UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

COORDENAÇÃO DE LICENCIATURA EM INFORMÁTICA

DESENVOLVIMENTO DE SISTEMAS PARA INTERNET E DISPOSITIVOS MÓVEIS

OÉLITON CAPPELLETTO

CONTROLE DE RECEITAS E DESPESAS PARA DISPOSITIVOS ANDROID

MONOGRAFIA DE ESPECIALIZAÇÃO

FRANCISCO BELTRÃO

2014

OÉLITON CAPPELLETTO

CONTROLE DE RECEITAS E DESPESAS PARA DISPOSITIVOS ANDROID

Monografia de Especialização apresentada a Coordenação de Licenciatura em Informática, da Universidade Tecnológica Federal do Paraná, como requisito parcial para obtenção do título de “Especialista em Desenvolvimento de Sistema para Internet e Dispositivos Móveis”.

Orientador: Prof. MSc.Robison Cris Brito

FRANCISCO BELTRÃO

2014

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Francisco Beltrão
Diretoria de Pesquisa e Pós-Graduação
Curso de Especialização em Desenvolvimento de Sistemas para Internet e Dispositivos Móveis

TERMO DE APROVAÇÃO

Dia 02 do mês de outubro de 2014 às: 20:50 horas, na sala COLIN - Anfiteatro do Campus Francisco Beltrão, realizou-se a apresentação pública da monografia pelo estudante Oéliton Cappelletto, intitulada “Controle de Receitas e Despesas para Dispositivos Android.” Finalizada a apresentação e arguição, a Banca Examinadora declarou **aprovada** a monografia do estudante, como requisito parcial para obtenção do título de Especialização em Desenvolvimento e Sistemas para Internet e Dispositivo Móveis.

_____
Professor MSc.Robison Cris Brito - UTFPR
(Orientador)

_____
Professor Andrei Carniel - UTFPR
(Convidado)

_____
Professor Vinícius Pegorini - UTFPR
(Convidado)

_____
Professor Dr. Ademir Roberto Freddo - UTFPR
(Coordenação)

A folha de aprovação com as rubricas encontram-se disponíveis na Diretoria de Pesquisa e Pós-Graduação, UTFPR, Francisco Beltrão.

AGRADECIMENTOS

Certamente estes parágrafos não irão atender todas as pessoas que fizeram parte dessa importante fase de minha vida. Portanto, desde já peço desculpas àquelas que não estão presentes entre essas palavras, podem estar certas que fazem parte do meu pensamento e de minha gratidão.

Ao professor orientador Prof. MSc. Robison Cris Brito, pela perseverança e a confiança em mim na conclusão deste trabalho, sempre incentivando e motivando, desde o trabalho de estágio até este momento, sempre disponível para ajudar e orientar em tudo o que fosse possível.

A todos os amigos e colegas de trabalho e de estudo que ajudaram em tudo o que puderam, principalmente nos períodos difíceis.

E a minha esposa, Leonice Pinheiro, que não deixou em momento algum que eu pudesse esquecer ou desistir deste trabalho, sacrificando seu tempo e nosso tempo para que pudesse concluí-lo.

RESUMO

As empresas de software desenvolvem sistemas que auxiliam no gerenciamento e prevenção a riscos financeiros de empresas, mas não é comum ver software para o gerenciamento financeiro doméstico. Grande parte das pessoas acabam endividadas justamente por falta de planejamento, organização e por não ter um controle financeiro próprio. Pensando neste contexto, o presente trabalho propõe o desenvolvimento de um software para dispositivos móveis Android (*smartphone* ou *tablet*) que ajude as pessoas a realizarem um controle financeiro. A vantagem de ser para dispositivos móveis é que o usuário pode levar seu controle financeiro para qualquer lugar, fazendo os lançamentos e pesquisas no momento em que necessitarem. Como resultado tem-se um software com interface atraente, recursos de gráfico e de fácil uso que pode ser facilmente utilizado por qualquer pessoa.

Palavras-chave: Aplicativo. Controle. Financeiro. Dispositivos Móveis. Android.

ABSTRACT

Software companies develop systems that assist in the prevention and management of financial risk companies, but it is not common to see software for home financial management. Most people end up in debt just by lack of planning, organization and for not having a proper financial control. Thinking this context, this paper proposes the development of a software for Android (smartphone or tablet) mobile devices that helps people achieve financial control. The advantage of being mobile is that you can take your financial control anywhere, doing releases and research at the time they need. As a result one has an attractive interface software, graphic capabilities and ease of use that can be easily used by anyone.

Keywords: Application. Control. Financial. Mobile Devices. Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

LISTA DE SIGLAS

ADT	Android Development Tools
API	Application Programming Interface
CPL	Common Public License
EPL	Eclipse Public License
IDE	Integrated Development Environment
JDK	Java Development Kit
SDK	Source Development Kit
SQL	Structured Query Language
UML	Unified Modeling Language
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Segundo Sabrina Okada (2012), o secretário da Fazenda do município, Moacir Bertaci, fala que o Brasil é um dos países com o maior nível de endividamento e que isso acontece pela falta de planejamento financeiro das pessoas, o que faz com que tenham dificuldades em manter equilibrado seu orçamento durante todo o mês utilizando apenas o salário, por isso muitos acabam contraindo dívidas ao optarem por empréstimos.

No dia a dia das pessoas, observa-se a necessidade de estimular a educação financeira, para evitar o aumento no número de devedores, proporcionando conhecimento sobre finanças e auxiliando na tomada de decisões.

É necessário saber o que comprar e como comprar para não adquirir dívidas que não possam ser pagas.

Para manter um bom controle dos gastos mensais é preciso ter o hábito de gastar menos do que se ganha, é preciso ter em mente que certos gastos são desnecessários e podem por em risco todas as metas traçadas. Observa-se que o grande problema não são as contas “maiores”, como aluguel, água, luz ou telefone, e sim, despesas do dia a dia, que acumuladas refletem em valores altos nos finais dos meses.

Sendo assim, é preciso traçar metas financeiras e levar sempre controlados os gastos para não acabar fazendo parte do quadro da porcentagem de pessoas endividadas. Uma boa forma de realizar o controle financeiro pessoal, é ter acesso a um sistema de controle de fácil acesso que esteja sempre presente quando necessário, algo que se tem hoje com o advento dos dispositivos móveis.

Segundo Jamil Chade (2014), do jornal Estadão, o número de celulares vai chegar a 7 bilhões até dezembro de 2014, o representara 96% de todos os habitantes.

Assim como existem muitas opções de dispositivos móveis, existem também muitas opções de software, muitos deles gratuitos, sendo que alguns deles se destinam ao controle financeiro, permitindo contabilizar as despesas do dia a dia, assim como planejar situações financeiras futuras. Entretanto, muitos destes softwares são complexos, trazendo muitos recursos, dos quais os usuários não

necessitam no dia a dia, o que torna seu uso difícil, em especial, para pessoas com poucos conhecimentos de utilização de dispositivos móveis.

Neste contexto, o software proposto tem o intuito de controlar as despesas e receitas do dia a dia com facilidade, utilizando apenas alguns cliques na tela do dispositivo móvel, sendo possível ver quanto foi gasto e quanto pode-se gastar no mês.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver um software que permita o controle financeiro (contas a pagar e receber) utilizando um dispositivo móvel Android.

1.2.2 Objetivo Específico

Dentre os objetivos específicos do trabalho, destacam-se:

Desenvolver a Análise de um aplicativo simples para o controle financeiro diário para dispositivos móveis;

Desenvolver o aplicativo utilizando a plataforma Android;

Implementar recursos de gráficos e técnicas para melhorar a usabilidade do aplicativo.

1.3 JUSTIFICATIVA

Atualmente, com o fato de muitas pessoas permanecerem fora de casa durante todo o dia, fazendo suas refeições nas ruas, passando por muitas vitrines de lojas apresentando descontos em produtos, e com a facilidade de compras pela internet, existe a necessidade de um controle financeiro para dispositivos móveis, permitindo o controle dos gastos a qualquer hora e lugar.

Quando se utiliza planilhas ou agenda para controlar gastos, como muitas vezes estes não se encontram com as pessoas no dia a dia, pode ocorrer dos lançamentos serem realizados apenas uma vez ao dia, muitas vezes a noite, sendo comum nestes casos do usuário esquecer-se de fazer os lançamentos.

Pensando nesta situação, foi desenvolvido um software para dispositivos móveis que possibilita de forma prática e rápida realizar o controle financeiro diário. Este software pode ser instalado em dispositivos móveis de baixo custo, como *smartphones* e *tablet* Android, permitindo que o usuário acesse a qualquer hora, em qualquer lugar e realize o controle financeiro.

A plataforma Android é formada por um sistema operacional e um ambiente de desenvolvimento, este coordenado pela Google.

Em alguns outros sistemas operacionais para dispositivos móveis, os programas não são código aberto, de modo que qualquer alteração ou utilização de recursos de baixo nível na plataforma se torna difícil, em muitas situações, estes só podem ser feitos comprando *frameworks* do próprio desenvolvedor ou de terceiros.

Já o Android é a primeira plataforma para aplicações móveis completamente livre e de código aberto, o que representa uma grande vantagem para os aplicativos desenvolvidos para esta plataforma, uma vez que diversos programadores do mundo poderão contribuir para melhorar a plataforma.

Segundo site Tecmundo (2014), o sistema operacional Android é o mais utilizado no mundo para *smartphones* liderando o mercado, o que motivou o desenvolvimento do presente trabalho para a plataforma Android.

Os dados referentes ao domínio de mercado de cada plataforma em 2012 e 2013 são apresentados na Tabela 1.

Tabela 1 – Visão do mercado de sistemas operacionais de smartphones.

Sistemas Operacionais	2013 Domínio de mercado (%)	2012 Domínio de mercado (%)
Android	78,6	69,0
iOS	15,2	18,7
Windows Phone	3,3	2,4
BlackBerry	1,9	4,5
Outros	1,0	5,4
Total	100	100

Fonte: Dificuldade Zero, 2014.

1.4 ESTRUTURA DO TABALHO

O trabalho está dividido em 4 Capítulos. Destes, este é o primeiro e apresenta a introdução, objetivos e justificativa para a realização deste trabalho. O Capítulo 2 apresenta a fundamentação teórica para o desenvolvimento do mesmo. Já o Capítulo 3 apresenta os materiais e métodos utilizados no desenvolvimento. O Capítulo 4 apresenta os resultados do trabalho, e por fim, o Capítulo 5 apresenta as conclusões, as dificuldades encontradas e as sugestões para trabalho futuro.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta o referencial teórico do trabalho, exibindo as características do sistema Android, assim como os conceitos básicos de seu funcionamento. Por fim, serão apresentados alguns aplicativos para dispositivos móveis semelhantes ao proposto pelo presente trabalho.

2.1 ANDROID

O Android Surgiu em novembro de 2007 através de um consórcio denominado Open Handset Alliance ou Aliança de Telefonia Móvel Aberta, que seria um grupo formado por gigantes do mercado de telefonia de celulares liderados pela Google. (Ricardo L. Lecheta, 2009, p.23).

Android é o primeiro sistema operacional para dispositivos móveis baseado em Linux, o que revolucionou o mercado mundial de aparelhos celulares. (Ricardo L. Lecheta, 2009, p.26).

Proporciona aos usuários e desenvolvedores uma SDK (*Standard Development Kit*, ou Kit de Desenvolvimento Padrão) para a criação de aplicativos móveis utilizando a linguagem Java, que é uma linguagem de alto nível, portável para diversas outras plataformas e que é baseada na orientação a objetos.

A primeira versão do Android foi disponibilizada em Abril de 2009 sob o nome de Cupcake, sendo esta a versão 1.5.

O Android é a primeira plataforma para aplicação móveis completamente livre e de código aberto (*open-source*), o que representa uma grande vantagem para sua evolução, uma vez que diversos programadores do mundo poderão contribuir para melhorar a plataforma. (Ricardo L. Lecheta, 2009, p.27).

Uma característica da plataforma Android é que não existem diferenças entre as aplicações e recursos nativos do aparelho em relação às aplicações construídas a partir da SDK. Isso significa que aplicações poderosas podem ser desenvolvidas para serem utilizadas com todos os recursos disponíveis no aparelho, inclusive os de baixo nível. (ABLESON, 2012).

> Criar aplicativos para Android garante flexibilidade ao desenvolvedor, dinamismo e a possibilidade de explorar a fundo as capacidades do equipamento, pois permite a utilização sem restrições de qualquer funcionalidade do dispositivo móvel. Todos os recursos de hardware e recursos essenciais do sistema operacional podem ser aproveitados utilizando as API`s (*Application Programming Interface*) nativas do próprio sistema. (Ricardo L. Lecheta, 2009, p.24).

A Figura 1 apresenta os principais elementos e camadas da arquitetura Android.

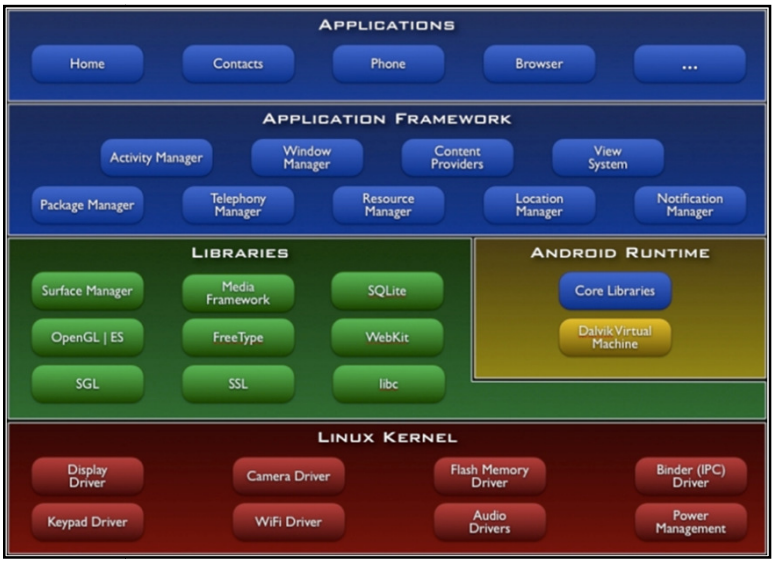

Figura 1 - Arquitetura Android.
Fonte: DroidNew, 2014.

No nível que apresenta o Linux Kernel Figura 1, temos a base da pilha, para desenvolvê-la foi utilizado à versão 2.6 do Sistema Operacional Linux, nele fica os programas de gerenciamento de memória, configurações de segurança e vários drives de hardware.

A camada de biblioteca é um conjunto de instruções que dizem ao dispositivo como lidar com diferentes tipos de dados, incluindo um conjunto de biblioteca C / C + + usadas por diversos componentes do sistema e são expostas a desenvolvedores através da estrutura de aplicativo Android.

A camada de tempo de execução Android Runtime, inclui um conjunto de bibliotecas do núcleo Java (*Core Libraries*). Para desenvolver aplicações para o Android, os programadores utilizam a linguagem de programação Java, nesta camada encontraremos a Máquina Virtual Dalvik (DVM).

Camada de *framework* de aplicação (*Application Framework*), programas que gerenciam as aplicações básicas do telefone.

Camada de aplicações e as funções básicas do dispositivo. Esta é a camada de interação entre o usuário e o dispositivo móvel, nela encontramos aplicativos cliente de e-mail, programa de SMS, calendário, mapas, navegador, contatos entre outros.

2.2 APLICATIVO PARA CONTROLE DE RECEITAS E DESPESAS PARA DISPOSITIVOS ANDROID

No Google Play (2014) existem vários aplicativos para controle de gastos, todos para auxiliar no controle de despesas e receitas, uns com mais recursos do que outros, sendo que todos estes aplicativos podem ser baixados e testados, muitos deles gratuitos.

O aplicativo controle de despesas da AGBA Android Apps (2014) - Figura 2, está disponível gratuitamente para *download*, sendo um aplicativo fácil de utilizar e com bons recursos de software, um deles é o cadastro de despesas com um *layout* agradável, outro é uma tela em que pode ser visto o resumo dos lançamentos em lista, além da opção de gerar gráficos e exporta os dados para uma planilha do Microsoft Excel.

Figura 2 - Controle de despesas da AGBA Android Apps.
Fonte: Google Play, 2014.

O controle financeiro disponibilizado gratuitamente pela Ikoa (2014), este apresentado na Figura 3, é mais simples do que o aplicativo apresentado anteriormente, tendo uma tela para lançamento, que após a gravação apresenta em uma lista os resumos dos lançamentos realizados.

Um diferencial deste controle é o cadastro dos participantes, sendo assim um aplicativo multiusuário, muito útil pois pode controlar as despesas de outros membros da família, como filhos e esposa.

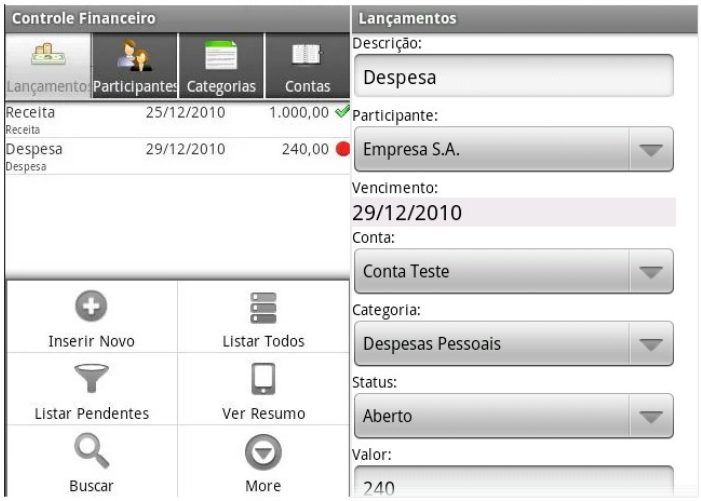

Figura 3 - Controle financeiro da Ikoa.
Fonte: Google Play, 2014.

Já o controle de gastos da Aon Sistemas (2014), Figura 4, é gratuito e possui telas para lançamentos com um *layout* simples. Permite realizar os lançamentos, tela com os resumos e também um gráfico tipo pizza para ver onde foram realizados os lançamentos.

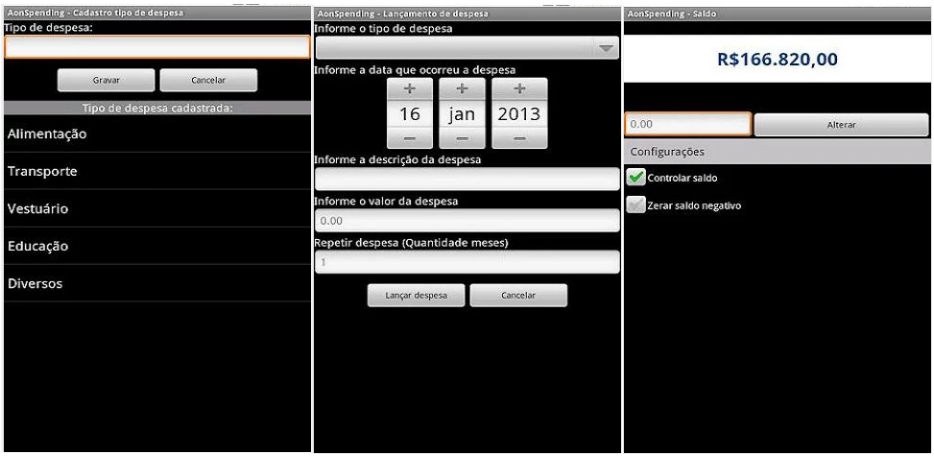

Figura 4 - Controle de gastos da Aon Sistemas.
Fonte: Google Play, 2014.

3 METODOLOGIA

Este capítulo apresenta as ferramentas utilizadas para a elaboração do aplicativo móvel de controle de receitas e despesas. Também serão apresentadas as metodologias utilizadas para atingir o resultado final.

3.1 MATERIAIS

Os softwares utilizados para desenvolver o aplicativo de controle de receitas e despesas são apresentados no Quadro 1.

Ferramenta/Tecnologia	Versão	Referência	Finalidade
Android SDK	2.2	http://developer.android.com/sdk/index.html	Ambiente com bibliotecas e emulador do Android.
SQLite	3.5.9	http://www.sqlite.org/changes.html#version_3_5_9	Banco de dados.
Eclipse Juno	4.2	http://www.eclipse.org	Ambiente de desenvolvimento.
Framework HoloGraphLibrary	1.0	https://bitbucket.org/danielnadeau/holographlibrary	*Framework* para geração de gráficos.
Astah Community	6.7.0(Modelo de versão 36)	http://astah.net/editions/community	Documentação da modelagem baseada na UML.
Linguagem Java	JDK 7.0	http://www.oracle.com	Linguagem de programação.
Android Development Tools (ADT)	22.3.0	http://developer.android.com/index.html	*Plugin* para o Eclipse IDE com pacotes com base na API do Android.

Quadro 1 – Ferramentas utilizadas no projeto.
Fonte: Autoria própria.

Para o desenvolvimento da aplicação foi utilizado o sistema operacional Microsoft Windows 7 (*Seven*) Professional e a ferramenta Astah Community para análise da aplicação.

Para exemplificar graficamente o uso de cada ferramenta, a Figura 5 é apresentada, onde é possível verificar que o Astah Community foi utilizado para a análise do sistema, influenciando em como as demais ferramentas seriam utilizadas. Já o *Framework* HoloGraphLibrary, o *plugin* do Android IDE (Integrated Development Environment) *development Tools* (ADT) e o Android SDK são integrados ao

Ambiente de desenvolvimento Eclipse, que gerará uma aplicação móvel que interage com o banco de dados SQLite.

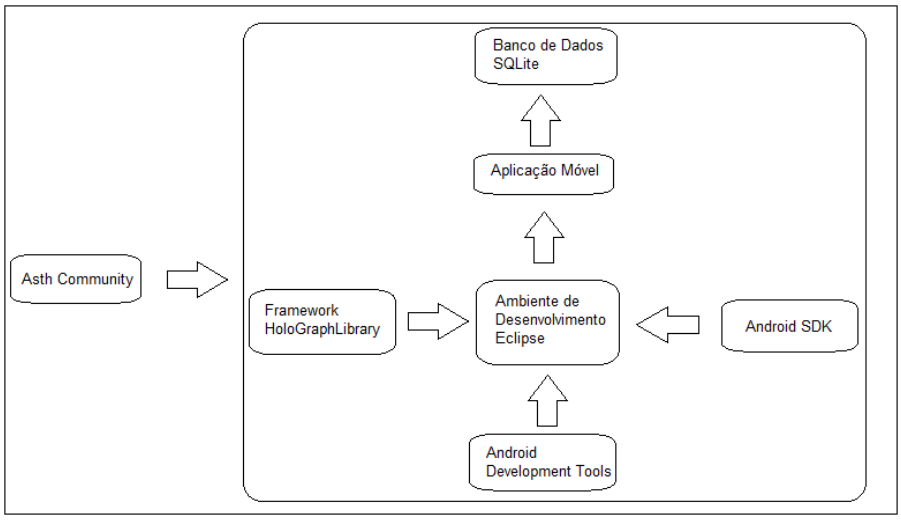

Figura 5 – Esquema dos materiais utilizados.
Fonte: Autoria própria.

As características e dados relevantes sobre cada ferramenta são apresentadas na sequência:

a) IDE Eclipse: A IDE Eclipse (2014) foi criada pela Internacional Business Machines (IBM). É um ambiente de desenvolvimento de aplicações feita com a linguagem Java, disponibilizado sob as licenças CPL e EPL. Este ambiente proporciona ao desenvolvedor, ferramentas úteis de formatação de código, marcação de pontos de parada (*breakpoints*) e outras ferramentas úteis para facilitar o desenvolvimento.

b) Android Development Tools (ADT): É um *plugin* para o Eclipse IDE que é projetado para construir aplicativos Android. O ADT amplia os recursos do Eclipse para ser possível criar telas com rapidez e facilidade, o *plugin* adicionar pacotes com base na API.

c) Android SDK: O Android SDK (2014), é uma ferramenta *open-source* que provê de bibliotecas e ferramentas de desenvolvimento necessárias para construir aplicações Android, assim como auxilia nos testes e execuções dos programas.

d) SQLite: O SQLite (2014) é um banco de dados transacional nativo do sistema Android que não depende de instalação de servidores e não possui necessidade de configurações.

e) *Framework* HoloGraphLibrary: HoloGraphLibrary (2014) é uma classe para geração de gráficos que pode apresentar os gráficos de três formas, em colunas, linhas ou pizza. A utilização desta classe facilita na geração de gráficos para dispositivos móveis Android.

f) Astah Community: O Astah Community (2014) é uma ferramenta que permite a fácil manipulação de todos os modelos de diagramas existentes para modelagem de software baseados no padrão UML. A ferramenta possui recursos que objetivam a praticidade de trabalho, como configuração de atalhos de componentes, bem como o alinhamento automático destes.

3.2 MÉTODOS

O desenvolvimento do aplicativo para controle de gastos e pesquisas de preço foi dividido em várias etapas, seguindo o modelo sequencial linear de Pressman (2006). Essas etapas são:

a) Análise: O caso de uso foi definido como forma de representar o problema para o qual seria proposta uma solução. O caso de uso foi documentado utilizando a ferramenta Astah Community UML.

b) Projeto: O diagrama de classe, com a linguagem UML, foi produzido utilizando a ferramenta Astah Community para facilitar a manutenção das classes do projeto, assim como para ter uma visão geral do código.

c) Desenvolvimento: O desenvolvimento foi realizado utilizando as tecnologias listadas na seção 3.1 - Materiais

d) Testes: Para os testes foi instalado o aplicativo em quatro *smartphones* distintos, com diferentes pessoas para testes.

4 APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS

Este capítulo apresenta uma visão geral da análise realizada para o desenvolvimento do sistema. Este não é apresentado em detalhes, uma vez que o objetivo principal deste trabalho é o desenvolvimento o sistema para dispositivo móvel de controle de receitas e despesas, assim como a utilização de alguns recursos gráficos da plataforma Android.

Após, é apresentado o aplicativo desenvolvido, focando nas telas e suas funcionalidades. Por fim, são apresentados os detalhes técnicos de desenvolvimento do aplicativo móvel, assim como os testes realizados no programa.

4.1 MODELAGEM DE SOFTWARE

Está seção apresenta a análise do sistema para controle de receitas e despesas que definem as atividades principais do sistema, diagrama de caso de uso, diagrama de classe, diagrama de atividade e diagrama de entidade e relacionamento.

Assim, após a definição do tema o primeiro passo no desenvolvimento do aplicativo foi a análise para definir as funcionalidades destes. Foi optado inicialmente no desenvolvimento do diagrama de caso de uso mostrado na Figura 6, no qual foram apresentados os recursos estariam a disposição do usuário no aplicativo.

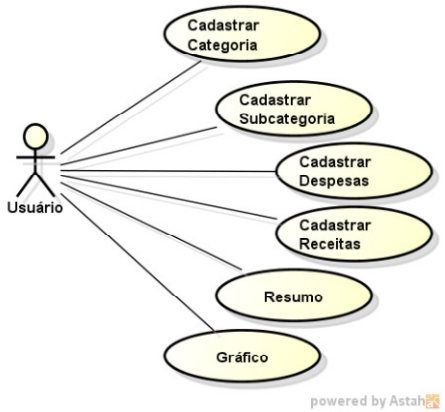

Figura 6 – Diagrama de casos de uso.
Fonte: Autoria própria.

Algumas das funcionalidades do sistema foram definidas após o estudo dos sistemas semelhantes, sendo elas a geração de resumo e apresentação de gráficos, os quais foram apresentados na seção 2.2.

Na tela de lançamento será possível cadastrar a categoria e a subcategoria referente ao lançamento. Após o cadastro, o usuário poderá realizar o lançamento das receitas e despesas. As informações lançadas poderão ser acompanhadas na tela de resumo e de gráfico.

Na Figura 7 está o diagrama de atividade do aplicativo, onde inicialmente o usuário pode cadastrar uma categoria ou subcategoria, desta forma será permitido o lançamento das receitas e despesas. Após o lançamento, os dados lançados poderão ser consultados na tela de resumo ou pela tela de gráfico.

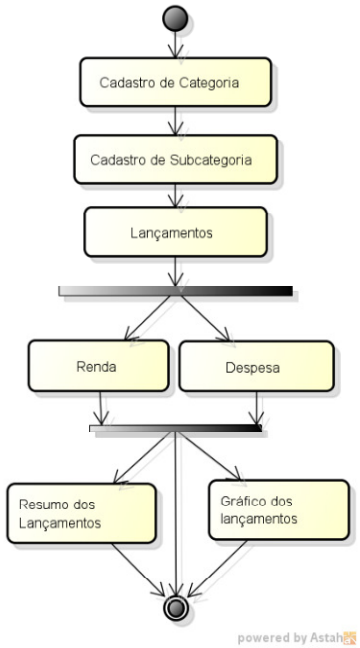

Figura 7 – Diagrama de Atividade.
Fonte: Autoria própria.

O aplicativo, apesar de relativamente simples, possui uma organização de classes bem aprimorada, facilitando futuras inclusões de funcionalidades ou modificação do sistema. O Diagrama de Classe com todas as classes do aplicativo é apresentado na Figura 8.

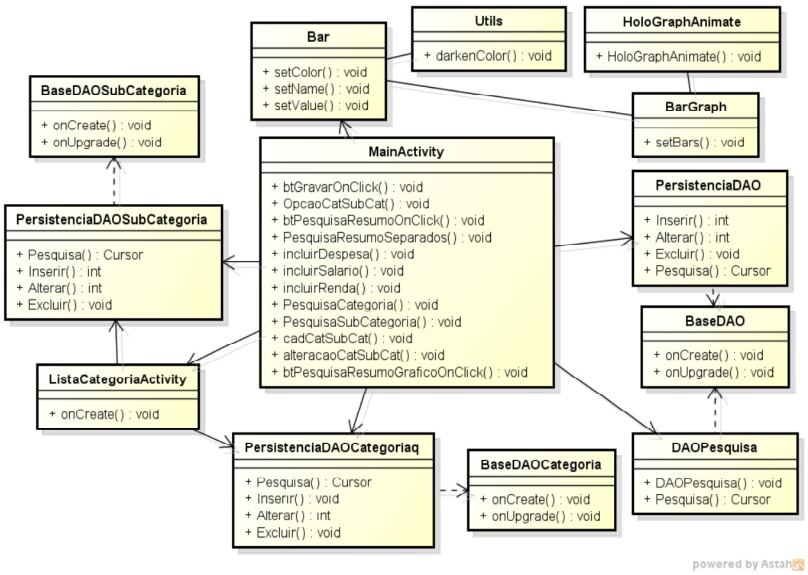

Figura 8 – Diagrama de Classe.
Fonte: Autoria própria.

A principal classe utilizada no aplicativo é a MainActivity, está classe que manipula a parte visual da interface gráfica principal, chamando as rotinas de gravação e de validação. Esta classe herda da ActionBarActivity (Classe do próprio Android) que permite o controle dos componentes visuais.

A conexão de banco de dados e a criação de tabelas é realizada pelas classes BaseDAO, BaseDAOCategoria e BaseDAOSubCategoria, estas por sua vez são responsáveis pelo controle de conexão com o banco e a manutenção das tabelas. O Diagrama de Entidade e Relacionamento do aplicativo é apresentado na Figura 9.

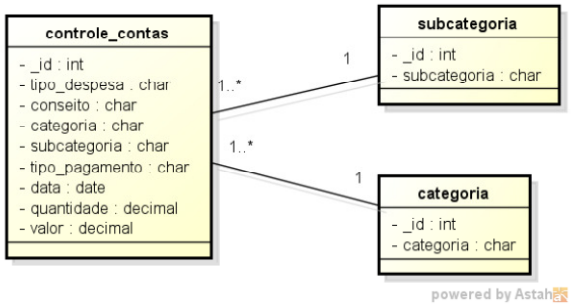

Figura 9 – Diagrama de Entidade e Relacionamento.
Fonte: Autoria própria.

As tabelas do banco de dados são: controle_contas, categoria e subcategoria. A tabela “controle_contas” possui relacionamento com as tabelas subcategoria e categoria.

As tabelas categoria e subcategoria possuem a finalidade de classificação sobre os gastos, por exemplo, vamos supor que uma pessoa possui dois carros e os dois necessitem efetuar a manutenção preventiva e são gerados 2 valores diferentes. Ter dois lançamentos com a mesma finalidade porém em valores e/ou datas diferentes poderia deixar a compreensão um tanto vaga.

Para evitar esse tipo de problema uma segunda categoria faz-se necessária, ou seja, agora o usuário pode lançar categoria “Manutenção Preventiva”, subcategoria “Veículo X”, ou vice versa. Como podem existir muitas categorias subcategorias, e ainda várias combinações entre elas, as duas tabelas não possuem relacionamento, o que dá maior versatilidade ao software. Todas as tabelas apresentadas são persistidas no banco de dados SQLite.

Para a persistência dos dados, foram utilizadas três tabelas no banco de dados SQLite, cuja as estruturas são apresentas nas Tabelas 2, 3 e 4.

Tabela: Controle_Contas	
Nome do Campo	**Tipo de Dado**
_id	Integer
Tipo_despesa	Text
Conceito	Text
Categoria	Text
Subcategoria	Text
Tipo_pagamento	Text
Data	Date
Quantidade	Decimal
Valor	Decimal

Quadro 2 – Estrutura da tabela Controle de Contas do banco de dados.
Fonte: Autoria própria.

Tabela: Categoria	
Nome do Campo	**Tipo de Dado**
_id	Integer
Categoria	Text

Quadro 3 – Estrutura da tabela Categoria do banco de dados.
Fonte: Autoria própria.

Tabela: Subcategoria	
Nome do Campo	**Tipo de Dado**
_id	Integer
Subcategoria	Text

Quadro 4 – Estrutura da tabela Subcategoria do banco de dados.
Fonte: Autoria própria.

4.2 APLICAÇÃO DESENVOLVIDA PARA DESPOSITIVOS MÓVEIS

Após realizada a análise e definida a estrutura de tabela para persistência dos dados, o passo seguinte foi o desenvolvimento do aplicativo.

Para desenvolvimento do sistema móvel foi utilizada a versão 2.2 do Android, sendo esta chamada de “Froyo”. A escolha desta versão aconteceu por ser a disponível no *smartphones* do autor do trabalho, assim como a maioria dos *smartphones* comercializados hoje com o sistema Android são compatíveis com esta versão. A compatibilidade de versões se dá de forma retroativa, ou seja, um aplicativo desenvolvido para Android 2.2 roda em todos os dispositivos com Android 2.2 ou superior. O Quadro 5 apresenta todas as versões do Android disponíveis no mercado até o fechamento deste texto, bem como onde se encontra a versão 2.2 utilizada no presente trabalho.

Plataforma	Codinome	Versão API
Android 1.0	Astro	1
Android 1.1	Astro	2
Android 1.5	Cupcake	3
Android 1.6	Donut	4
Android 2.0	Eclair	5
Android 2.0.1	Eclair	6
Android 2.1	Eclair	7
Android (2.2), (2.2.1), (2.2.2), (2.2.3)	Froyo	8
Android 2.3	Gingerbread	9
Android (2.3.3), (2.3.4), (2.3.5), (2.3.6), (2.3.7)	Gingerbread	10
Android 3.0	Honeycomb	11
Android 3.1	Honeycomb	12
Android (3.2), (3.2.1), (3.2.2), (3.2.6)	Honeycomb	13
Android (4.0.1), (4.0.2)	Ice Cream Sandwich	14
Android (4.0.3), (4.0.4)	Ice Cream Sandwich	15
Android 4.1.1	JellyBean	14
Android 4.1.2	JellyBean	16
Android (4.2), (4.2.1), (4.2.2)	JellyBean	17
Android 4.3	JellyBean	18
Android 4.4	KitKat	19

Quadro 5 – Versão do Android.
Fonte: Autoria própria.

O smartphone utilizado nos testes, durante o desenvolvimento é o Samsung Galaxy Pocket, apresentado na Figura 10, que possui todos os recursos necessários para o desenvolvimento do aplicativo proposto neste trabalho.

Figura 10 – *Smartphone* Samsung galaxypocket.
Fonte: Samsung, 2014.

Para o desenvolvimento foi utilizada a IDE Eclipse com o *plugin* ADT, o qual fornece um ambiente simples para o desenvolvimento da interface visual do aplicativo.

Na Figura 11 é apresentada uma prévia do ambiente para criação das interfaces visuais do menu principal e de cada tela do aplicativo. Foi utilizado um assistente visual para criação das telas. Através deste, bastou arrastar os componentes que estão na paleta de componentes para a área disponível no *layout* da tela e organizá-los de acordo com a necessidade, enquanto o código XML do *layout* é gerado automaticamente.

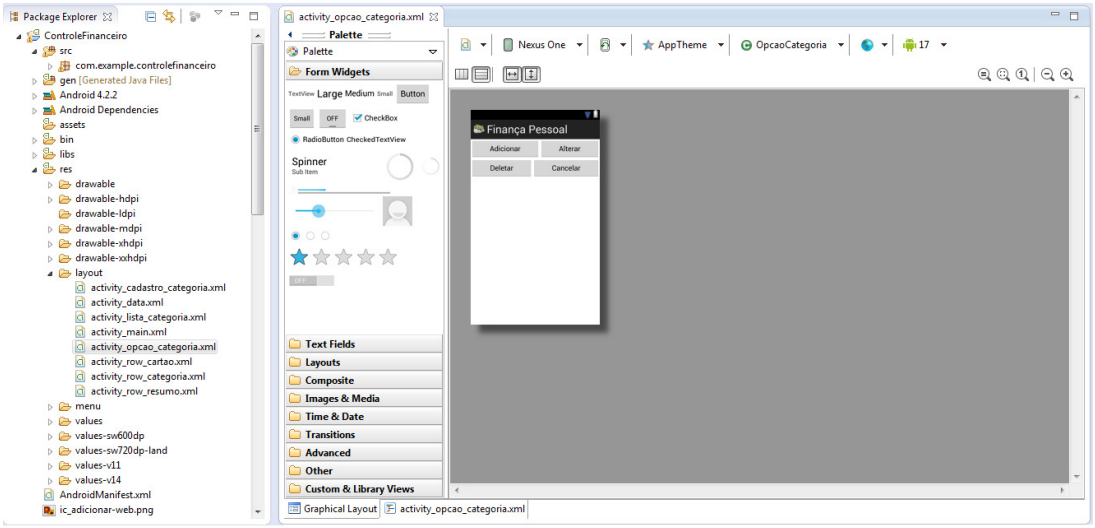

Figura 11 – *Layout* XML do Android no Eclipse.
Fonte: Autoria própria.

4.3 DESENVOLVIMENTO DE APLICAÇÃO MÓVEL

Com o aplicativo desenvolvido será possível fazer o controle de despesas e receitas, ao iniciar o aplicativo a primeira tela apresentada será a tela de lançamento, está tela permitira fazer os lançamentos de despesas e receitas, com isso permitindo ser feita a pesquisa dos lançamentos na tela de resumo, nesta tela será apresentado os lançamentos feitos, na tela de gráfico pode ser feita a pesquisa dos lançamentos e apresentado em gráficos.

Ao iniciar o aplicativo, a primeira tela é a tela de lançamento, apresentada na Figura 12. Esta tela será utilizada para os lançamentos de receitas e despesas.

Figura 12 – Tela de lançamento.
Fonte: Autoria própria.

Se durante o lançamento o usuário identificar a necessidade de uma categoria ou subcategoria ainda não cadastrada, o mesmo pode clicar no botão + (mais), ao lado destes campos, o que apresenta uma tela para manutenção de categorias e subcategorias, conforme Figura 13.

Figura 13 – Tela de alteração de categoria e subcategoria.
Fonte: Autoria própria.

Ao clicar sobre a opção adicionar, a tela referente a Figura 14 é apresentada, solicitando um texto para a categoria ou subcategoria. Para o cadastro, precisa apenas digitar a descrição e clicar no botão Salvar.

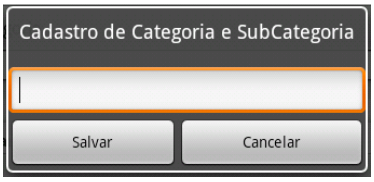

Figura 14 – Tela de cadastro de categoria e subcategoria.
Fonte: Autoria própria.

Quando o usuário clicar no campo Data de Lançamento, um assistente para digitação de datas é apresentado. Este assistente auxilia na digitação de uma data, evitando erros com datas inconsistentes, conforme apresentado na Figura 15.

Figura 15 – Tela de seleção de data.
Fonte: Autoria própria.

Os demais campos da tela são: Tipo de lançamento, para informar se é despesa ou receita, quantidade que é o valor monetário referente ao lançamento, Conceitos que é a descrição do lançamento, e um *check box* para informar se a compra foi realizada com cartão.

Após o lançamento, a tela fica com alguns campos em branco e outros com valores informados para um novo lançamento.

Ao clicar na aba Resumo, é apresentada a tela para uma visualização rápida do que foi lançado no sistema, como apresentado na Figura 16. Nesta tela, o usuário terá uma prévia das despesas e receitas lançadas, permitindo também um filtro data os campos Data Inicial e Data Final.

Figura 16 – Tela de resumo dos lançamentos.
Fonte: Autoria própria.

Ao clicar na aba Gráfico, é apresentado um gráfico de barra, conforme a Figura 17. Nesta tela o usuário poderá acompanhar as movimentações feitas por um gráfico facilitando a visualização de seus lançamentos, cada coluna representa uma informação (receitas ou despesas).

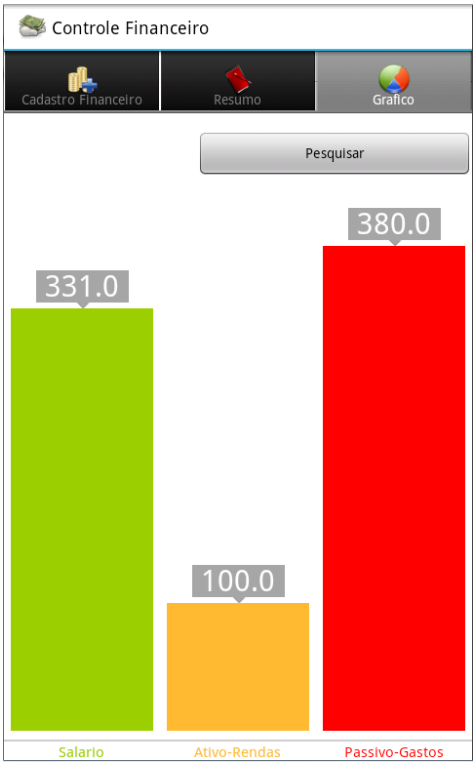

Figura 17 – Tela de gráfico.
Fonte: Autoria própria.

Como observado ao longo de todas as telas do aplicativo, o maior desafio no desenvolvimento para dispositivos móveis e torná-lo rápido e eficiente, uma vez que o perfil dos usuários desta plataforma é a agilidade, já que estes usam os dispositivos esporadicamente, e quando o necessitam, precisam de facilidade e rapidez, já que o lançamento será feito no dia a dia, entre uma compra e outra, na padaria, na loja.

Assim, todas as funcionalidades foram adicionadas na tela principal, a poucos cliques de distância uma da outra. Os componentes visuais também tentam ser auto explicativos, já que não se costuma desenvolver manuais para sistemas móbile.

4.4 PRINCIPAIS CÓDIGOS UTILIZADOS NO SISTEMA

Na Listagem 1 está o código XML em que é feito o *layout* das telas do Android. Esta listagem faz referência a tela do gráfico no aplicativo.

Listagem 1 - Código XML da tela de gráficos.
Fonte: Autoria própria.

```
1.  <LinearLayout android:id="@+id/llTab4"
2.          android:layout_width="match_parent"
3.          android:layout_height="match_parent"
4.          android:orientation="vertical">
5.          <TextView
6.                  android:layout_width="fill_parent"
7.                  android:layout_height="wrap_content"
8.                  android:text=""/>
9.          <LinearLayout android:layout_width="match_parent"
10.                 android:layout_height="match_parent"
11.                 android:orientation="horizontal">
12.                 <TextView
13.                         android:layout_width="wrap_content"
14.                         android:layout_height="wrap_content"
15.                         android:text=""
16.                         android:layout_weight="40"/>
17.                 <Button
18.                         android:id="@+id/btPesquisaResumoGrafico"
19.                         android:layout_width="wrap_content"
20.                         android:layout_height="wrap_content"
21.                         android:layout_weight="40"
22.                         android:text="@string/pesquisa"
23.                         android:onClick="btPesquisaResumoGraficoOnClick"/>
24.         </LinearLayout>
25.         <TextView
26.                 android:layout_width="fill_parent"
27.                 android:layout_height="wrap_content"
28.                 android:text=""/>
29.
30.         <LinearLayout android:layout_width="wrap_content"
31.                 android:layout_height="wrap_content"
32.                 android:orientation="vertical">
33.
34.                 <br.com.controlefinanceiro.holographlibrary.BarGraph
35.                         android:layout_width="match_parent"
36.                         android:layout_height="wrap_content"
37.                         android:id="@+id/graph"/>
38.         </LinearLayout>
39. </LinearLayout>
```

Usando o XML para o desenvolvimento do *layout* do aplicativo, fica mais simples e rápido a criação de telas, com a vantagem de separar o código referente a interface visual do aplicativo (.xml) do código referente a lógica de negócio (.java).

Como observado na Listagem 1, ao longo da interface existem vários componentes visuais, sendo estes identificados pelo parâmetro android:id. Também foram usados gerenciadores de *layout* LinearLayout para organizar os componentes na tela, alguns deles verticais (os componentes são organizados um abaixo do outro), outros horizontais (componentes organizados um ao lado do outro).

Para o tratamento de clique dos botões, a propriedade android:onClick é usada, e faz referência há um método específico dentro da classe Java, que será tratado quando o usuário clicar sobre o botão.

O componente holographlibrary.BarGraph (linha 35 da Listagem 1), é responsável por apresentar o gráfico na tela, a criação deste componente foi possível graças a utilização do *Framework* HoloGraphLibrary.

Na Listagem 2 é apresentada a classe responsável pela conexão com o banco de dados, utilizando a extensão da classe SQLiteOpenHelper (linha 1 da Listagem 2), que gerencia versões de banco utilizada caso exista a necessidade de atualizar o aplicativo.

Listagem 2 - Código da classe BaseDAO.
Fonte: Autoria própria.

```
1.  public class BaseDAO extends SQLiteOpenHelper{
2.   private static final String DATABASE_NAME = "controle_contas.db";
3.  private static final int DATABASE_VERSION = 1;
4.  private String sqlCreate ="CREATE TABLE IF NOT EXISTS controle_contas (_id
    INTEGER                PRIMARY KEY AUTOINCREMENT NOT NULL, tipo_despesa TEXT,
    conseito TEXT," +
                           " categoria TEXT, subcategoria TEXT, tipo_pagamento TEXT,
                           data DATE, quantidade REAL, valor DECIMAL);";


5.  public BaseDAO(Context context) {
6.
7.        super(context, DATABASE_NAME, null, DATABASE_VERSION);
8.
9.  }
10.
11. @Override
12. public void onCreate(SQLiteDatabase database) {
13.
14. database.execSQL(sqlCreate);
15.
16. }
17.
18. @Override
19. public void onUpgrade(SQLiteDatabase db, intoldVersion, intnewVersion) {
20. db.execSQL("DROP TABLE IF EXISTS controle_contas");
21. onCreate(db);
22.
23. }
24.
25. }
```

A tabela controle_contas foi criada por uma execução do SQL na classe BaseDAO que pode ser visualizada na linha 4 da Listagem 3.

A mesma padronização foi utilizada na criação das classes BaseDAOCategoria, responsável pela conexão da tabela de categoria, e BaseDAOSubCategoria, responsável pela tabela de subcategoria. Foi criada também a classe BaseDAO para cada tabela do aplicativo.

O código da classe ListaCategoriaActivity, responsável pela apresentação da lista de categoria e subcategoria é apresentada na Listagem 3.

Listagem 3 - Código da class ListaCategoriaActivity.
Fonte: Autoria própria.

```
1.  public class ListaCategoriaActivity extends Activity {
2.
3.          @Override
4.          protected void onCreate(Bundle savedInstanceState) {
5.                  super.onCreate(savedInstanceState);
6.                  setContentView(R.layout.activity_lista_categoria);
7.
8.                  DAOCAT = new PersistenciaDAOCategoria(this);
9.                  DAOSUBCAT = new PersistenciaDAOSubCategoria(this);
10.                 List<String> array = new ArrayList<String>();
11.                 lvListaCategoria = (ListView)
    findViewById(R.id.lvListaCategoria);
12.                 Intent tipoitem = getIntent();
13.                 String tipo = tipoitem.getStringExtra("tipo");
14.
15.                 if (tipo.equals("C")) {
16.                         Cursor consumo = DAOCAT.Pesquisa("categoria", camposcat);
17.                         while (consumo.moveToNext()) {
18.
            array.add(consumo.getString(consumo.getColumnIndex("categoria")));
19.
            array.add(consumo.getString(consumo.getColumnIndex("_id")));
20.                         }
21.
22.                         SimpleCursorAdapterdadosTela = new
    SimpleCursorAdapter(this, R.layout.activity_row_categoria, consumo, camposcat,
    camposTela);
23.                         lvListaCategoria.setAdapter(dadosTela);
24.
25.                 }else{
26.
27.                         Cursor consumo = DAOSUBCAT.Pesquisa("subcategoria",
    campossub);
28.
29.                         while (consumo.moveToNext()) {
30.
            array.add(consumo.getString(consumo.getColumnIndex("subcategoria")));
31.
            array.add(consumo.getString(consumo.getColumnIndex("_id")));
32.                         }
33.
34.                         SimpleCursorAdapterdadosTela = new
    SimpleCursorAdapter(this, R.layout.activity_row_categoria, consumo, campossub,
    camposTela);
35.                         lvListaCategoria.setAdapter(dadosTela);
36.                 }
37.
38.                 lvListaCategoria.setOnItemClickListener(new
    AdapterView.OnItemClickListener() {
39.                         @Override
40.                         public void onItemClick(AdapterView<?> arg0, View arg1,
    int arg2, long arg3) {
41.                                 Intent i = getIntent();
42.                                 i.putExtra("returnid",
    Integer.parseInt(((TextView)
    arg1.findViewById(R.id.tvIdCategoria)).getText().toString()));
43.                                 i.putExtra("descricao", ((TextView)
    arg1.findViewById(R.id.tvCategoria)).getText().toString());
44.                                 setResult(0, i);
45.                                 finish();
46.                         }
47.                 });
48.         }
49. }
```

Na classe MainActivity foi utilizada a classe Cursor (linha 16 da Listagem 4), para recuperar as informações da base de dados, está classe permite manipular os dados retornados pelo método SQLiteDatabase que estão nas classes

PersistenciaDAOCategoria e PersistenciaDAOSubCategoria (Linha 8 e 9 da Listagem 4).

Os dados do Cursor é formatado em um ArrayList, que apresentará os itens na tela pelo SimpleCursorAdapter (linha 22 da Listagem 4). Está classe auxilia na apresentação da lista, podendo informar os dados que serão apresentados e um *layout* que o desenvolvedor pode criar ou utilizar um dos que a API do Android disponibiliza.

Já a classe MainActivity é responsável pela manipulação das telas de lançamento, resumo e gráfico, nesta classe foi utilizado o TabHost que permite criar várias abas em uma mesma tela, conforme Listagem 4.

Listagem 4 - Código da class MainActivity.
Fonte: Autoria própria.

```
1.  public class MainActivity extends ActionBarActivity {
2.          TabHostth;
3.
4.          @Override
5.          protected void onCreate(Bundle savedInstanceState) {
6.                  super.onCreate(savedInstanceState);
7.                  setContentView(R.layout.activity_main);
8.
9.                  th = (TabHost) findViewById(R.id.thPrincipal);
10.                 th.setup();
11.
12.                 TabSpec specs = th.newTabSpec("Tag1");
13.                 specs.setContent(R.id.llTab1);
14.                 specs.setIndicator("CadastroFinanceiro",
    getResources().getDrawable(R.drawable.ic_novo));
15.                 th.addTab(specs);
16.
17.                 specs = th.newTabSpec("Tag2");
18.                 specs.setContent(R.id.llTab2);
19.                 specs.setIndicator("Resumo",
    getResources().getDrawable(R.drawable.ic_pesquisa));
20.                 th.addTab(specs);
21.
22.                 specs = th.newTabSpec("Tag4");
23.                 specs.setContent(R.id.llTab4);
24.                 specs.setIndicator("Grafico",
    getResources().getDrawable(R.drawable.ic_grafico));
25.                 th.addTab(specs);
26.         }
27. }
```

No caso do aplicativo de controle de gastos foi utilizado uma Activity criando três abas: a primeira para o lançamento (linha 12 da Listagem 5), a segunda para resumo (linha 17 da Listagem 5) e a terceira para apresentar os gráficos (linha 22 da Listagem 5), assim a classe que manipula os componentes é a mesma, não havendo necessidade de criar um layout novo para cada tab.

Com a utilização de abas, o aplicativo fica mais rápido, não sendo necessário carregar várias classes e criar vários *layout*, pois será carregado apenas uma vez ao iniciar o aplicativo.

A classe PersistenciaDAO manipula as informações do banco de dados do aplicativo, e é apresentado em detalhes na Listagem 5.

Listagem 5 - Código da classe PersistenciaDAO.
Fonte: Autoria própria.

```
1.  public class PersistenciaDAO {
2.  private SQLiteDatabase database;
3.      private BaseDAOdbBase;
4.      private long retorno;
5.
6.      public PersistenciaDAO(Context context) {
7.         dbBase = new BaseDAO(context);
8.      }
9.      public void open() throws SQLException {
10.         database = dbBase.getWritableDatabase();
11.     }
12.     public void close() {
13.        dbBase.close();
14.     }
15.     public long Inserir(String tabela, ContentValues values) {
16.        open();
17.         retorno = database.insert(tabela, null, values);
18. close();
19.         return retorno;
20.     }
21.     public long Alterar(String tabela, long id, ContentValues values) {
22.        open();
23.        retorno = database.update(tabela, values, "_id = " + String.valueOf(id),
    null);
24.         close();
25.         return retorno;
26.     }
27.
28.     public void Excluir(String tabela, long id, ContentValues values) {
29.        open();
30. database.delete(tabela, "_id = " + String.valueOf(id), null);
31.         close();
32.     }
33.
34.     public Cursor Pesquisa(String tabela, String[] campos, String where){
35.        open();
36.        Cursor consumo;
37.        if (where.equals("")) {
38.                consumo = database.query(tabela, campos, null, null, null, null,
    null);
39.                }else{
40.                        consumo = database.query(tabela, campos, where, null,
    null, null, null);
41.                }
42.        return consumo;
43.     }
44. }
```

A classe BarGraph e HoloGraphLibrary apresentadas na Listagem 6, contém os métodos responsáveis por incluir, alterar, deletar e pesquisar as informações contidas no banco de dados da aplicação, a mesma estrutura foi utilizada para as classes PersistenciaDAOCategoria e PersistenciaDAOSubCategoria.

A classe BarGraph é fornecida pelo *Framework* HoloGraphLibrary, cuja utilização é detalhada na Listagem 6. Está classe é utilizada para a geração de gráfico, para criar as colunas é feito um calculo para definir o tamanho de cada uma (Linha 6 a 18 da Listagem 6), cada coluna terá uma cor diferente da outra podendo assim identificar melhor as colunas (Linha 21 da Listagem 6).

Listagem 6 - Código da classe BarGraph, *Framework* HoloGraphLibrary. Fonte: Autoria própria.

```
1.  canvas.drawColor(Color.TRANSPARENT);
2.  NinePatchDrawable popup =
    (NinePatchDrawable)resources.getDrawable(R.drawable.popup_black);
3.  float maxValue = 0;
4.  float padding = 7 * resources.getDisplayMetrics().density;
5.  float bottomPadding = 30 * resources.getDisplayMetrics().density;
6.  for (final Bar bar : mBars) {
7.          int left = (int) ((padding * 2) * count + padding + barWidth * count);
8.          int right = (int) ((padding * 2) * count + padding + barWidth * (count +
    1));
9.          int width = (int)(defaultBarWidth + (padding *2));
10.         float textWidth = mPaint.measureText(bar.getName());
11.         while (right -left + (padding * LABEL_PADDING_MULTIPLIER) < textWidth) {
12.                 mPaint.setTextSize(mPaint.getTextSize() - 1);
13.                 float newTextWidth = mPaint.measureText(bar.getName());
14.                 if (textWidth == newTextWidth) break;
15.                 textWidth =newTextWidth;
16.         }
17.         count++;
18. }
19. for (final Bar bar : mBars) {
20.         if (isAnimating()){
21.                 if (bar.mAnimateSpecial == ANIMATE_INSERT) {
22.            alpha = ((int) (getAnimatedFractionSafe() * bar.getColorAlpha()));
23.            popupAlpha = ((int) (getAnimatedFractionSafe() * 255));}
24.         else if (bar.mAnimateSpecial == ANIMATE_DELETE){
25.         alpha = ((int) ((1 - getAnimatedFractionSafe()) * bar.getColorAlpha()));
26. popupAlpha = ((int) ((1- getAnimatedFractionSafe()) * 255));}
27.                 else {
28.                         alpha = bar.getColorAlpha();
29.                         popupAlpha = 255;
30.                 }} else {
31.                 mPaint.setAlpha(bar.getColorAlpha());
32.                 popupAlpha = 255;     }
33.         barWidth = barWidths[count];
34.         int top = (int) (getHeight() - bottomPadding
35.                         - (usableHeight * (bar.getValue() / maxValue)));
36.         int right = (int) (left + barWidth );
37.         int bottom = (int) (getHeight() - bottomPadding);
38.         oldright = right;
39.         mBoundsRect.set(left, top, right, bottom);
40.         mPaint.setColor(bar.getColor());
41.         if (isAnimating()) mPaint.setAlpha(alpha);
42.         canvas.drawRect(mBoundsRect, mPaint);
43.         Path p = bar.getPath();
44.         p.reset();
45.         p.addRect(mBoundsRect.left, mBoundsRect.top,mBoundsRect.right,
46.                 mBoundsRect.bottom,Path.Direction.CW);
47.         bar.getRegion().set(mBoundsRect.left, mBoundsRect.top, mBoundsRect.right,
48.                 mBoundsRect.bottom);
49.         if (mShowAxisLabel) {
50.                 mPaint.setColor(bar.getLabelColor());
51.                 mPaint.setTextSize(labelTextSize);
52.                 if (isAnimating()) mPaint.setAlpha(alpha);
53.                 float textWidth = mPaint.measureText(bar.getName());
54. int x = (int) (((mBoundsRect.left + mBoundsRect.right) / 2) - (textWidth / 2));
```

4.5 TESTE DO APLICATIVO

Para desenvolvimento do aplicativo foi utilizado com base um *Smartphone* Samsung galaxy pocket com a versão 2.2 do Android instalado, a fim de realizar testes reais da aplicação.

A execução e depuração do aplicativo podem ser feitas instalando corretamente os drives do *smartphone* fornecidos pela fabricante e habilitando no aparelho a função de depuração USB, sendo esta uma opção disponível em todas as versões do Android. Com os drives instalados, basta configurar a IDE Eclipse para que execute a aplicação usando o aparelho, para configurar o ambiente Eclipse para que seja possível rodar o aplicativo no *smartphone*, na barra de ferramentas principal em *run* acessar *run configurations*, será aberta uma nova janela, selecionar *Android Application* selecionar a aba *Terget* e selecionar a opção *Always Prompt to pick device*, com isso será possível selecionar em que ambiente Android deseja utilizar para rodar o aplicativo.

Para os testes com dispositivos reais foram utilizados os *smartphones*, Galaxy Pocket versão 2.2, Galaxy Y versão 2.3, Galaxy S Duos 2 versão 4.2, Galaxy Gran Duos versão 4.1.

Foi encontrado problema quando executado o aplicativo nos dispositivos, quando o usuário não informava o campo de categoria ou subcategoria, ocasionava erro no aplicativo fechando deixando o usuário sem saber o motivo, foi validado para que não seja permitido salvar com os campos vazios.

Outro problema era com o tamanho da tela de cada aparelho pois cada aparelho tem uma resolução diferente, para isso foi acrescentado uma barra de rolagem para as telas, permitindo assim visualizar todos os componentes movendo a tela para cima ou para baixo.

Durante a realização dos testes foi observado a necessidade de apresentar mensagens de *feedback* ao usuário informando que os campos vazios devem ser preenchidos, sendo os campos de valor, data, categoria e subcategoria.

5 CONCLUSÃO

O sistema desenvolvido tem finalidade de controlar as despesas do dia a dia, facilitando o controle financeiro pessoal dos usuários.

Durante o desenvolvimento do aplicativo, foi realizado um estudo do sistema operacional Android, mostrando o diferencial da plataforma, já que esta é baseada em Linux, sem qualquer restrição por ser gratuito e ter seu código aberto.

Assim foi possível criar um aplicativo que seja simples e prático de trabalhar, possibilitando acesso rápido e pratico aos usuários.

No processo da análise foi feita uma pesquisa com os aplicativos existentes no mercado, com base nisso foi optado pelo aplicativo a ser desenvolvido. Após um estudo de funcionalidades foram definidas as principais, com isso em mente foi possível definir as classes da aplicação e suas funcionalidades.

A codificação do aplicativo foi realizada utilizando os conceitos de orientação a objetos, que é uma das características da linguagem Java, desta forma cada classe tem um papel definido. Para a criação dos gráficos foi optado pela utilização do *framework* HoloGraphLibrary, este processo de codificação foi bem extenso pela falta de conhecimento e experiência com a linguagem e como deveria ser utilizada a ferramenta, qual os parâmetros e funções a utilizar do *framework*.

O processo de apresentação textual foi o que demandou mais esforço durante o desenvolvimento, pois é difícil explicar tudo o que foi feito e os sentimentos até a conclusão do projeto.

5.1 PROBLEMAS ENCONTRADOS

Durante o desenvolvimento do trabalho, foram encontradas algumas dificuldades, em especial, com relação a criar interfaces rápidas e intuitivas, pois a maioria das aplicações exemplos ou material encontrado se refere a aplicações com várias telas, o que não era o foco do presente projeto.

Outra dificuldade foi na mudança do *framework* para geração de gráficos. A primeira versão do aplicativo utilizava o *framework* GraphView, entretanto este

framework era muito limitado. Após uma pesquisa, foi encontrado o *framework* HoloGraphLibrary, que possuía mais funcionalidades, como apresentação de gráficos em linhas e formato de pizza, e assim, toda a lógica de manipulação de gráfico foi alterada.

5.2 TRABALHOS FUTUROS

Para trabalhos futuros sugere-se o desenvolvimento de novos recursos para o aplicativo, como exportar para os dados para Microsoft Excel e melhorar a parte visual, deixando a interface ainda mais amigável.

Outra sugestão de trabalho futuro é a disponibilização do aplicativo no Google Play, para que mais usuários possam baixar e testar.

REFERÊNCIAS

ABLESON, W. F. et al. **Android in Action**. 3ª. Ed. Nova York: Manning, 2012.

AGBA Android Apps. **Google play.** Disponível em: <https://play.google.com/store/apps/details?id=com.agudoApp.salaryApp> Acesso em 19 agosto 2014.

Aon Sistemas. **Google play**. Disponível em: <https://play.google.com/store/apps/details?id=br.com.aon.spending.demo> Acesso em 19 agosto 2014.

CHADE, Jamil. **Estadão Economia.** Disponível em: <http://economia.estadao.com.br/noticias/geral,numero-de-celulares-se-igualara-a-numero-de-habitantes-da-terra-este-ano,183736e> Acesso em 19 agosto 2014.

DIAS, Eduardo, Campos. **Dificuldade Zero**. Disponível em: <http://dificuldadezero360.blogspot.com.br/2014/02/o-atual-mercado-dos-sistemas.html> Acesso em 19 agosto 2014.

DroidNew. Disponível em: < http://droidnew.com/2013/11/07/introducao-ao-android/ /> Acesso em 19 agosto 2014.

Google play. Disponível em: <https://play.google.com/store?hl=pt_BR> Acesso em 19 agosto 2014.

HAMANN, Renan. **TecMundo**. Disponível em: <http://www.tecmundo.com.br/sistema-operacional/60596-ios-android-windows-phone-numeros-gigantes-comparados-infografico.htm> Acessado em 2014.

Ikoa. **Google play**. Disponível em: <https://play.google.com/store/apps/details?id=com.controleFinanceiro> Acesso em 19 agosto 2014.

LECHETA, RICARDO L. **Google Android.** 3ª. Ed. Novatec. 2013.

NADEAU, Daniel. **HoloGraphLibrary**. Disponível em: < https://bitbucket.org/danielnadeau/holographlibrary/wiki/Home > Acesso em 23 agosto 2014.

OKADA, Sabrina. TMV. Disponível em: <http://www.saibaja.com.br/tv-matao/noticia/372-compras-por-impulso-podem-gerar-dividas> Acessado em 2014.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª. Ed. São Paulo: McGraw-Hill do Brasil. 2006.

Samsung. Disponível em: <http://www.samsung.com/br/consumer/cellular-phone/cellular-phone-tablets/smartphones/GT-S5300ZKBZTO> Acesso em 19 agosto 2014.