilizando o padrão é composta por algumas listas de atividade e as atividades, com isso quando se clica em alguma atividade da lista, é gerada outra tela com a atividade. Esta atividade tem a mesma idéia das classes apresentadas anteriormente, que é baixar uma imagem e fazer vínculo a um texto e o conjunto em um layout. Assim todas as classes que contêm as atividades herdam da classe principal **CloudImageText**.

Na figura 9.3 é mostrado o diagrama UML completo. Assim todas as classes herdam da classe principal a **CloudImageText**, pois todas utilizam do mesmo recurso que é baixar uma imagem juntar a um texto e colocar este conjunto na tela. As diferenças acontecem somente na parte de ações de cliques na tela, estes métodos são tratados em cada subclasse. Os métodos de ação de cliques não são abstratos na classe **CloudImageText**, pois contêm métodos diferentes nas classes que herdam dela, assim não faria sentido colocar os métodos

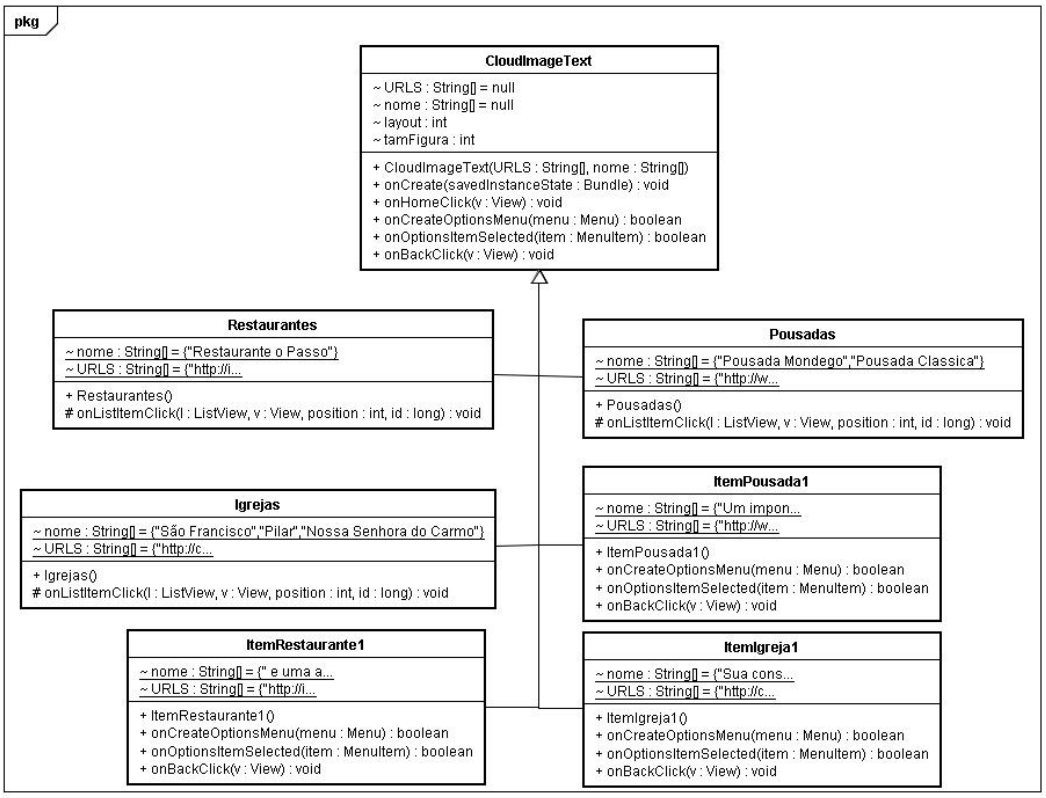

Figura 9.3: Digrama UML subclasses e superclasse

abstratos.

9.3 Funcionamento do Padrão

O padrão que é proposto tem como finalidade colocar uma imagem junto a um texto em um layout. Para a utilização do padrão devem-se passar dois vetores de string no construtor da subclasse que herda da superclasse. O primeiro vetor de string deve ser URL's, pois o primeiro parâmetro são as imagens e estas são baixadas através de algumas interface de conexão. Já o segundo parâmetro são os textos ou nomes que tem vínculo com a imagem. Com esses parâmetros passados a superclasse fica encarregada de baixar a imagem e junta-la ao texto em um layout.

Capítulo 10

Conclusões

Neste trabalho foram analisados alguns tipos de interfaces para dispositivos móveis, uma interface mais simples, uma com abas e finalizando com uma contextual.

Também foi proposto um novo padrão de projeto chamado CloudDownloader, este padão tem o objetivo de fazer o download de uma imagem e fazer vínculo desta a um texto, podendo ser colocados vários conjuntos (imagem + texto) na tela.

Como o padrão CloudDownloader utiliza recursos de download, foram feitos testes pensando em desempenho, pois como se trata de aplicações responsivas é esperado pelo usuário uma certa velocidade ou algum feedback do que está acontecendo.

Para trabalho futuros pretende-se elaborar este padrão para outros dipositivos, um exemplo é o iPhone, pois este padrão é fechado funcionando somente em Android.

Referências Bibliográficas

Android (2010). Android developers. `http://developer.android.com/index.html`.

Berkenbrock, C.D.M. Hirata, C. F. C. P. M. (2009). Requisitos de usabilidade para o desenvolvimento e avaliação de aplicações cooperativas móveis. *Proceedings of the 2009 13th International Conference on Computer Supported Cooperative Work in Design*.

Cheverst, K.; Davies, N.; Mitchell, K.; Friday, A. e Efstratiou, C. (2000). Developing a context-aware electronic tourist guide: Some issues and experiences. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 17–24.

Debunne, G. (2010). Multithreading for performance. `http://android-developers.blogspot.com/2010/07/multithreading-for-performance.html`.

Duziani, C. F. M.; Bonifácio, A. S.; e Gomes da Costa, Y. M. e Ângelo Fabris, A. (2004). Um estudo sobre padrões de projeto aplicados a garbage collection. *Advanced Information Networking and Applications, International Conference on*.

Gamma, E.; Helm, R. e Johnson, R. (2000). *Padrões de Projeto*. Bookman, 1 edição.

Hirata, C.M. Berkenbrock, C. (2008). Implementação e análise de uso de uma aplicação para edição cooperativa em dispositivos móveis. *Simpósio Brasileiro de Sistemas Colaborativos*.

Huang, T.-C.; Yang, C.-S.; Bai, S.-W. e Wang, S.-H. (2003). An agent and profile management system for mobile users and service providers. *Advanced Information Networking and Applications, International Conference on*, pp. 0–574.

Ribeiro, D. F. (2007). Estudo de interface humano-máquina em dispositivos móveis. *Departamento de Informática e Estatística - Universidade Federal de Santa Catarina*.

Tanenbaum, A. S. e Woodhull, A. S. (2000). *Sistemas Operacionais*. Bookman, 2 edição.

Zemel, T. (2009). Padrões de projeto (ou design patterns): o que são, para que servem e qual sua implicação de uso. `http://codeigniterbrasil.com/passos-iniciais/padroes-de-projeto-ou-design-patterns-o-que-sao-para-que-servem-e-qual-sua-implicacao-de`