



UNIVERSIDADE
FEDERAL
DE PERNAMBUCO



UNIVERSIDADE FEDERAL DE PERNAMBUCO

Graduação em Ciência da Computação

DIOGO RODRIGUES CABRAL

**VISUALIZAÇÃO E MANIPULAÇÃO DE DADOS EM DISPOSITIVOS
MÓVEIS**

RECIFE

2015

DIOGO RODRIGUES CABRAL

**VISUALIZAÇÃO E MANIPULAÇÃO DE DADOS EM DISPOSITIVOS
MÓVEIS**

Trabalho de conclusão de curso apresentado à disciplina de TCC para obter o grau de Bacharel em Ciência da Computação sob a orientação do professor Fernando da Fonseca de Souza.

RECIFE

2015

*Não sabendo que era impossível, foi lá e fez
(Jean Cocteau).*

AGRADECIMENTOS

Agradeço a minha família, que mesmo diante de todas as dificuldades nunca deixou de me dar o suporte necessário, a minha namorada Raissa Andrade, que soube me confortar nos momentos de tensão, medo e receio durante a minha vida na universidade, além de me auxiliar na elaboração deste trabalho sempre que eu precisei, aos meus queridos amigos, em especial Bruno Dutra, Camila Nery, Caio César e Ezequiel Matos, que sempre confiaram em meu potencial e me apoiaram e ao meu orientador Fernando da Fonseca pela oportunidade de realizar um trabalho do meu interesse e me dar todas as ferramentas necessárias para a conclusão do mesmo.

RESUMO

A utilização de bancos de dados em aplicações é necessária quando se está referindo ao consumo de informação por parte dos usuários. Atualmente, inúmeros tipos de dados estão disponíveis para que os desenvolvedores tenham mais liberdade em trazer esse conteúdo ao usuário e, por isso, Sistemas de Gerenciamento de Bancos de Dados (SGBD) são essenciais no auxílio à esquematização dessas aplicações. No contexto dos dispositivos móveis, entretanto, existem algumas dificuldades em relação à visualização e à manipulação dos dados de forma intuitiva e amigável no nível de interface de usuário, levando em conta as restrições por se tratar de aparelhos de menores dimensões do que outros computadores. Neste trabalho serão analisados aplicativos para *smartphones* (com foco nos dispositivos que possuem o sistema operacional Android) que tentam se comportar como um SGBD tradicional a fim de estudar suas limitações em aspectos como, por exemplo, usabilidade e tratamento de dados multimídia. Aliado a isso, serão feitos estudos para otimização da interface voltados para o contexto de bancos de dados, com o objetivo de dar mais conforto e naturalidade para o usuário nesse tipo de aplicação. Ao final do estudo, uma ferramenta será elaborada visando resolver algumas das limitações encontradas.

Palavras-chave: Banco de dados, Smartphones, Sistema Android, Interface de usuário.

ABSTRACT

When we are dealing with the consumption of information by users, it's necessary the use of databases in applications. Nowadays, numerous types of data are available so that developers can have more freedom in bringing this content to the user and, therefore, database management systems are essential to help the design of such applications. In the context of mobile devices, however, there are some difficulties with the visualization and manipulation of data in an intuitive and friendly way in terms of user interface. This happens because of the limitations brought by these devices based on the fact that their dimensions are smaller than those present on other computers.

In this work smartphone applications that try to behave as a management database system (focusing on devices with Android operating system) will be analyzed with the purpose of studying their restrictions in aspects such as usability and multimedia data processing. Allied to this, there will be conducted studies to optimize interfaces in the context of databases, in order to give more comfort and easiness to the user in such type of applications. At the end of this study, a tool will be developed aimed at solving some of the limitations encountered.

Keywords: Database, Mobile devices, Android system, User interface.

ÍNDICE DE FIGURAS

Figura 2.1 - Listagem de tabelas do banco de dados (a) e criação de tabela (b)	18
Figura 2.2 - Adição de campo da tabela em processo de criação	19
Figura 2.3 - Tela de exibição ao clicar no botão "Dados"	20
Figura 2.4 - <i>Popup</i> para edição de dados (a) e consultas SQLite (b)	21
Figura 2.5 - Inserção de dados do tipo BLOB (a) e visualização de dados do tipo BLOB (b)	22
Figura 2.6 - Lista de tabelas do banco (a) e criação de nova tabela (b)	24
Figura 2.7 - Visualização de dados	25
Figura 2.8 - Problema com visualização de dado multimídia	25
Figura 2.9 - Tela de edição de dados	26
Figura 2.10 - Filtros de visualização de dados	27
Figura 2.11 - Criação de um banco de dados	28
Figura 2.12 - Listagem de tabelas (a) e suas opções (b)	29
Figura 2.13 - Criação de colunas	30
Figura 2.14 - Listagem de colunas	31
Figura 2.15 - Tela de exibição de dados	31
Figura 2.16 - Criação de novos dados	32
Figura 2.17 - Problema ao tentar utilizar o campo do tipo multimídia	33
Figura 2.18 - Tela de aplicação de filtros para visualização	34
Figura 2.19 - Criação de chave estrangeira	35
Figura 2.20 - Tela inicial do aplicativo	36
Figura 2.21 - Listagem de tabelas	37
Figura 2.22 - Exibição de colunas (a) e botão de inserção de nova coluna (b)	38
Figura 2.23 - Opções para manipulação dos dados	39
Figura 2.24 - Inserção em campos do tipo DateTime (a) e Image (b)	40
Figura 2.25 - Problema com visualização de tabela	41

Figura 2.26 - Inserção de chave estrangeira	42
Figura 3.1 - Diagrama sobre experiência do usuário	46
Figura 3.2 - Exemplo do diagrama de fluxo de utilização	47
Figura 3.3 - Fluxo de utilização para o SQLite Editor	48
Figura 3.4 - Fluxo de utilização para o SQLite Master	48
Figura 3.5 - Fluxo de utilização para o SQLite Magic	49
Figura 3.6 - Fluxo de utilização para o PortoDB	49
Figura 4.1 - Tela de listagem e criação de bancos de dados	57
Figura 4.2 - Telas de listagem e criação de tabelas	58
Figura 4.3 - Opções de alteração para tabelas	58
Figura 4.4 - Listagem de colunas	60
Figura 4.5 - Opções de alterações para colunas	60
Figura 4.6 - Opções durante a criação de colunas	62
Figura 4.7 - Tela de visualização de dados	63
Figura 4.8 - Fluxo de funcionamento de edição de colunas	64
Figura 4.9 - Fluxo de funcionamento da exclusão de colunas	64
Figura 4.10 - Tela de visualização de dados com coluna do tipo BLOB	65
Figura 4.11 - Tela de inserção de dados	67
Figura 4.12 - Tela de inserção de dados com colunas do tipo BLOB	67
Figura 4.13 - Mensagem alertando o usuário sobre problemas na inserção	69
Figura 4.14 - Tela de edição de dados	70
Figura 4.15 - Fluxo de utilização para o Data Handling	71
Figura 5.1 - Resumo do diagrama criado no Capítulo 3 aplicado a todos os aplicativos	74

ÍNDICE DE QUADROS

Quadro 2.1 - Avaliação final do SQLite Editor	22
Quadro 2.2 - Avaliação final do SQLite Master	27
Quadro 2.3 - Avaliação final do SQLite Magic	35
Quadro 2.4 - Avaliação final do PortoDB	43
Quadro 2.5 - Todas as avaliações realizadas no capítulo	43
Quadro 3.1 - Padrão do aplicativo baseado em telas e botões	50
Quadro 4.1 - Avaliação final do Data Handling	70
Quadro 4.2 - Padrão do aplicativo baseado em telas e botões	71
Quadro 5.1 - Todas as avaliações realizadas no trabalho	74
Quadro 5.2 - Padrão de uso dos aplicativos baseados em telas e botões	74

LISTA DE ABREVIACOES

SO	Sistema Operacional
SGBD	Sistema Gerenciador de Banco de Dados
BLOB	<i>Binary Large Object</i>
ISO	<i>International Standard Organization</i>
XML	<i>eXtensible Markup Language</i>

SUMÁRIO

1. INTRODUÇÃO	12
1.1. Objetivo Geral.....	13
1.2. Objetivos específicos	14
1.3. Estrutura do trabalho.....	14
1.3.1. Trabalhos relacionados	15
1.3.2. Análise da interface gráfica e experiência com o usuário.....	15
1.3.3. Data Handling – visualização e manipulação de dados em smartphones ...	15
1.3.4. Conclusão	15
1.3.5. Referências	15
 2. TRABALHOS RELACIONADOS	 16
2.1. SQLite Editor	17
2.2. SQLite Master	23
2.3. SQLite Magic.....	27
2.4. PortoDB	36
2.5. Considerações finais sobre as análises	43
 3. ANÁLISE DA INTERFACE GRÁFICA E EXPERIÊNCIA DO USUÁRIO	 45
3.1. Fluxo de utilização	46
3.2. Exibição de dados.....	50
3.3. Considerações finais	52
 4. DATA HANDLING – VISUALIZAÇÃO E MANIPULAÇÃO DE DADOS DE BD EM SMARTPHONES	 53
4.1. Ideação da solução	53
4.2. Esquema de projeto	55
4.3. Funcionalidades e design.....	56
4.4. Análise da aplicação.....	70
4.5. Considerações finais	71

5. CONCLUSÃO	73
5.1. Limitações.....	75
5.2. Trabalhos futuros	76
REFERÊNCIAS	77

1. INTRODUÇÃO

Nos últimos anos, empresas de consultoria vêm investindo em análises sobre o uso dos sistemas operacionais (SO). Os números obtidos pela Gartner¹ em uma pesquisa sobre o sistema operacional Android no ano de 2013 mostram que ele já era o líder entre os consumidores brasileiros, no comparativo com outras plataformas. A pesquisa ainda apontou que ele estava, na época, presente em 85,1% dos aparelhos vendidos no Brasil. (TECMUNDO, 2014).

Além desse tipo de estudo, outras pesquisas focam na obtenção de dados sobre os desenvolvedores das atuais plataformas. Segundo a pesquisa realizada pela Appfigures², a loja de aplicativos Android (Google Play Store) teve um aumento de 50% na sua quantidade de aplicativos em 2014 (apud BLOG DO ANDROID, 2015).

Observando o número de desenvolvedores que publicaram aplicativos em diferentes distribuidores, é visto um cenário familiar. A comunidade de desenvolvedores da Google Play cresceu tremendamente em 2014, sendo maior que a da Apple pelo terceiro ano consecutivo, chegando a 400 mil diferentes desenvolvedores (APPPFIGURES, 2015, tradução nossa).

Com todas essas informações que apontam para o crescimento da plataforma, é esperado que os dados que são consumidos em *smartphones* continuem aumentando. Sendo assim, é importante que os desenvolvedores desses aplicativos saibam as melhores maneiras de lidar com banco de dados, no objetivo de estabelecer uma relação de familiaridade com este ambiente.

¹ <http://www.gartner.com>

² <https://appfigures.com>

Durante o desenvolvimento de um banco de dados relacional, é importante ter a noção de como o esquema está se comportando não só de maneira conceitual, mas também de maneira visual, para que não seja apenas uma construção abstrata. Para auxiliar nessa análise, existem os sistemas gerenciadores de bancos de dados para linguagens utilizadas em sistemas *web* e *desktop*.

Os sistemas gerenciadores de bancos de dados (SGBD) são uma peça fundamental na infraestrutura de software de qualquer empresa, seja ela de informática ou não. Um SGBD confiável deve apresentar uma série de funcionalidades, tais como: segurança dos dados, consistência, disponibilidade, recuperação de falhas, desempenho, controle de coerência, etc (ZIMBRÃO, s.d.).

A utilização de SGBD para proporcionar essa facilidade ainda é precária no contexto dos dispositivos móveis. Algumas ferramentas são encontradas atualmente para tentar suprir essa necessidade, mas como ainda não há nada consolidado no mercado nessa vertente, é necessário um estudo mais detalhado dessas ferramentas e de suas limitações técnicas e visuais, visando melhorias relevantes para desenvolvedores e empresas que trabalham com a plataforma.

1.1. Objetivo Geral

O objetivo deste trabalho, essencialmente prático, é descobrir as limitações existentes, tanto de interface quanto de funcionalidades relevantes ao usuário, no contexto das aplicações voltadas a visualização e manipulação de dados para o sistema Android³ (mais

³ <https://www.android.com/>

especificamente em sistemas de bancos de dados SQLite⁴, construindo assim o embasamento necessário para trazer melhorias à comunidade de desenvolvedores que fazem uso desse tipo de ferramentas.

1.2. Objetivos específicos

Como objetivos específicos do trabalho têm-se:

- i) Realizar uma análise sobre as principais aplicações disponíveis no mercado de modo a identificar as limitações existentes; e
- ii) Especificar e implementar uma ferramenta visando resolver ou reduzir as limitações encontradas para permitir uma melhora no desenvolvimento de aplicações de visualização de dados em *smartphones* no contexto de banco de dados.

1.3. Estrutura do trabalho

Além deste capítulo, este documento é formado por:

- 1. Trabalhos relacionados;
- 2. Análise da interface gráfica e experiência com o usuário;
- 3. Data Handling – visualização e manipulação de dados em *smartphones*;
- 4. Conclusão; e
- 5. Referências.

⁴ <https://www.sqlite.org/>

1.3.1. Trabalhos relacionados

Avaliará as atuais soluções para visualização e manipulação de dados, com foco em discutir as limitações encontradas. Tal estudo será realizado baseado na utilização dos sistemas, testando suas funcionalidades, tendo em vista as métricas necessárias para que a aplicação seja aceitável no contexto de bancos de dados. Ao final do capítulo, um quadro comparativo entre as avaliações estará presente para que o leitor consiga assimilar todas as informações descritas.

1.3.2. Análise da interface gráfica e experiência com o usuário

Discorrerá sobre formas de otimizar esse tipo de aplicação, fornecendo maior usabilidade e legibilidade para o público alvo, por meio da análise da sua interface gráfica. Essa análise será acrescida de fluxos que destacarão a quantidade de telas e botões necessária para que o usuário consiga executar uma funcionalidade específica. Ao final do capítulo, haverá um quadro mostrando a quantidade de telas e botões diferentes utilizados pelos mesmos sistemas, dando um valor mais realista de etapas visualizadas pelo usuário, para comparar os seus desempenhos.

1.3.3. Data Handling – visualização e manipulação de dados em *smartphones*

Apresentará uma implementação de uma aplicação baseada na melhoria das restrições encontradas, mostrando a forma de utilização da ferramenta e dando destaque as suas funcionalidades e interface gráfica. Além disso, todas as métricas utilizadas para a avaliação dos trabalhos relacionados serão aplicadas, para que possa haver um novo comparativo.

1.3.4. Conclusão

É mostrado um breve resumo do estudo realizado e da arquitetura apresentada. Também serão mostradas as limitações da ferramenta desenvolvida e descritas sugestões de trabalhos futuros.

1.3.5. Referências

Serão listadas as referências bibliográficas utilizadas durante todo o trabalho.

2. TRABALHOS RELACIONADOS

No contexto de visualização e manipulação de dados móveis, é interessante elaborar um estudo sobre as ferramentas atualmente disponíveis para os desenvolvedores de aplicações do sistema operacional Android e, diante de suas funcionalidades e apresentações para o usuário, conhecer seus pontos positivos e negativos. Essa análise tem como finalidade construir uma base conceitual sobre esse contexto, para que se tenham meios de idealizar melhorias e incluí-las na ferramenta que será desenvolvida neste trabalho, visando trazer maiores facilidades para a comunidade de desenvolvedores que utilizam bancos de dados em dispositivos móveis.

Quando falamos na plataforma mobile geralmente não temos em mente utilizar uma ferramenta que permita a digitação de comandos SQL. O motivo disso é que, em geral, a digitação de grandes quantidades de caracteres não é muito adequada nas plataformas mobile, sejam *smartphones* ou *tablets*. Contudo, em algumas situações de emergência pode ser necessário realizar uma consulta simples ou mesmo checar por algum dado no servidor em caráter extraordinário. E aí é que entram algumas aplicações mobile para salvar o dia do DBA (PICHILIANI, 2013).

A escolha dos aplicativos a serem analisados foi feita baseada em critérios como: ser gratuito (pois foi considerado que assim tem maior alcance de usuários), ter grande quantidade de *downloads*, grande quantidade de votos positivos na loja de aplicativos Android e funcionalidades descritas pelo desenvolvedor.

Foram escolhidos os seguintes aplicativos: SQLite Editor⁵, SQLite Master⁶, SQLite Magic⁷ e PortoDB⁸. Esses aplicativos serão analisados a seguir, a partir dos seguintes critérios: interface intuitiva, suporte a algumas restrições e configurações básicas de bancos de dados, exibição e manipulação de todos os tipos de dados disponíveis na aplicação e utilização de dados multimídia. Para todos os critérios será estabelecida uma métrica atribuindo valores de 1 a 5, sendo 1 a nota mais baixa na categoria e 5 a mais alta.

⁵ https://play.google.com/store/apps/details?id=dk.andersen.asqlitemanager&hl=pt_BR

⁶ https://play.google.com/store/apps/details?id=com.dundastech.sqlitemasterlight&hl=pt_BR

⁷ https://play.google.com/store/apps/details?id=air.SQLite.Magic&hl=pt_BR

⁸ https://play.google.com/store/apps/details?id=com.portofarina.portodb&hl=pt_BR

Para as restrições e configurações básicas, foram realizados os seguintes testes, separados por tópico:

1. Chave primária
 1. Criar tabela sem chave primária
 2. Criar tabela com duas chaves primárias
 3. Criar tabela com chave primária composta
2. Chave estrangeira
 1. Criar coluna com chave estrangeira
 2. Inserir linha com chave estrangeira sem referência
 3. Editar linha para uma chave estrangeira sem referência
 4. Remover uma linha que é referenciada por outra tabela
3. Campos auto-incrementais
 1. Alterar um dado auto-incremental
 2. Inserir dados num campo auto-incremental

2.1. SQLite Editor

O SQLite Editor, desenvolvido pela WeaveBytes⁹, é uma ferramenta que permite consultar e manipular bancos de dados já existentes no sistema (criados ou não por meio do aplicativo). Além disso, ele permite ao desenvolvedor a criação de um novo banco de dados no qual, posteriormente, também podem ser aplicadas funcionalidades citadas anteriormente. Na Google Play Store¹⁰, este aplicativo está com a quantidade de *downloads* estimada em um valor entre 100.000 e 500.000 e possui 709 avaliações, com uma nota média de 3.2 (em uma escala de 1 a 5).

Para a criação de um banco de dados no sistema, ao clicar no botão destinado a esta funcionalidade (que está bem visível na página principal), o aplicativo fornece a opção de

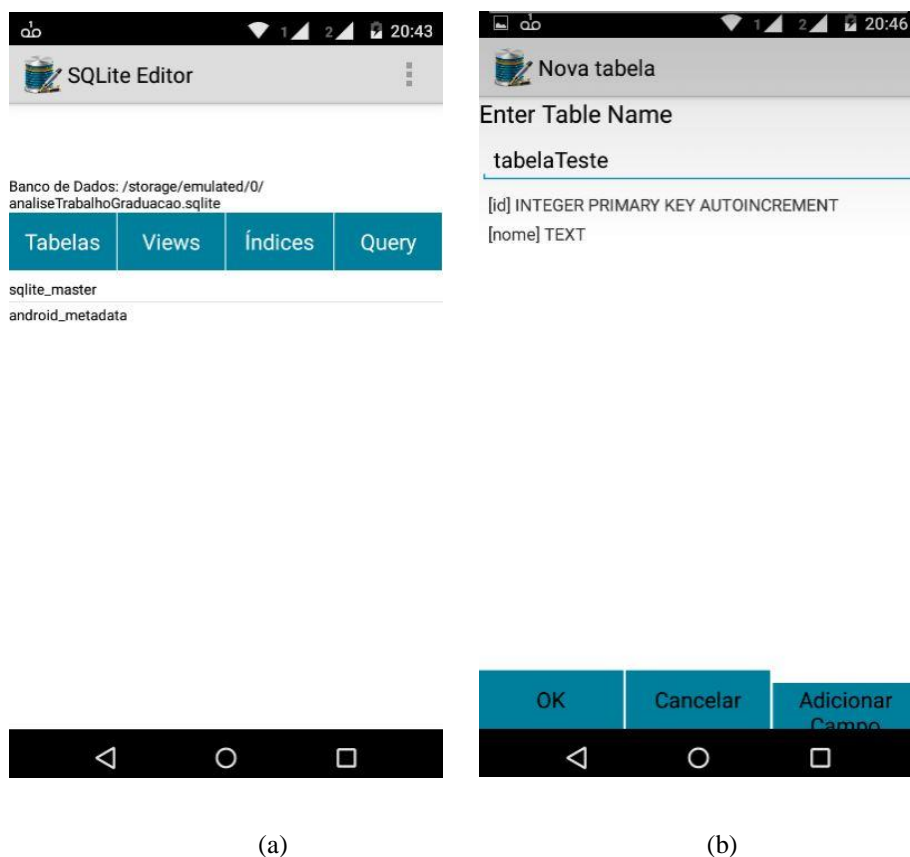
⁹ <http://www.weavebytes.com/>

¹⁰ https://play.google.com/store?hl=pt_BR

escolha do nome do banco e exibe em qual diretório ele vai ser salvo ao clicar no botão. Os bancos criados pelo SQLite Editor ficam em uma pasta comum aos outros aplicativos do *smartphone*, o que permite que eles também possam fazer uso dos mesmos.

Assim que o banco é criado, o sistema exibe as tabelas já existentes (como mostra a Figura 2.1), ou seja, as tabelas criadas automaticamente pelo Android. Para criar tabelas, o menu superior deve ser acessado, o que deixa esta funcionalidade fora do campo de visão do usuário. Quando a tela de criação é acessada, pode-se atribuir um nome para a tabela e confirmar para salvá-la, mas o sistema acusa um erro por ainda não existir ao menos uma coluna na tabela. Na mesma tela, existe um fluxo para a adição de colunas e assim conseguir concluir o processo descrito anteriormente.

Figura 2.1- Listagem de tabelas do banco de dados (a) e criação de tabela (b).



Fonte: SQLite Editor

A aplicação fornece algumas opções para a construção de colunas, como mostra a Figura 2.2, de forma mais consistente, como a possibilidade de escolher o tipo dela, indicar necessidade de ser chave primária (e assim podendo habilitar a coluna auto-incremental e decrescente), não nula e única. Por fim, pode ser escrita, caso necessário, uma coluna relativa a uma chave estrangeira e sua referência para a tabela externa. Até então, alguns problemas

como a falta de exibição de tabelas e colunas existentes no momento da criação da chave estrangeira, e a dificuldade em utilizar chaves primárias simples e compostas, já foram notados. O sistema permite a criação de tabelas sem a indicação de qualquer campo sendo chave primária e não dá a opção de criação de chave composta. Ao tentar inserir duas chaves primárias, como descrito no Teste 1.2. proposto, o sistema corretamente exibe um erro indicando a impossibilidade de executar tal tarefa.

Figura 2.2- Adição de campo da tabela em processo de criação.



The screenshot shows the SQLite Editor application interface. At the top, there's a status bar with icons for signal, battery, and time (20:45). Below the title bar, the app name 'SQLite Editor' is displayed. The main form has several fields: 'Name' with the value 'id', 'Type' with a dropdown menu showing 'INTEGER', 'Not Null' with an unchecked checkbox, 'Unique' with an unchecked checkbox, 'PK' with a checked checkbox, 'DESC' with an unchecked checkbox, 'AutoInc' with an unchecked checkbox, 'Default' with a text input field containing 'Enter default value', 'FK Table' with a text input field containing 'Enter FK table', and 'FK Field' with a text input field containing 'Enter FK field'. At the bottom, there are two buttons: 'OK' and 'Cancelar'. The bottom of the screen shows the Android navigation bar with back, home, and recent apps icons.

Fonte: SQLite Editor.

O processo de manipulação de dados inicia ao escolher uma tabela específica que se deseja trabalhar, na listagem de tabelas. Nela é possível inserir novos dados e editar ou remover dados existentes. As funcionalidades se dividem entre botões na tela e botões no menu superior, o que interrompe a continuidade da experiência do usuário durante a utilização.

Para as operações básicas de inserção e edição dos dados, o aplicativo apresenta, de forma não intuitiva, as opções no botão “Dados”. Ao clicar nele, o usuário pode selecionar a linha que deseja editar e assim alterar ou apagar o dado escolhido, como mostra a Figura 2.3.

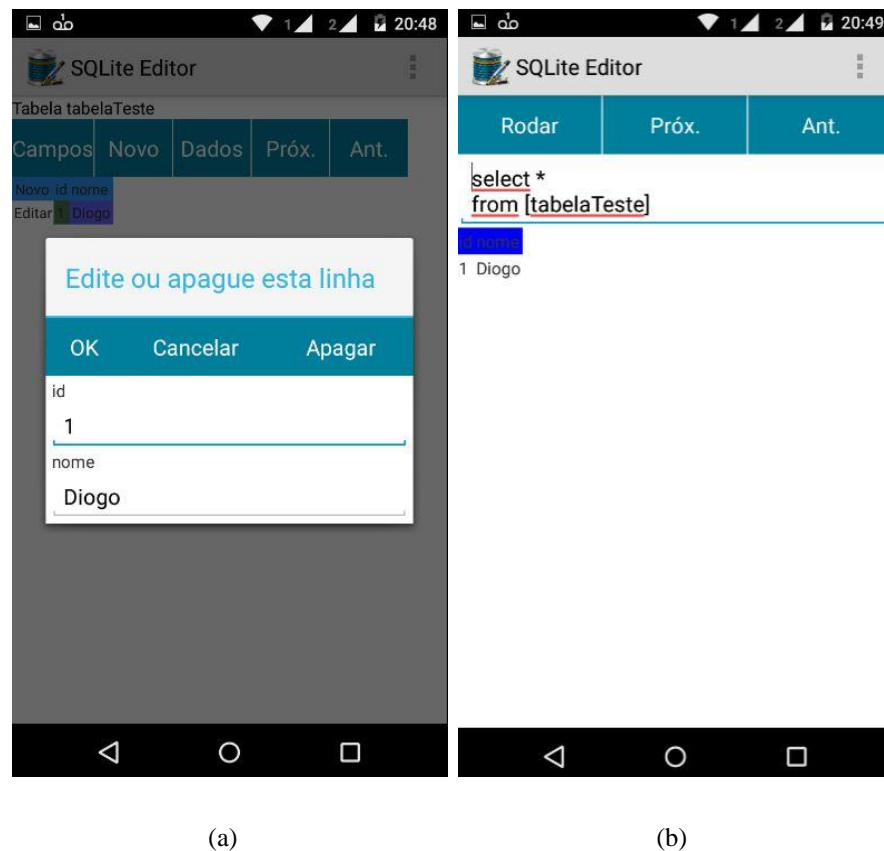
Figura 2.3- Tela de exibição ao clicar no botão “Dados”.



Fonte: SQLite Editor.

Com base na lista de restrições e configurações básicas previamente discutidas, os testes 2.1., 2.2., 2.3., 2.4., 3.1. e 3.2. foram realizados nessa etapa do fluxo do aplicativo. Para o Teste 2.1., o resultado foi satisfatório, pois a tarefa de criação de uma chave estrangeira foi concluída com sucesso. Porém, os resultados dos testes 2.2., 2.3. e 2.4. foram insatisfatórios, pois não foi encontrado nenhum tipo de controle de segurança para a integridade dos dados em relação ao esquema montado previamente e, portanto, foi permitido realizar tais comandos, mesmo que incorretos.

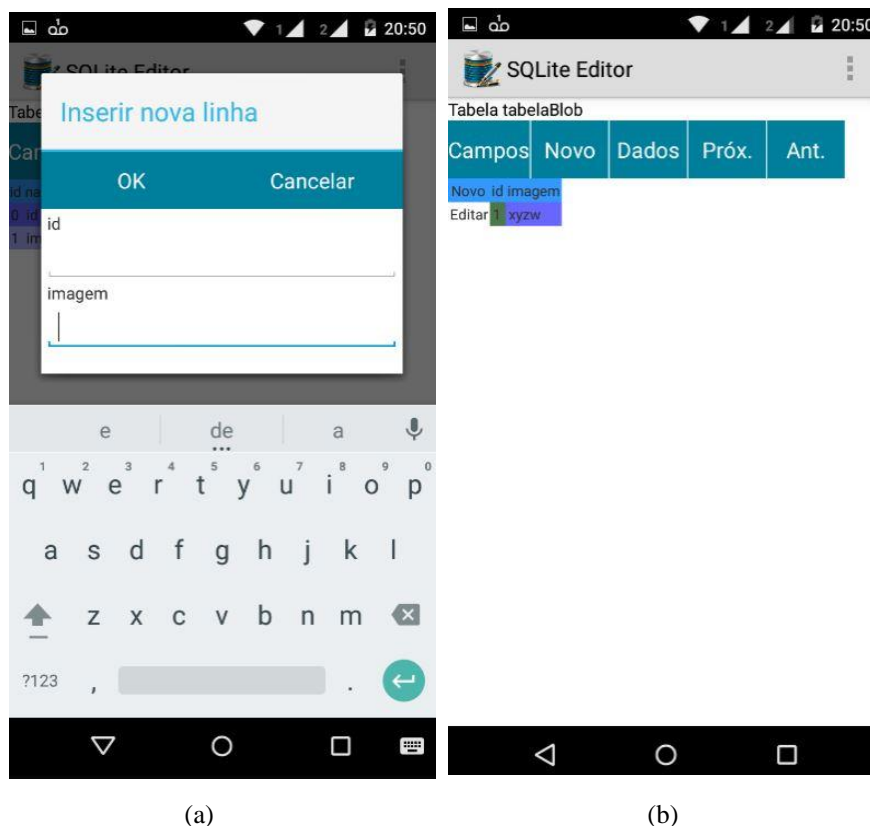
Ainda se tratando da manipulação dos dados, existe a opção de utilização de consultas SQLite, auxiliadas por botões para a inserção do tipo de consulta e em quais tabelas serão feitas. As consultas servem não apenas para a criação de tabelas e colunas ou inserções, como também para filtrar os dados que serão exibidos. A cláusula *where* (assim como outros tipos de detalhamento de consulta) fica a critério do usuário, e não foi encontrado nada para auxiliá-lo nessa etapa. A Figura 2.4 ilustra tanto o processo de manipulação de um determinado dado quanto a utilização das consultas SQLite no sistema.

Figura 2.4- *Popup* para edição de dados (a) e consultas SQLite (b).

Fonte: SQLite Editor.

Em relação à manipulação e à visualização de dados do tipo multimídia (ilustrado na Figura 2.5), o SQLite Editor fornece a opção de, no momento da criação de uma coluna da tabela, utilizar o tipo de armazenamento BLOB (*Binary Large Object*), que serve justamente para armazenar arquivos deste tipo. Após a criação da coluna na tabela, foi verificado que o sistema não consegue inserir de forma a tornar verdadeiramente utilizável, nesse contexto, um BLOB. Ele apenas permite a inserção de caracteres textuais no campo destinado ao objeto binário, o que deixa completamente inviável ao usuário que deseja fazer uso desse tipo de armazenamento. Por essa razão, o sistema também não consegue exibir de maneira adequada os dados multimídia, sendo ele o único tipo de dados que possui essa problemática.

Figura 2.5- Inserção de dados do tipo BLOB (a) e visualização de dados do tipo BLOB (b).



Fonte: SQLite Editor.

Torna-se nítido que em relação ao SQLite Editor existe uma necessidade grande de monetização, ainda que o aplicativo seja grátis, pois a todo momento o usuário se depara com propagandas que tomam todo o espaço da tela, obrigando-o a interagir com elas para voltar a usar o sistema. Propagandas exibidas dessa forma também acabam dificultando a experiência do usuário e podem ser tomada como um problema a ser melhorado.

O Quadro 2.1 mostra as notas finais atribuídas ao aplicativo, considerando todos os critérios descritos no início do capítulo.

Quadro 2.1- Avaliação final do SQLite Editor.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
Nota	3	3	4	1

Fonte: O Autor.

2.2. SQLite Master

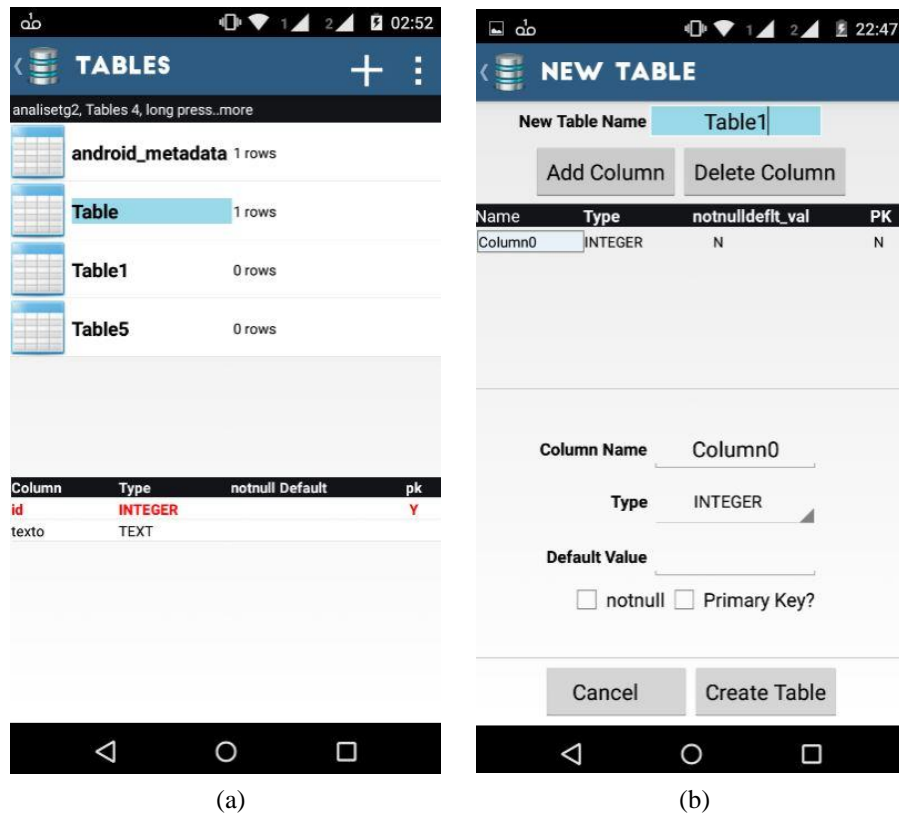
O SQLite Master, desenvolvido pela Amos Mobile¹¹, é uma ferramenta que também permite consultar e manipular bancos de dados novos ou já existentes no sistema. Na Google Play Store, este aplicativo está com a quantidade de *downloads* estimada em um valor entre 10.000 e 50.000 e possui 124 avaliações, com uma nota média de 3.0 (em uma escala de 1 a 5).

Ao acessar o sistema, fica claro que uma navegação pelas pastas do *smartphone* é possível, para que o usuário possa realizar uma busca por repositórios SQLite que deseje manipular. A interface da tela principal é composta por várias abas com outras funcionalidades, que não são pertinentes a este estudo, e um botão na parte superior que possibilita a criação de um novo banco de dados. Durante uma tentativa de criação de um banco, um problema no direcionamento dos arquivos foi encontrado. Por não fornecer uma abstração no caminho do diretório do sistema, o usuário é forçado a reconhecer quais pastas permitem este tipo de funcionalidade. Tentando incluir um banco de dados em uma pasta que não tem permissão para tal, é exibido um erro que não oferece detalhes sobre o que de fato está acontecendo e o motivo pelo qual a operação não pode ser concluída com sucesso.

A Figura 2.6 mostra a forma de exibição das tabelas, assim que o banco é acessado, e as opções disponíveis durante a inserção de uma nova tabela (que incluem, na mesma tela, as opções de criação de colunas). Não foram encontrados meios de criação de chave primária composta, chave estrangeira e nem campos auto-incrementais, o que se torna uma dificuldade para a elaboração de esquemas mais complexos. Além disso, as opções de edição da tabela criada são complexas para um usuário com pouca experiência (pois apresenta a alteração apenas via código SQLite puro) e não são exibidas opções de edição de colunas.

¹¹ <https://play.google.com/store/apps/developer?id=Amos+Mobile>

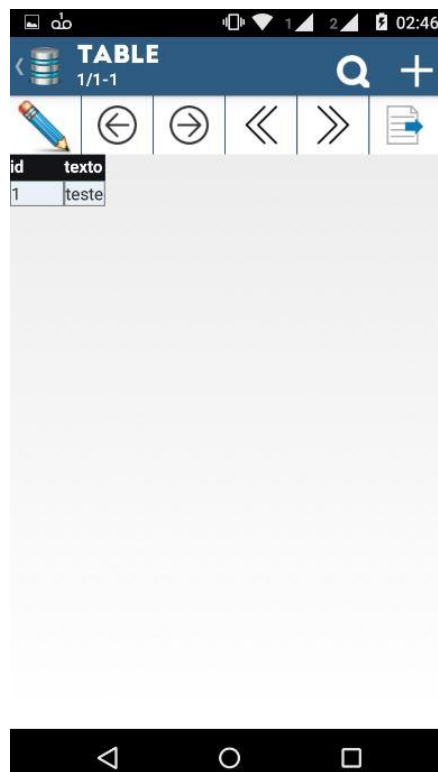
Figura 2.6- Lista de tabelas do banco (a) e criação de nova tabela (b).



Fonte: SQLite Master.

Ao clicar em uma das tabelas, a tela de visualização de dados é exibida. Nela, são encontradas as opções para inserção de uma nova linha e para edição das que foram previamente inseridas (como se pode ver na Figura 2.7). Durante a criação de um novo dado, é interessante observar o fato de a chave primária ser exibida em destaque, para lembrar ao usuário que deve ser preenchida corretamente para a conclusão da funcionalidade. Foi encontrada uma dificuldade ao tentar utilizar dados multimídia, pois o aplicativo não oferece opção de preenchimento para o campo destinado a ele na inserção (mesmo possibilitando a criação de uma coluna do tipo BLOB) e, ao criar a linha com esse dado, a visualização exibe null como consequência disso, como mostra a Figura 2.8. Antes de editar uma linha, é necessário clicar na linha desejada e em seguida apertar no botão “lápiz” para enfim acessar essa funcionalidade. Ao entrar na tela de edição, as opções de excluir e atualizar a linha aparecem em dois botões superiores, como mostra a Figura 2.9.

Figura 2.7- Visualização de dados.



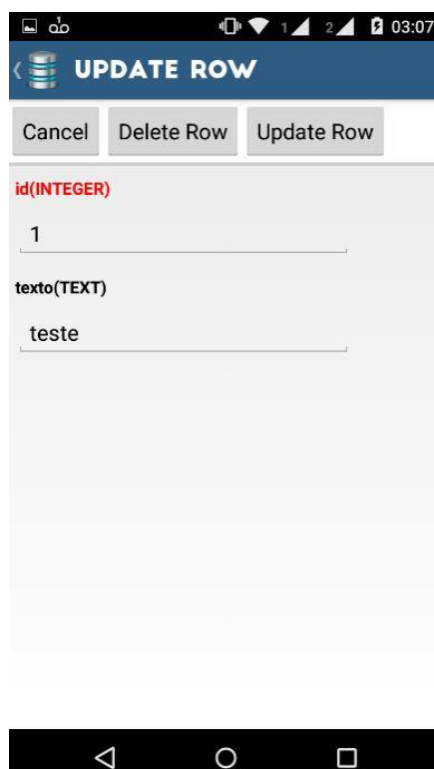
Fonte: SQLite Master.

Figura 2.8- Problema com visualização de dado multimídia.



Fonte: SQLite Master

Figura 2.9- Tela de edição de dados.

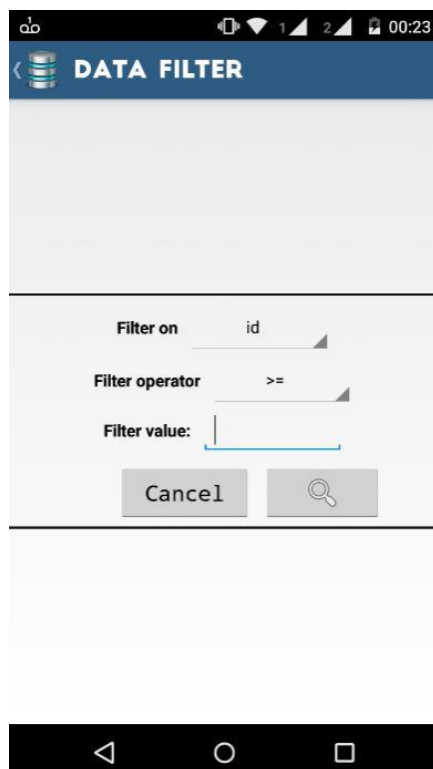


Fonte: SQLite Master.

Após fazer essa análise geral, é possível realizar os testes citados no início do capítulo. O aplicativo permite a criação de tabelas sem chave primária (Teste 1.1), porém exibe um erro ao tentar fazer a inserção de duas colunas sendo chaves primárias (Teste 1.2) e não oferece a opção de criação de chave composta (Teste 1.3), como já mencionado. Os testes seguintes (dos tópicos 2 e 3) não puderam ser realizados pois, como foi dito, o sistema não fornece opções para chave estrangeira e campos auto-incrementais, impossibilitando a execução das tarefas necessárias.

Por fim, é notável que o sistema requer poucos passos para a execução de suas funcionalidades, dando maior conforto para os usuários durante a utilização. Porém, ele deixa a desejar em alguns momentos, como na atualização das tabelas, que ele faz apenas por meio de código SQLite. Outra carência que o sistema apresenta é o filtro para visualização, pois o que existe é limitado às operações lógicas em dados (como mostra a Figura 2.10), dificultando a realização de consultas mais complexas. O Quadro 2.2 mostra as notas finais atribuídas ao aplicativo, considerando todos os critérios descritos no início do capítulo.

Figura 2.10- Filtros de visualização de dados.



Fonte: SQLite Master.

Quadro 2.2- Avaliação final do SQLite Master.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
Nota	3	1	3	1

Fonte: O Autor.

2.3. SQLite Magic

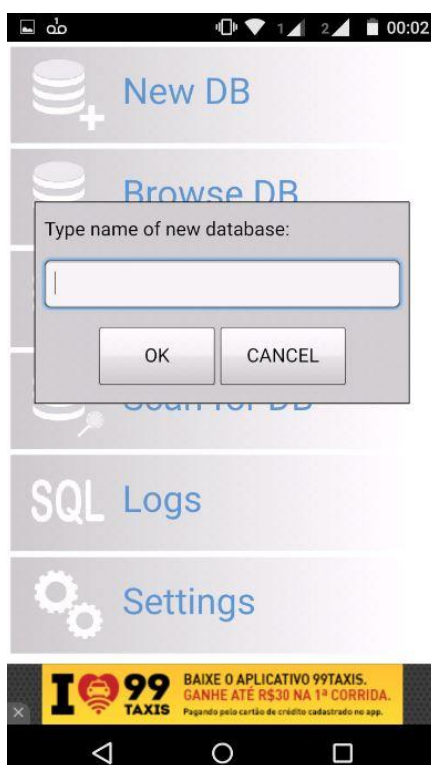
O SQLite Magic, desenvolvido pela iPoint¹², é uma ferramenta que também permite consultar e manipular bancos de dados novos ou já existentes no sistema. Na Google Play

¹² <http://www.ipoint.sk/>

Store, este aplicativo está com a quantidade de *downloads* estimada em um valor entre 10.000 e 50.000 e possui 142 avaliações, com uma nota média de 3.8 (em uma escala de 1 a 5).

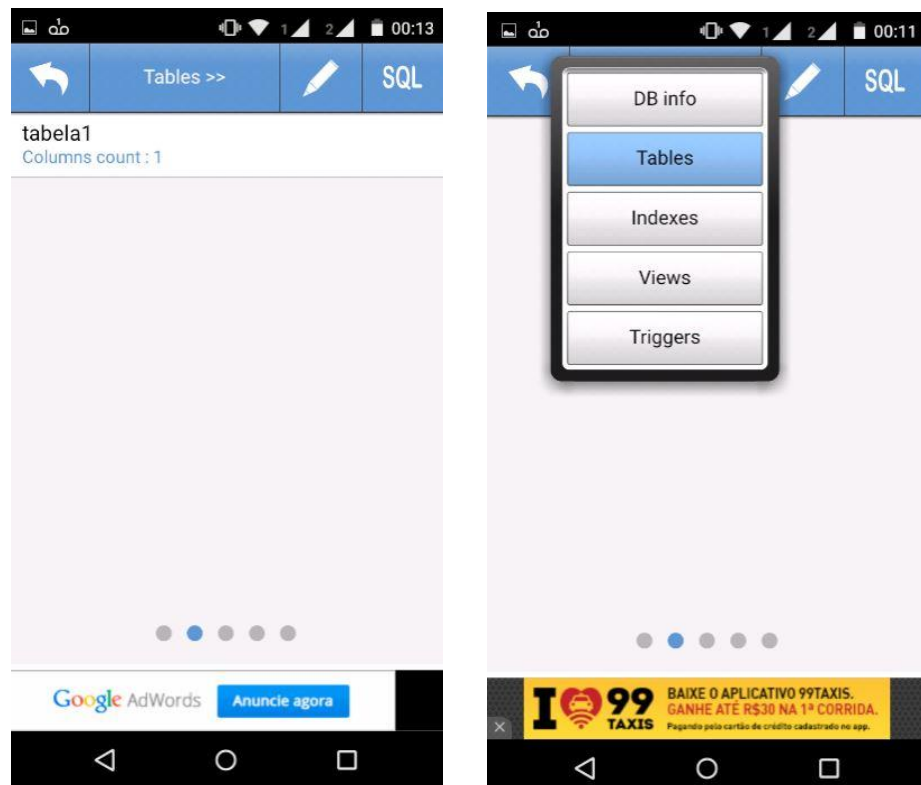
Ao acessar o sistema, uma lista de botões é exibida, mostrando claramente as funcionalidades básicas do sistema. A primeira delas é a de criação de um novo banco de dados, que ao clicar solicita ao usuário a inserção do nome para o novo repositório, como mostra a Figura 2.11. Em seguida, a tela de listagem de tabelas é exibida e nela se pode observar algumas opções como inserção de uma nova tabela e manipulação via código SQLite. É interessante citar a existência de um *menu* na região superior, no qual o usuário pode acessar além das tabelas os *indexes*, *views* e *triggers* criados. A Figura 2.12 mostra todo o conteúdo citado anteriormente.

Figura 2.11- Criação de um banco de dados.



Fonte: SQLite Magic.

Figura 2.12- Listagem de tabelas (a) e suas opções (b).



(a)

(b)

Fonte: SQLite Magic.

Ao clicar no botão de criação de uma nova tabela, um campo pedindo o nome da tabela é exibido, e ao inserir, o usuário é levado diretamente para a criação da primeira coluna. Nesta área, muitas opções ficam disponíveis para uma configuração completa da coluna como mostra a Figura 2.13. Nota-se a presença da possibilidade da criação de campos do tipo BLOB, *Image*, auto-incremental, além de criação de chaves estrangeiras, que serão posteriormente analisadas para a realização dos testes citados no início do capítulo.

Figura 2.13- Criação de colunas.

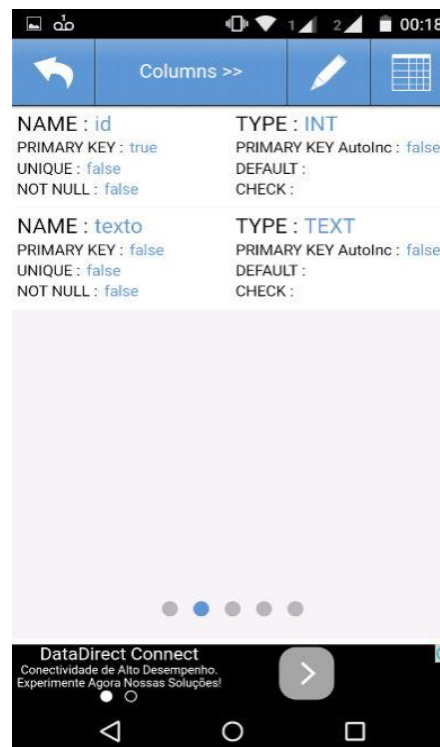
The image displays two screenshots of the SQLite Magic application's 'New Column' screen. The left screenshot shows the initial form with fields for NAME, TYPE (WORD, AUTOINC, AUTOINC INTEGER), PRIMARY KEY, DEFAULT, CONSTRAINT NAME, and NOT NULL. The right screenshot shows the same form with the 'AUTOINC INTEGER' type selected and various constraints (PRIMARY KEY, DEFAULT, NOT NULL, UNIQUE, CHECK, COLLATE, FOREIGN KEY) set to 'NO' or 'YES'.

Fonte: SQLite Magic.

Após a criação da tabela, a listagem das colunas é exibida (Figura 2.14) com várias opções como inserir novas colunas, mostrar os dados já inseridos e, ao clicar em uma coluna listada, editar ou excluir a coluna. Na tela de edição, o usuário volta a ter a sua disposição todas as opções citadas anteriormente para a criação da mesma, com a exceção da troca de nome da coluna, que passa a não ser permitida.

Na tela de exibição de dados (Figura 2.15) aparecem duas *slidebars* que chamam a atenção, pois ocupam todo o espaço da parte inferior da tela. Elas não têm descrição para que o usuário saiba sua utilidade, mas pode-se observar que se trata de uma funcionalidade de redimensionamento da tabela de exibição dos dados em questão. Além disso, existem as opções de criação de novas linhas e de filtrar os dados a serem exibidos. Um problema é encontrado ao tentar visualizar dados multimídia, pois o sistema apenas exibe três pontos nos quais deveria vir o dado desejado.

Figura 2.14- Listagem de colunas.



Fonte: SQLite Magic.

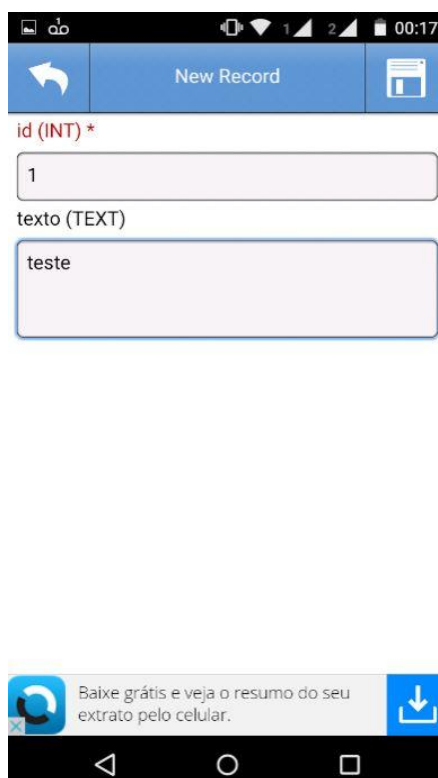
Figura 2.15- Tela de exibição de dados.



Fonte: SQLite Magic.

Na criação de novas linhas (Figura 2.16), a chave primária aparece com destaque para que o usuário seja alertado da obrigatoriedade do preenchimento da mesma. Os demais campos aparecem como campos de texto a serem preenchidos. Ao se tentar utilizar o campo BLOB ou *Image* criados, depara-se com a ausência do mesmo no momento da criação da linha, ou seja, o sistema permite a criação, mas não a manipulação de campos com esse tipo de dado, o que pode se tornar um problema, como mostra a Figura 2.17.

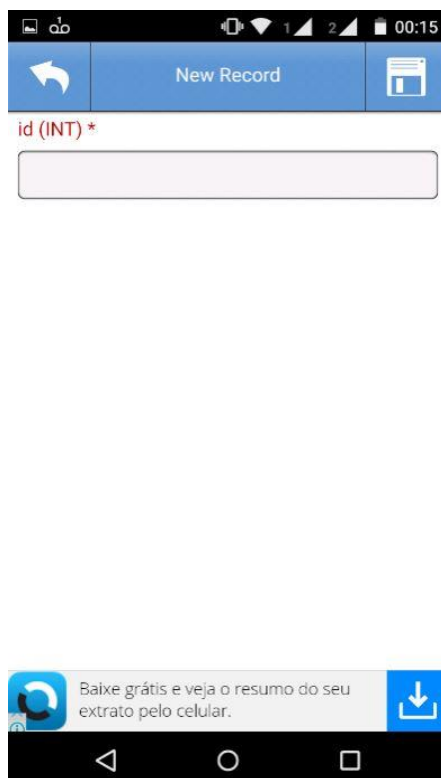
Figura 2.16- Criação de novos dados.



The screenshot displays a mobile application interface for creating a new record. At the top, there is a header bar with a blue background. On the left, there is a white back arrow icon. In the center, the text 'New Record' is displayed in white. On the right, there is a white save icon. Below the header bar, there are two input fields. The first field is labeled 'id (INT) *' in red text and contains the value '1'. The second field is labeled 'texto (TEXT)' in black text and contains the value 'teste'. At the bottom of the screen, there is a banner with a blue background. On the left, there is a white circular icon with a blue arrow. In the center, the text 'Baixe grátis e veja o resumo do seu extrato pelo celular.' is displayed in white. On the right, there is a white download icon. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

Fonte: SQLite Magic.

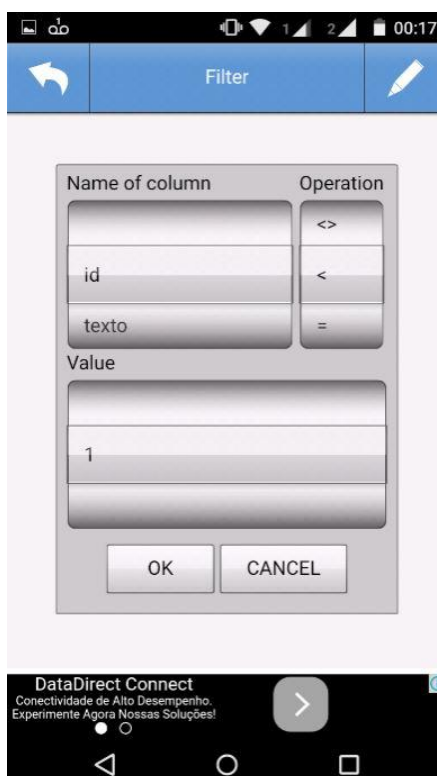
Figura 2.17- Problema ao tentar utilizar o campo do tipo multimídia.



Fonte: SQLite Magic.

Na área destinada aos filtros de visualização (Figura 2.18), a interface deixa a experiência do usuário fluir de uma maneira contínua e clara, pois tudo que ele pode executar é exibido em uma mesma forma de escolha. Nesse aplicativo, as condições para filtragem se resumem às operações lógicas básicas.

Figura 2.18- Tela de aplicação de filtros para visualização.



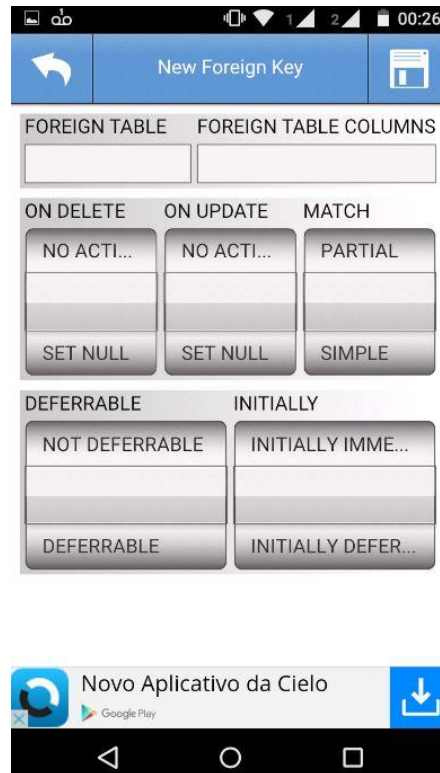
Fonte: SQLite Magic.

Com as funcionalidades e as possibilidades que o sistema oferece devidamente analisadas, foram postos em prática os testes propostos. Nesse aplicativo, foi constatado que existe a permissão de criação de tabela sem chave primária (Teste 1.1), que ao tentar adicionar dois campos como chave primária (Teste 1.2) ele exibe um erro e que não há opções para a criação de chave primária composta (Teste 1.3). Para os testes referentes à chave estrangeira (testes do tópico 2), foi criada uma coluna com essa propriedade na área específica para tal operação (como mostra a Figura 2.19). Ao executar os testes, nenhum erro foi encontrado, ou seja, houve permissão para a execução de todos os itens da lista proposta, o que acarreta em um problema de consistência da base de dados. Em relação aos campos auto-incrementais, também houve permissão para executar os testes 3.1 e 3.2, o que comumente não é possível em um SGBD tradicional.

Por fim, é observado que o fluxo de utilização do aplicativo segue um padrão, o que facilita a rápida aprendizagem do usuário. Alguns problemas foram encontrados em relação à interface, como botões sem descrição de sua função, mas nada que impedisse uma experiência agradável. Outra questão foram as dificuldades técnicas encontradas em relação ao banco de dados, que poderiam se tornar uma complicação para o desenvolvimento de um esquema mais

complexo. O Quadro 2.3 mostra as notas finais atribuídas ao aplicativo, considerando todos os critérios descritos no início do capítulo.

Figura 2.19- Criação de chave estrangeira.



Fonte: SQLite Magic.

Quadro 2.3- Avaliação final do SQLite Magic.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
Nota	4	1	3	1

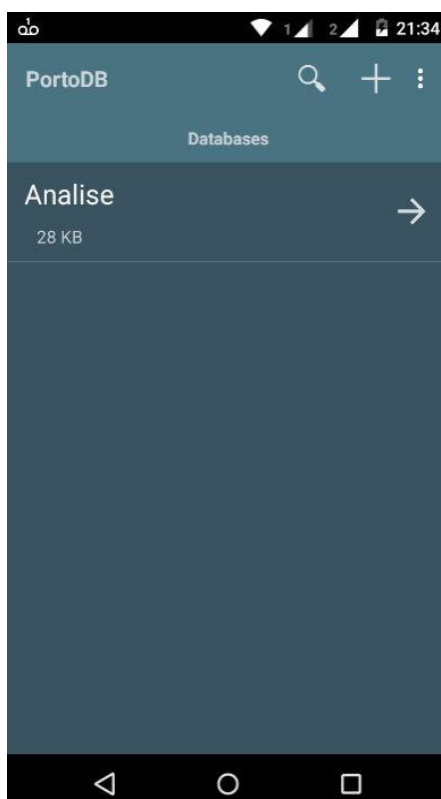
Fonte: O Autor.

2.4. PortoDB

O PortoDB, desenvolvido pela PortoFarina¹³, é a última ferramenta a ser analisada. Nela, é possível consultar e manipular bancos de dados novos ou já existentes no sistema, como nos aplicativos citados anteriormente. Na Google Play Store, este aplicativo está com a quantidade de *downloads* estimada em um valor entre 5.000 e 10.000 e possui 157 avaliações, com uma nota média de 4.4 (em uma escala de 1 a 5).

Com uma interface minimalista e direta, o sistema tem início na tela de listagem de banco de dados já existentes, como mostra a Figura 2.20. Ela apresenta botões de busca, criação de novo repositório e outras opções, como recuperação via *backup* e acesso às configurações da aplicação, em um menu extra.

Figura 2.20- Tela inicial do aplicativo.

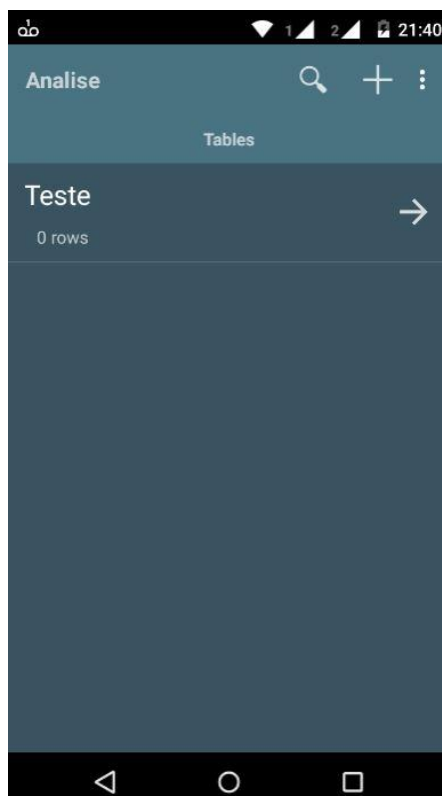


Fonte: PortoDB.

¹³ <https://play.google.com/store/apps/developer?id=PortoFarina>

Ao clicar em um item da lista de repositórios, a tela de listagem de tabelas (Figura 2.21) surge, possibilitando a continuidade do fluxo de utilização. Mantendo o mesmo estilo e funcionalidades de configuração e busca da anterior, esta tela apresenta como novidade a criação de tabelas, mostrando que a reutilização de componentes de interface é um ponto importante para trazer facilidade e comodidade ao usuário.

Figura 2.21- Listagem de tabelas.



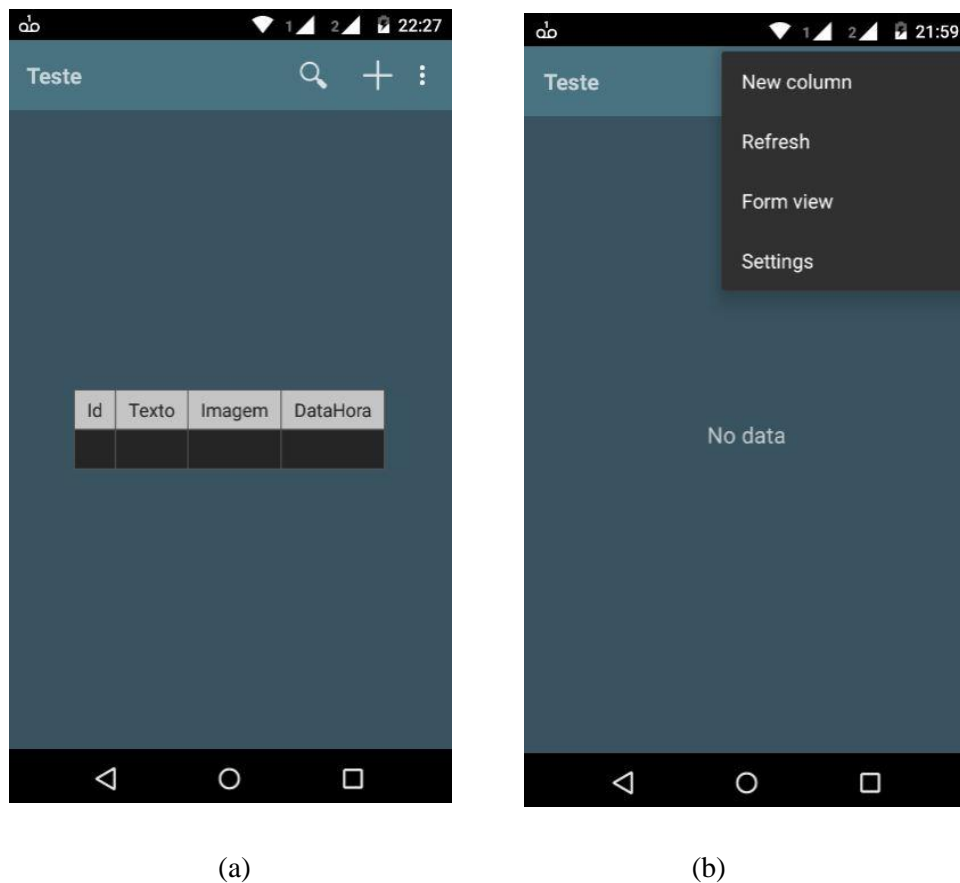
Fonte: PortoDB.

A partir da escolha da tabela que se deseja utilizar, ficam disponíveis a inserção e edição de colunas na tela de exibição das mesmas. Aqui, diferente do que foi mostrado anteriormente, esta funcionalidade não fica no botão “+” (botão este que vai ter outra funcionalidade, que será mostrada em seguida), mas sim no menu exibido na direita, como mostra a Figura 2.22. Essa quebra de padrão pode acarretar em uma dificuldade para os novos usuários.

Na inserção da coluna, uma caixa em destaque aparece e nela é possível inserir seu nome e o seu tipo. Em relação aos dados multimídia, o sistema oferece apenas a opção do tipo Image e não de algum tipo mais geral, como o BLOB, que permite a inserção de outros tipos de formatos além de imagens.

Para manipular a coluna, duas ações são possíveis: clique simples, que levará à caixa de renomeação da mesma, e clique longo (ambos no nome da coluna), que levará a mais opções de manipulação, como alteração do tipo, eliminação e movimentação da coluna na visualização da tabela.

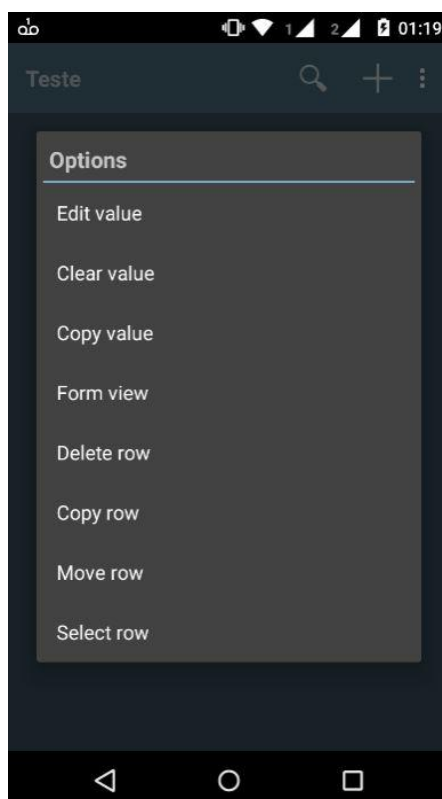
Figura 2.22- Exibição de colunas (a) e botão de inserção de nova coluna (b).



Fonte: PortoDB.

Ao inserir a primeira coluna da tabela, o botão “+” volta a aparecer. A função dele nessa tela é a de inserir novas linhas na tabela, para que o usuário edite seu conteúdo. A manipulação dessas linhas é feita pelo clique em cada campo destinado ao dado e, ao fazer um clique longo, novamente mais opções ficam disponíveis (Figura 2.23).

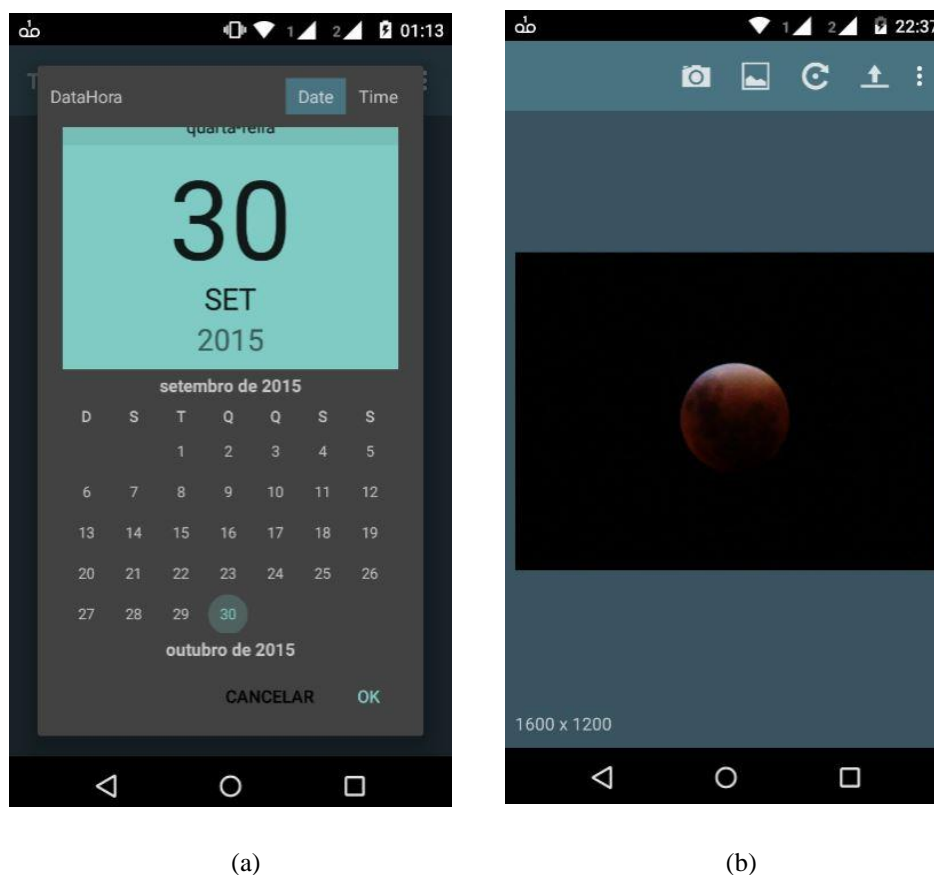
Figura 2.23- Opções para manipulação dos dados.



Fonte: PortoDB.

É interessante observar que nesse aplicativo os diversos tipos de dados recebem um tratamento diferente durante a inserção de conteúdo. A Figura 2.24 mostra, como por exemplo, a inserção em campos do tipo DateTime e Image, nos quais um calendário é exibido (com a opção de adicionar o tempo via interface por meio do botão “Time”) para o primeiro e para o segundo uma nova tela, que conta com a opção de escolha da imagem (tanto pelo uso da câmera quanto pela galeria de imagens do *smartphone*) e rotação da mesma.

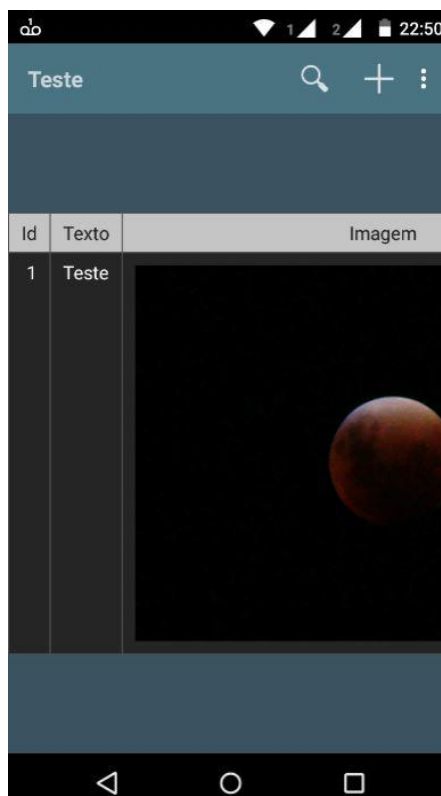
Figura 2.24- Inserção em campos do tipo DateTime (a) e Image (b).



Fonte: PortoDB.

Porém, ao inserir uma imagem na visualização de colunas, o usuário precisa deslizar lateralmente a tabela para conseguir vê-la por completo, pois, como a imagem pode ser grande, ela acaba tomando muito espaço. Nas configurações do sistema foi encontrada a opção de redimensionamento da imagem na visualização, mas para a imagem utilizada não houve nenhuma alteração gráfica em qualquer uma das opções fornecidas. Esse problema na exibição da tabela com uma coluna preenchida com imagem pode ser visualizado na Figura 2.25.

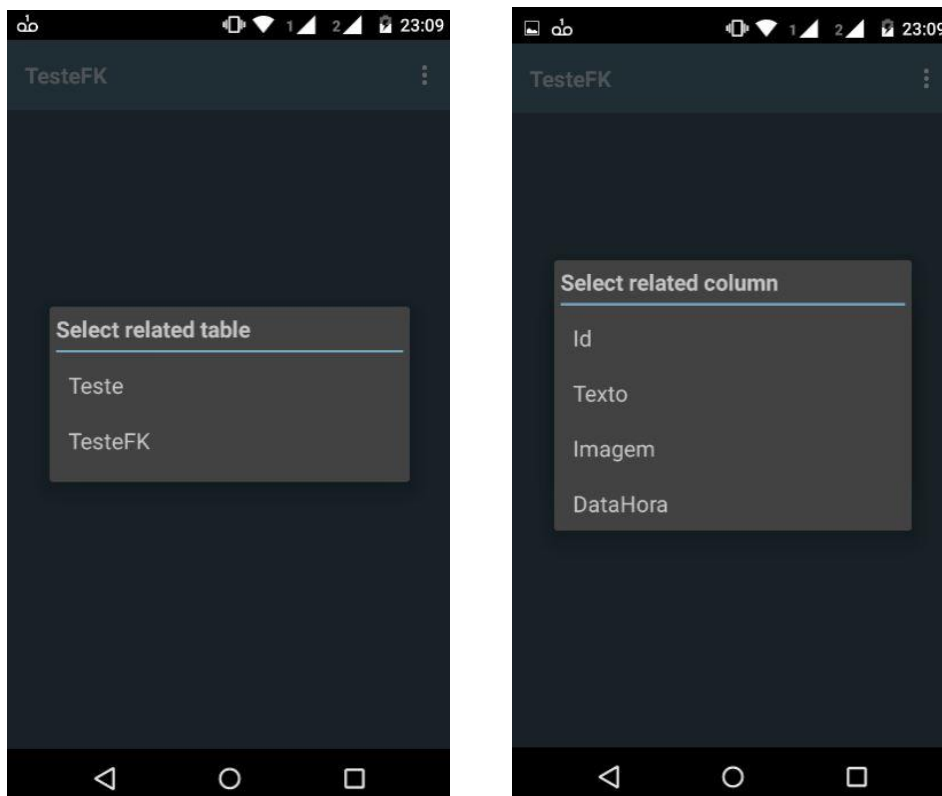
Figura 2.25- Problema com visualização de tabela.



Fonte: PortoDB.

Em relação aos testes citados no início do capítulo, o PortoDB não possibilita a realização dos que se referem às chaves primárias (testes do tópico 1), pois não foi encontrada uma opção para inserir explicitamente uma coluna com essa propriedade. Para os testes relacionados à chave estrangeira, uma coluna com esse tipo (chamado *Relationship*) foi adicionada (Figura 2.26). Durante a execução desta ação, o aplicativo se mostra claro quanto aos passos que o usuário deve executar para conseguir finalizá-la com êxito (primeiro escolhendo a tabela a qual se deseja referenciar, e em seguida a coluna da que foi escolhida), o que traz maior segurança para o usuário e, além disso, mostra o êxito do Teste 2.1. Após a criação, os testes 2.2 e 2.3 foram realizados e para ambos o aplicativo se mostrou robusto, pois ele permite apenas a escolha do campo de referência já inserido previamente. Já em relação ao Teste 2.4, houve um comportamento peculiar, pois ao excluir um campo que é referenciado em outro, o campo que é chave estrangeira passa a não ter nada em seu conteúdo, mas nenhuma mensagem de segurança foi exibida ao excluir o dado importante para essa questão. Os testes do tópico 3 não puderam ser realizados pois não foi encontrada forma de se trabalhar com campos auto-incrementais.

Figura 2.26- Inserção de chave estrangeira.



Fonte: PortoDB.

Ao utilizar o sistema, é observado que houve um cuidado grande em relação à interface do usuário, pois poucos questionamentos aparecem ao seguir o seu fluxo. O mesmo cuidado é observado em relação ao tratamento de dados diferenciados, pois é possível fazer a grande maioria das manipulações deles via botões ou regiões clicáveis. Por outro lado, a limitação dos tipos de dados multimídias possíveis e o fato de alguns detalhamentos não serem encontrados, como mencionado anteriormente para o caso das chaves primárias e campos auto-incrementais, podem gerar dificuldades, caso o usuário deseje elaborar um esquema mais complexo. Por fim, o Quadro 2.4 mostra as notas atribuídas ao aplicativo.

Quadro 2.4- Avaliação final do PortoDB.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
Nota	5	3	4	3

Fonte: O Autor.

2.5. Considerações finais sobre as análises

Após a realização das análises, é possível encontrar as dificuldades atuais no mercado, de maneira geral, em relação à visualização e à manipulação de dados. O Quadro 2.5 traz todas as avaliações feitas durante o capítulo, com a finalidade de encontrar tais carências.

Quadro 2.5- Todas as avaliações realizadas no capítulo.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
SQLite Editor	3	3	4	1
SQLite Master	3	1	3	1
SQLite Magic	4	1	3	1
PortoDB	5	3	4	3

Fonte: O Autor.

Pelas notas atribuídas, fica claro que as necessidades maiores estão em usufruir de melhor forma as regras básicas de banco de dados e em utilizar os dados multimídia. Estes pontos são, de fato, importantes, pois, como se pretende melhorar o ambiente de desenvolvimento de banco de dados, é de se esperar que os desenvolvedores consigam fazer uso de uma plataforma completa e que consigam elaborar esquemas complexos, com certo nível de detalhamento. Com esses quesitos identificados, é importante usar todo o

embasamento adquirido na elaboração de uma nova ferramenta, corrigindo os problemas analisados.

No próximo capítulo, serão abordadas formas de trabalhar a interface no contexto de banco de dados, para que, por exemplo, as necessidades com dados multimídia sejam supridas e os usuários tenham uma boa experiência ao utilizar a aplicação desenvolvida.

3. ANÁLISE DA INTERFACE GRÁFICA E EXPERIÊNCIA DO USUÁRIO

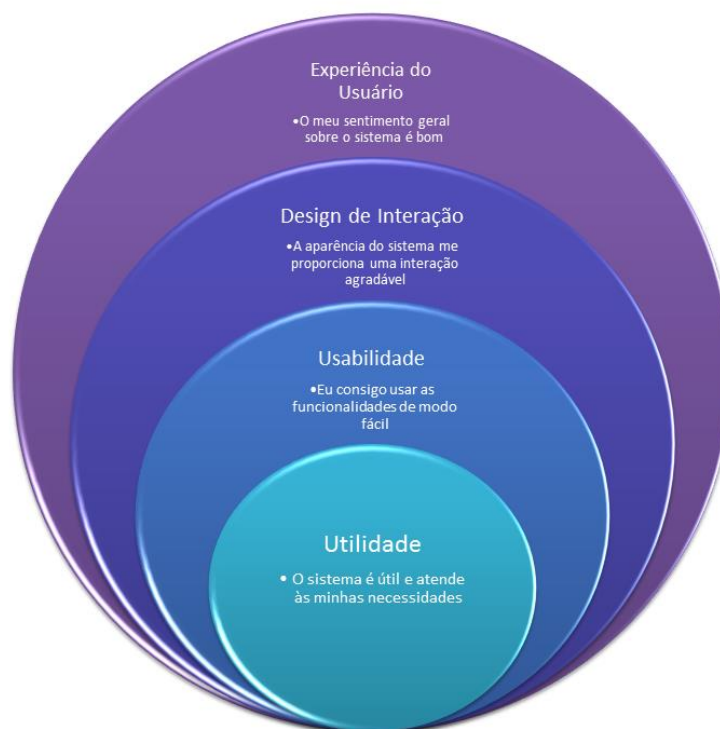
Quando se está interessado em desenvolver alguma aplicação, é necessária a preocupação com algumas variáveis que vão impactar na experiência do usuário, como, por exemplo, a plataforma para qual ela será desenvolvida, pois a interação do referido usuário com o *design* elaborado é a ferramenta mais importante para o desenvolvedor ajudá-lo a alcançar seu objetivo. Em se tratando de *smartphones*, algumas complicações são comuns para todos os tipos de sistemas que eles dão suporte, como, por exemplo, o tamanho da tela do aparelho e a forma de lidar com as informações existentes no aplicativo, para que o usuário não sinta dificuldades na utilização, justamente por conta do espaço reduzido para organizá-las. Resolver ou melhorar esses aspectos relacionados à usabilidade são fundamentais para conseguir que uma aplicação seja amplamente aceita no mercado ao qual ela está inserida.

A usabilidade possui diversas definições atualmente, mas todas elas seguem alguns pilares básicos. Segundo Nielsen (1993), a usabilidade é a medida de qualidade da experiência de um usuário ao interagir com um produto ou sistema. Ainda de acordo com ele, esta medida está distribuída a vários elementos, como a facilidade de aprendizagem por parte do usuário, a eficiência do sistema para o seu propósito, a facilidade de memorização (para que, uma vez que a aplicação seja utilizada, o processo de aprendizado não precise ser repetido), a segurança (em relação à prevenção de erros) e a satisfação do usuário (apud RABELO, s.d.).

Segundo a ISO (*International Standard Organization*) e sua norma ISO91260, a usabilidade é definida pelos seguintes aspectos: inteligibilidade, ou seja, o esforço do usuário para reconhecer o conceito lógico e a aplicabilidade do sistema; conformidade, no que diz respeito ao software estar de acordo com as regras que ele precisa seguir; operacionalidade; aprendizado e atratividade (apud RABELO, s.d.).

Outras normas e definições foram elaboradas, alteradas e difundidas durante anos e, por isso, as variações sobre o que é usabilidade são amplas. Porém, em um ponto todas elas entram em acordo: o usuário sempre deve se sentir bem durante a utilização e sempre deve alcançar seu objetivo no sistema, como mostra a Figura 3.1.

Figura 3.1- Diagrama sobre experiência do usuário.



Fonte: Tableless (2011).

Com a realização da análise dos aplicativos semelhantes no capítulo anterior, foi possível observar que existem algumas carências em relação à melhor forma de utilizar os elementos que o Android disponibiliza para as aplicações analisadas. Problemas encontrados no fluxo de utilização das aplicações, que por vezes se tornava confusa para o usuário, e na forma de exibição dos dados, como estrutura de tabelas e visualização dos dados do tipo multimídia (que mostraram problemas no dimensionamento desses dados em relação à tela), serão estudados neste capítulo, com a finalidade de encontrar melhores abordagens para a aplicação que será desenvolvida neste trabalho.

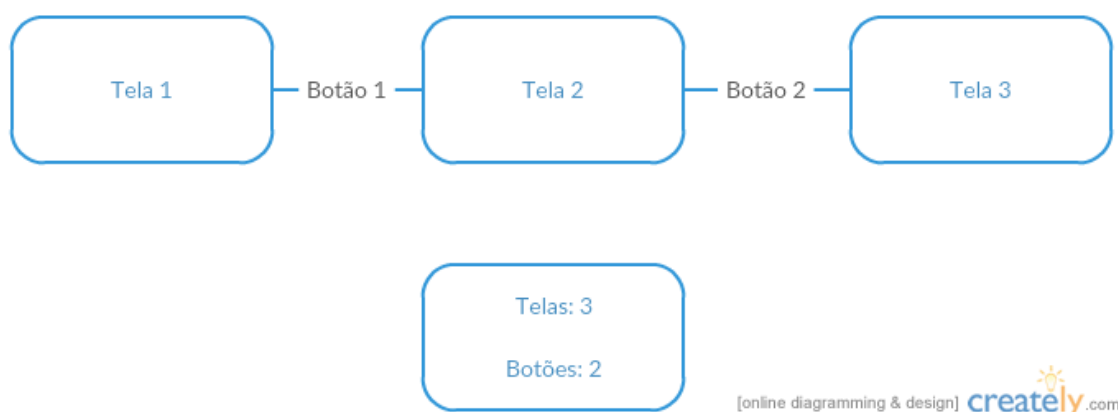
3.1. Fluxo de utilização

A primeira característica a ser analisada será o fluxo de utilização dos aplicativos citados no capítulo anterior. Para cada um deles será elaborado um diagrama, de modo a mostrar o processo que o usuário executa para a criação do banco de dados até a inserção da primeira informação (utilizando essa mesma ação para todas as aplicações e considerando a

criação de apenas um banco de dados, uma tabela e uma coluna), observando a quantidade de passos que o usuário leva para a realização da mesma. Essa etapa tem a finalidade de tornar esta ação mais rápida e clara possível para a nova aplicação, com o objetivo de melhorar a experiência do usuário e tornar menos confusos os processos relacionados à elaboração de bancos de dados.

Para o estudo, serão consideradas duas métricas: a quantidade de cliques (botões) e de telas que o usuário precisa acessar para a finalidade descrita anteriormente. Como não foram encontrados diagramas padronizados para exibir este tipo de informação, será exibido o modelo que foi criado especificamente para este trabalho. Nele, serão descritos as telas e os botões por onde o usuário passou e na parte inferior a quantidade total dessas métricas, considerando a utilização correta do sistema. A Figura 3.2 mostra um exemplo deste diagrama construído com a ferramenta Creately¹⁴, para que torne o entendimento mais claro.

Figura 3.2- Exemplo do diagrama de fluxo de utilização.

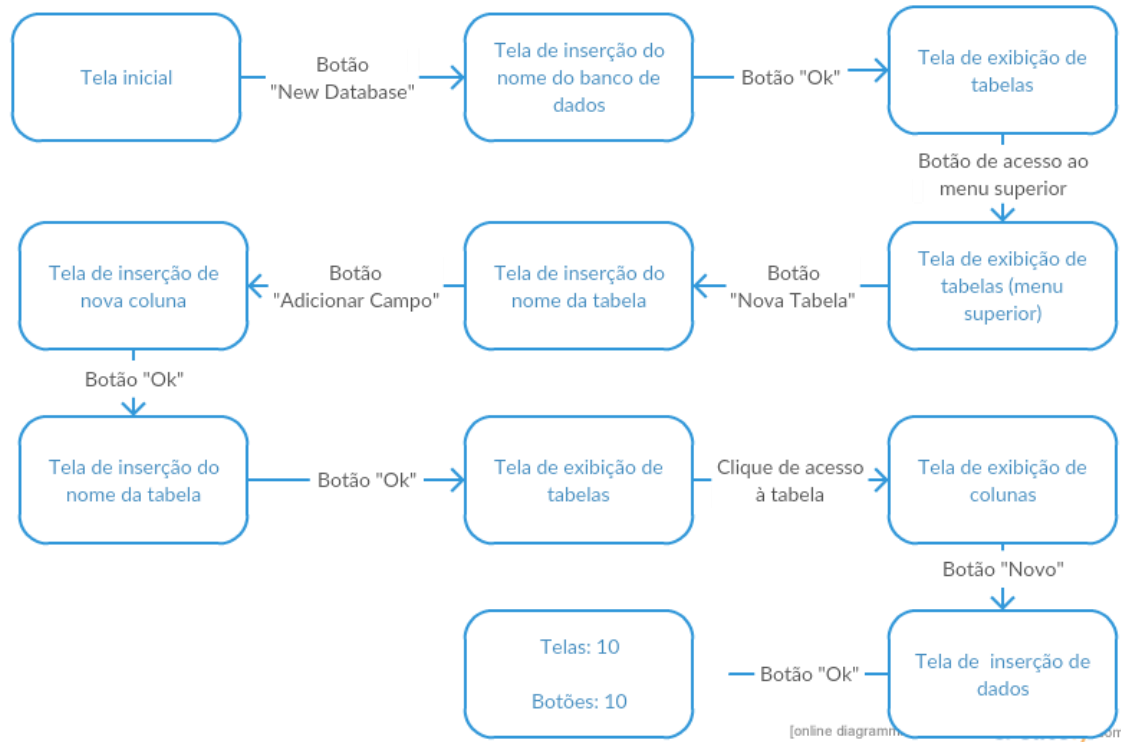


Fonte: O Autor.

Com a apresentação do modelo de diagrama, pode-se aplicá-lo aos aplicativos estudados no capítulo anterior. As Figuras 3.3, 3.4, 3.5 e 3.6 mostram a utilização dessas aplicações para a ação teste citada anteriormente.

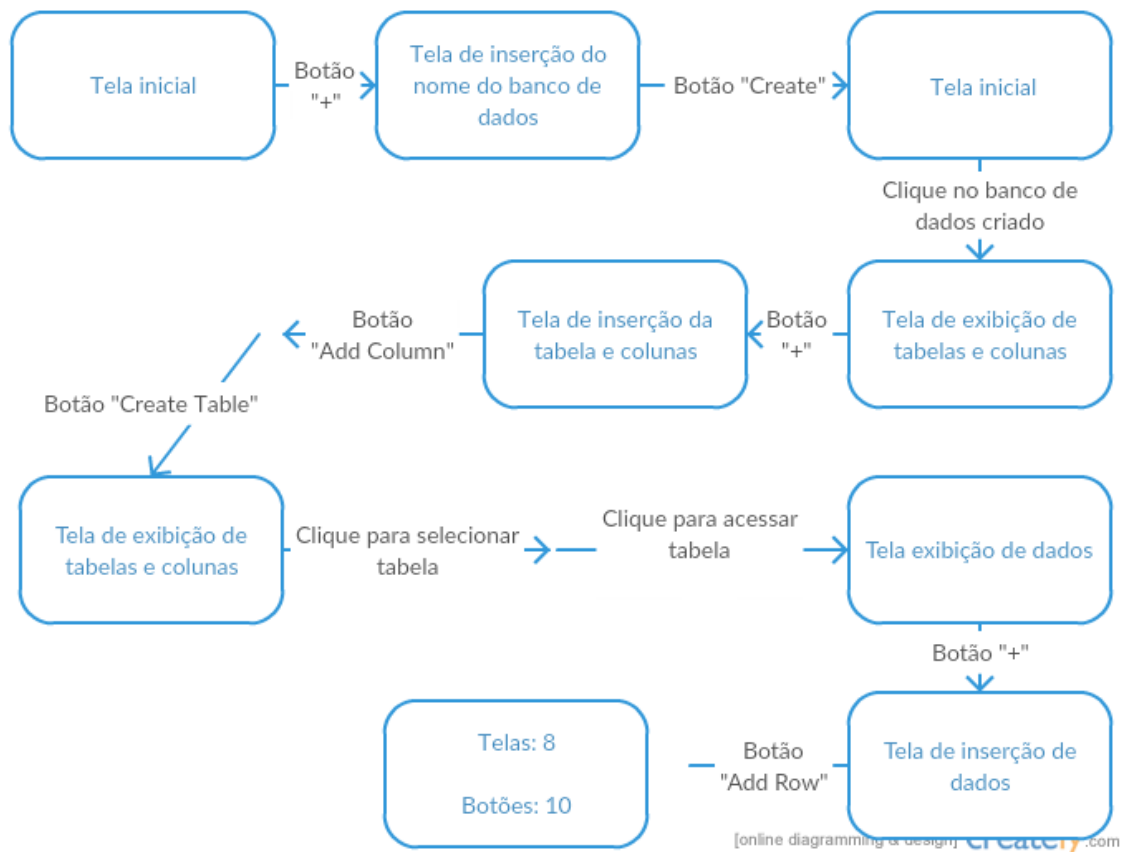
¹⁴ <http://creately.com/>

Figura 3.3- Fluxo de utilização para o SQLite Editor.



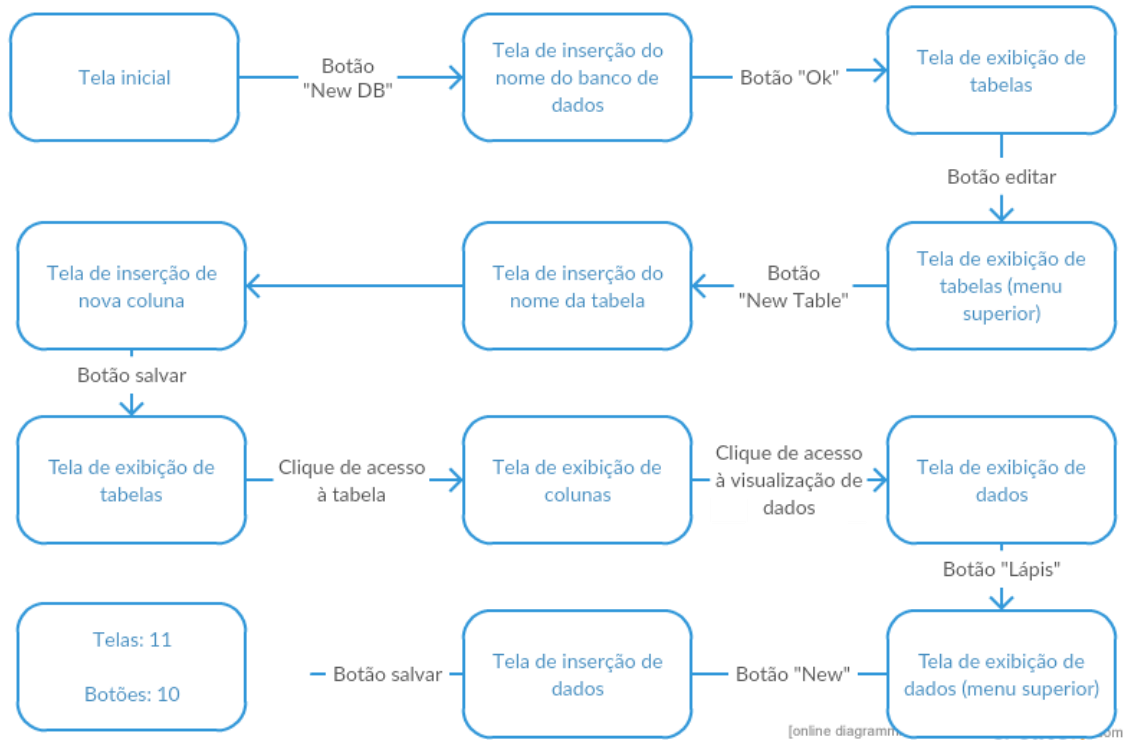
Fonte: O Autor.

Figura 3.4- Fluxo de utilização para o SQLite Master.



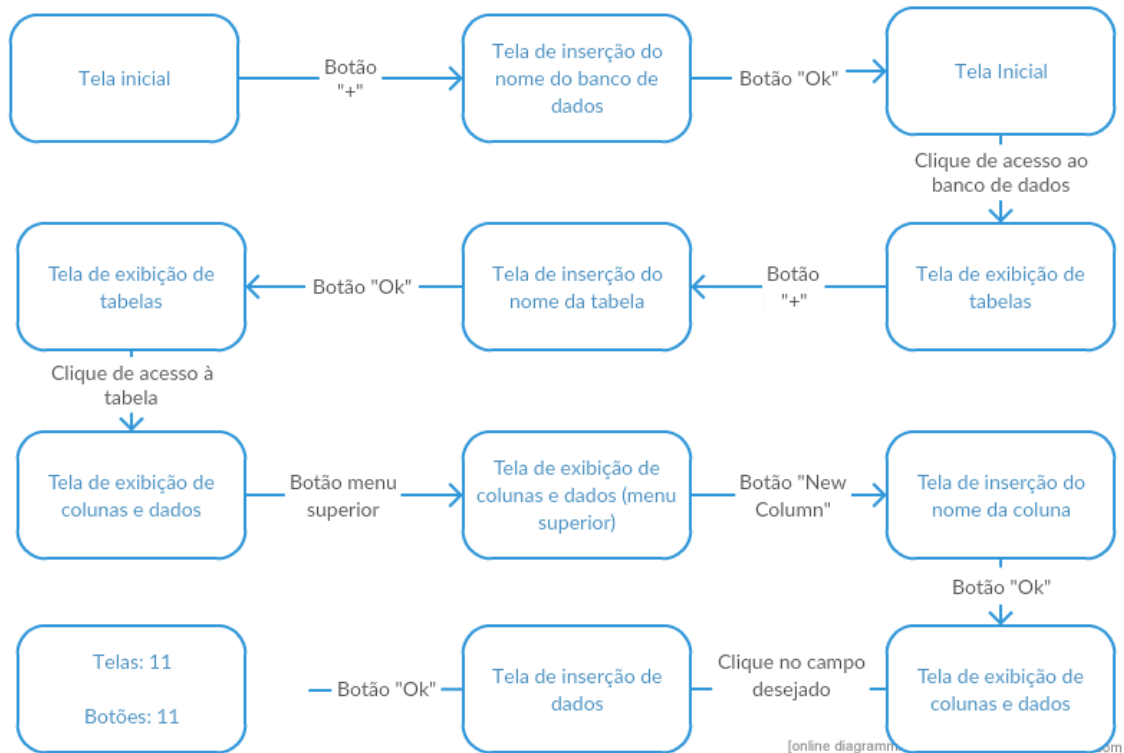
Fonte: O Autor.

Figura 3.5- Fluxo de utilização para o SQLite Magic.



Fonte: O Autor.

Figura 3.6- Fluxo de utilização para o PortoDB.



Fonte: O Autor.

A partir desses diagramas, algumas conclusões importantes para o desenvolvimento da nova aplicação podem ser tomadas. É interessante observar que o PortoDB utiliza mais telas e cliques que os demais, mesmo sendo o aplicativo que obteve as melhores notas no capítulo anterior. Isso é explicado pelo fato que ele utiliza telas repetidas (que nos diagramas são contadas como telas diferentes por se tratarem de momentos de utilização diferentes) e adota um padrão em suas telas (tornando-as semelhantes), provendo uma melhor adaptação do usuário ao sistema, já que ele tem mais contato com as mesmas interações. Para comparar essa diferenciação do uso das telas e interações, o Quadro 3.1 mostra as quantidades de cada aplicativo.

Quadro 3.1- Padrão do aplicativo baseado em telas e botões.

	Quantidade de telas diferentes utilizadas	Quantidade de botões diferentes utilizados
SQLite Editor	8	7
SQLite Master	6	8
SQLite Magic	10	9
PortoDB	8	6

Fonte: O Autor.

Na literatura, há diversos estudos explicando que quanto menos passos o usuário realizar para chegar ao seu objetivo, melhor será a aceitação dele e consequentemente sua experiência com o sistema. Logo, pode-se constatar que neste caso, baseado na análise do capítulo anterior e no quadro acima, o cenário ideal seria uma quantidade de telas reduzidas ao mínimo necessário e uma quantidade de botões mínima. Assim aplicações com essa complexidade, pela sua quantidade de informações, se tornariam mais agradáveis ao público alvo.

3.2. Exibição de dados

A exibição dos dados tem papel fundamental para este tipo de aplicação, porque é uma das principais vantagens que fará o usuário utilizar o sistema na elaboração de esquemas de bancos de dados. Uma boa abordagem para permitir a visualização de todos os tipos de

dados disponíveis no sistema, sem problemas ou interrupções em relação à interface gráfica, é o ideal para o cenário que está sendo estudado.

Para este tópico, foi planejada uma análise de como todos os aplicativos se comportam referentes à exibição dos dados do tipo multimídia, porque existe uma carência nesse sentido, como foi descrito no início do capítulo. Porém, apenas o aplicativo PortoDB permite a visualização desse tipo de dado e, sendo assim, será o único que terá seus problemas detalhadamente analisados.

No PortoDB, o usuário apenas consegue enxergar as imagens que a aplicação o permite inserir via *upload* de um arquivo do sistema, ou a partir da captura de imagem via câmera. Essas imagens aparecem junto aos demais campos (caso eles existam) e em um tamanho que pode acabar fazendo esses campos ficarem indisponíveis à primeira vista para o usuário, forçando-o a deslizar a tela para alcançar as colunas que deseja ter acesso.

No aplicativo foram encontrados dois ajustes de visualização (redimensionamento da imagem e movimentação para alterar a ordem das colunas), como citados no capítulo anterior, porém eles não foram suficientes para a resolução deste problema. O redimensionamento não surtiu efeito para nenhuma das opções de tamanho que o sistema disponibiliza. Já a troca da ordem de visualização das colunas permite que o usuário configure a seu gosto como gostaria de enxergar sua tabela, porém se ele precisar fazer isso sempre que deseje ter mais foco em campos textuais do que na coluna da imagem ou vice-versa acaba gerando uma ação repetitiva, o que não é interessante.

O PortoDB faz uso de *ImageViews*¹⁵, o que permite utilizar imagens no *layout* e fazer algumas manipulações sobre ela. Porém a utilização pura deste componente gera a problemática citada, então outras possibilidades devem ser analisadas. Para tentar amenizar o impacto não só de imagens, como também outros tipos de dados multimídias (como áudio e vídeo) é válido buscar primeiramente o que o sistema operacional Android oferece para o tratamento desse tipo de conteúdo.

Essa plataforma possui alguns benefícios para seus desenvolvedores, e um deles é permitir via *Intents*¹⁶ (uma abstração da ação que se deseja executar) chamar aplicações básicas do sistema para uso voltado à aplicação em execução. No caso da exibição dos dados multimídia, o *Intent* possui a constante *ActionView*, que, ao ser utilizada, avisa o sistema que o usuário está requisitando que esse tipo de dado seja aberto em alguma aplicação do Android que dê suporte a essa requisição (como por exemplo, a galeria de imagens padrão da

¹⁵ <http://developer.android.com/reference/android/widget/ImageView.html>

¹⁶ <http://developer.android.com/reference/android/content/Intent.html>

plataforma). Fazer uso dessa possibilidade levaria o usuário a uma visualização isolada da imagem, vídeo ou áudio, ou seja, apenas quando achar necessário. Dessa forma, a exibição dos demais dados do banco não será prejudicada pelo dado multimídia, pois não disputaria espaço na área destinada a ele.

Ainda sobre as ferramentas disponibilizadas pelo Android, outra opção seria o *Media Playback*¹⁷, que opera sobre os tipos de dados citados e possui métodos que facilitam o entendimento para seu uso. Este *framework* funciona usando o *link* do conteúdo desejado para abrir seu *player* próprio.

3.3. Considerações finais

Com a análise realizada percebe-se que pequenos detalhes durante a construção do *design* de uma aplicação fazem toda a diferença quando se trata da experiência que o usuário vai ter com ela. Uma decisão sem embasamento, seja referente ao que já existe atualmente ou com testes com o público alvo, pode acabar gerando, dentre outros problemas, dificuldade de aprendizado e consequentemente problemas de utilização. Tomando isso como um princípio, é importante que a aplicação a ser desenvolvida seja verdadeiramente voltada ao seu propósito e que possa ser utilizada uma maneira simples, para que tenha uma boa aceitação.

Por fim, cada melhoria apresentada será posta em prática e descrita detalhadamente no próximo capítulo, que tratará da nova aplicação que será desenvolvida. Nesta etapa, serão feitas as escolhas das estratégias que serão utilizadas dentre as estudadas tanto em relação as funcionalidades e regras quanto sobre a interface gráfica.

¹⁷ <http://developer.android.com/guide/topics/media/mediaplayer.html>

4. DATA HANDLING – VISUALIZAÇÃO E MANIPULAÇÃO DE DADOS DE BD EM SMARTPHONES

A partir dos resultados das análises realizadas neste trabalho, é possível a elaboração de uma ferramenta que possa resolver ou minimizar os problemas encontrados resultando em um sistema que se adeque da melhor forma possível ao público alvo.

Este capítulo trará uma abordagem exploratória sobre a ferramenta Data Handling, passando por cada etapa do seu desenvolvimento, como a motivação pela escolha do formato e fluxos de telas e a lógica utilizada no seu *back-end*. As dificuldades encontradas e a forma de contorná-las também serão explicitadas, com a finalidade de trazer uma visão completa do processo de desenvolvimento.

Como já existem ferramentas que realizam boa parte de algumas funcionalidades básicas, se tratando do contexto de banco de dados e sua utilização, não seria necessário iniciar um novo sistema ignorando todas as contribuições que já existam. Por isso, foram aproveitados alguns componentes dos aplicativos previamente descritos, sejam eles visuais ou de lógica interna. Durante este capítulo, algumas dessas referências serão citadas, reafirmando que o propósito desse projeto é realizar uma melhoria no que o mercado já possui.

4.1. Ideação da solução

O primeiro passo do projeto foi pensar em pontos críticos para a aplicação, ou seja, aquilo que mais necessitaria de uma lógica diferenciada. A partir das dificuldades encontradas, ficou claro que os pontos de melhoria se concentram tanto no *design* como na forma de trabalhar com dados do tipo BLOB. Fazer um sistema direto, simples, mas completo em seu propósito é o que se deseja alcançar, pois assim o usuário entenderá como poderá tirar total proveito dele sem precisar de muito tempo para atingir esse objetivo.

Para isso, muitos componentes básicos disponíveis para os desenvolvedores da plataforma Android foram pesquisados para trazer uma melhor navegação para o usuário.

Esses componentes serão devidamente explicados e exibidos futuramente, ao explicar o resultado do design para a aplicação.

Outro ponto crítico é a forma de tratar o banco de dados na aplicação, pois o usuário precisa ter a sensação de estar de fato utilizando diretamente uma base de dados e não apenas uma lógica feita para simular o mesmo. Visando isso, foram buscadas soluções que, ao mesmo tempo, pudessem servir para o desenvolvedor armazenar as informações necessárias da aplicação e para que o usuário pudesse ter seus próprios bancos de dados, conseguindo manipulá-los da forma que bem desejasse (obedecendo às regras básicas desses repositórios).

Com os pontos críticos listados, foi necessário listar as funcionalidades principais que seriam contempladas, para que a partir delas fosse montada uma estrutura inicial para os fluxos de utilização. Essas funcionalidades são: criar bases de dados, criar suas tabelas, criar as colunas dessas tabelas usando os tipos básicos (ou seja, os tipos que são adicionados a partir da entrada do usuário sem a necessidade de tratamento), criar colunas dessas tabelas usando o tipo multimídia (ou seja, utilização do tipo BLOB), inserir os dados do tipo básico, inserir os dados do tipo multimídia (utilizando os artifícios do Android para isso), visualizar os dados básicos, visualizar de forma diferenciada os dados do tipo multimídia e por fim, editar tanto as estruturas criadas como os dados inseridos. Essas funcionalidades conseguem, em sua totalidade, fornecer o que o usuário precisa, no contexto abordado.

Tendo essas definições, é possível começar a etapa prática do desenvolvimento, na qual a construção do aplicativo tem início de fato. Para isso, foi utilizada a IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado, em português) Eclipse¹⁸, cujo foco é voltado para aplicações feitas na linguagem Java¹⁹ (linguagem essa que é a utilizada para sistemas na plataforma Android) e que possui *plug-ins* para se adaptar à construção de aplicativos Android.

¹⁸ <https://eclipse.org/>

¹⁹ https://www.java.com/pt_BR/

4.2. Esquema de projeto

Um projeto Android é normalmente composto de alguns tipos de arquivos, os quais se fazem presentes neste projeto. Componentes como arquivos Java chamados de *Activity*²⁰, que representam cada um deles uma tela ou, ainda mais, uma funcionalidade distinta; arquivos do tipo XML²¹ (*eXtensible Markup Language* ou linguagem de marcação extensível, em português), responsáveis por organizar os componentes que ficarão disponíveis na interface do usuário; e por fim arquivos Java que representam objetos que serão utilizados no projeto.

Em relação aos objetos que serão essenciais para o andamento da aplicação, foi necessário criar quatro tipos deles: Banco, Tabela, Coluna e Alteração. Eles são responsáveis por fazer a intermediação entre o que o usuário deseja criar ou editar, com todas as suas características, para que sejam armazenados até o momento que serão de fato passados para o repositório.

Esses objetos também permitem operações otimizadas na aplicação, pois, sempre que o usuário deseja criar uma nova estrutura ela é armazenada em memória até que seja solicitado seu processamento em lote, para o banco de dados do usuário. Para ilustrar e tornar o entendimento o mais claro possível, suponha-se um caso no qual o usuário da aplicação crie um novo banco de dados, duas tabelas para ele e em cada uma delas, cinco colunas. Neste caso será criada uma instância do objeto Banco, duas do objeto Tabela e dez do objeto Coluna e, como todas as características estão dentro dessas instâncias, nenhum tipo de referência é perdido entre eles, na visão de um banco de dados tradicional. O objeto Alteração, por sua vez, é responsável por armazenar os seis tipos de alteração com suporte para o esquema do usuário: adição de nova coluna, adição de nova tabela, edição de coluna, edição de tabela, remoção de coluna e remoção de tabela. Ao receber esse tipo de objetos, métodos específicos para cada tratamento de alteração são acionados para executar o que foi passado para eles.

Quando uma funcionalidade de visualização, inserção ou edição de dados (que requerem de fato uma base de dados criada) for requisitada pelo usuário, essas instâncias citadas anteriormente serão repassadas para a classe responsável pela geração do repositório de forma concreta, no SQLite.

²⁰ <http://developer.android.com/reference/android/app/Activity.html>

²¹ <http://www.w3.org/XML/>

Outra característica importante do projeto é a distinção entre os repositórios do usuário e o da aplicação em si. Enquanto os primeiros são gerados a partir das escolhas do usuário em relação ao esquema do seu banco durante a utilização da aplicação, o segundo é responsável por armazenar os objetos Banco, Tabela e Coluna a cada etapa passada pelo usuário. Para ilustrar essa diferença, imagine que quando é criada uma instância de Banco, duas de Tabela e cinco de Coluna para cada tabela, esses dados são armazenados na base de dados da aplicação. Ao solicitar a visualização dos dados pela primeira vez, uma consulta é feita no repositório da aplicação para que, com os dados obtidos das instâncias, seja criado de fato o banco de dados do usuário e ele possa começar a manipulação de dados.

Com essas características importantes citadas, em relação à construção do projeto, pode-se tratar da lógica de cada funcionalidade além das escolhas de *design* feitas, baseadas em todos os pontos importantes acerca de interface que já foram exibidos neste trabalho.

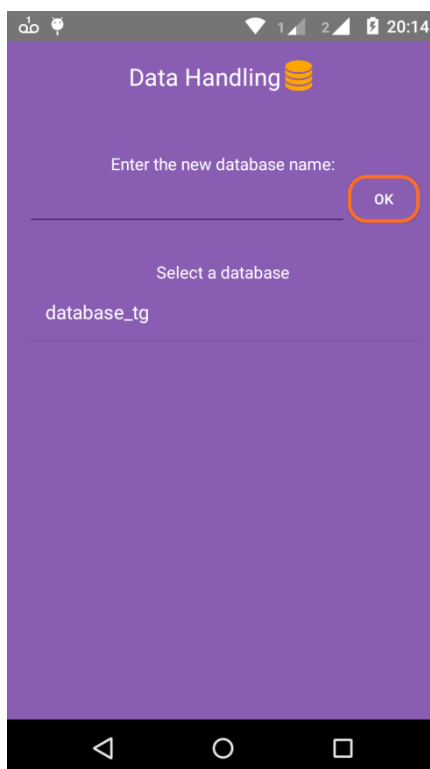
4.3. Funcionalidades e *design*

Tendo em mente as explicações fornecidas anteriormente, é possível começar a explorar, de maneira mais ampla, o sistema proposto. Nesta seção serão abordadas tanto as contribuições para a lógica de aplicações desse tipo quanto às melhorias de *design*, explicando sempre a forma de escolha dos componentes utilizados e a forma de elaboração do *design*.

Como visto nos estudos realizados em capítulos anteriores, interfaces que continham muitos botões ou muitas opções em uma mesma tela para o usuário acabavam se tornando confusas e de difícil utilização. Por essa razão, a simplicidade foi adotada em todo conteúdo apresentado, tanto do visual quanto dos comandos disponíveis (cliques de botão, *scroll* da tela, entre outros).

Por essa razão, o sistema já abre exibindo a tela de listagem dos bancos de dados já criados pelo usuário na aplicação, como mostra a Figura 4.1. Caso haja algum banco criado previamente, fica disponível a opção de acessá-lo para prosseguir o fluxo de utilização, mas, caso não haja nenhum, na mesma tela existe a opção de inserir o nome de um novo repositório e criá-lo.

Figura 4.1- Tela de listagem e criação de bancos de dados.

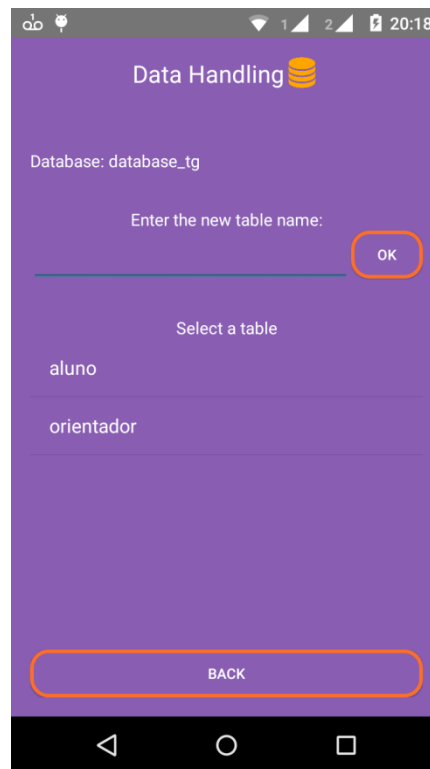


Fonte: O Autor.

Ao selecionar um dos bancos de dados, o usuário vai para a tela de escolha e criação de tabelas (Figura 4.2). Assim como na tela anterior, caso não existam tabelas criadas, o usuário pode criar nesta mesma interface a sua nova tabela, e acessá-la posteriormente. Este padrão foi baseado no aplicativo PortoDB e foi seguido em boa parte das interfaces, pois ao usá-lo ficou claro que esta maneira de apresentar dados e funcionalidades tornou a utilização mais simples e rápida, indo direto ao propósito.

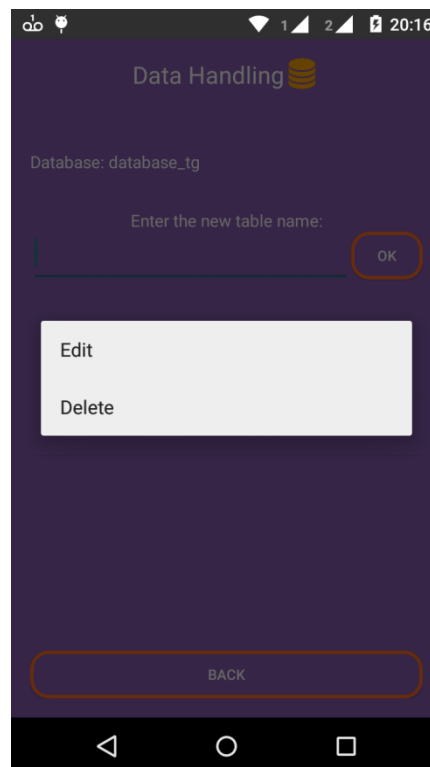
Ainda na lista de tabelas, quando o usuário realiza um clique longo em algum item, um *popup* para a edição ou remoção da tabela é exibido, como ilustra a Figura 4.3. Em caso de escolha da edição, a tela é mantida e agora o usuário pode escolher um novo nome para sua tabela. Ao escolher a opção de excluir a tabela, ela é automaticamente excluída (junto às colunas que a referenciam). Ambas as operações entram como uma nova instância do objeto Alteração que, quando executadas, são processadas imediatamente no banco da aplicação e posteriormente no banco do usuário.

Figura 4.2- Tela de listagem e criação de tabelas.



Fonte: O Autor.

Figura 4.3- Opções de alteração para tabelas.



Fonte: O Autor.

Cada tela apresentada até agora possui uma lista para a escolha do usuário, e esse padrão é mantido na tela de listagem das colunas da tabela, mas com algumas particularidades. A Figura 4.4 mostra a inclusão de dois botões novos, um para a criação de novas colunas e outro para salvar as alterações e visualizar os dados.

A criação de colunas teve que ser posta em uma tela à parte por duas razões: pelo espaço necessário para colocar as opções que o usuário necessita no momento da criação de uma coluna e para manter cada funcionalidade em sua área reservada.

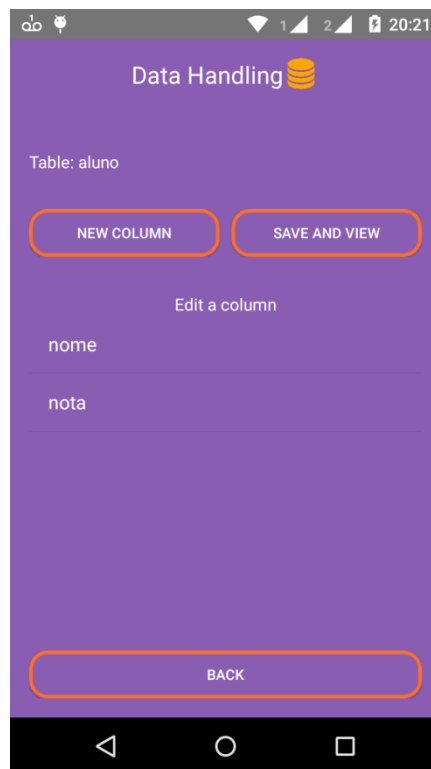
O botão para salvar e visualizar os dados, por sua vez, vai finalmente criar a base de dados, como explicado na seção anterior, e exibir a tela de visualização de dados, que será explicada futuramente neste documento. Por questões de organização no momento da exibição e limitações do tamanho do aparelho, esse botão está presente em cada tela de acesso ao conteúdo da tabela, e apenas ela será exibida na visualização, assim dando mais sentido ao fluxo linear que trouxe o usuário até esse ponto.

Outra característica importante dessa tela é que clicar em alguma das colunas listadas não mais levará para uma tela nova, pois não há mais fluxos desta forma a partir deste ponto. Sendo assim, a lista foi mantida para que o usuário possa editar as colunas previamente inseridas na tabela, ao realizar um clique longo no item desejado, gerando um *popup* para a sua alteração (edição ou exclusão), como mostra a Figura 4.5.

O funcionamento da edição é a mesma vista na tela de tabelas, ou seja, apenas o nome pode ser trocado. Isso acontece para evitar que o usuário acabe editando chaves primárias, alterando os tipos de dados (gerando eventuais inconsistências com o que já está previamente inserido) dentre outras problemáticas em relação à consistência e à correção de um banco de dados. No que se refere à exclusão de uma coluna, o funcionamento é o mesmo, ou seja, ela apenas é devidamente excluída.

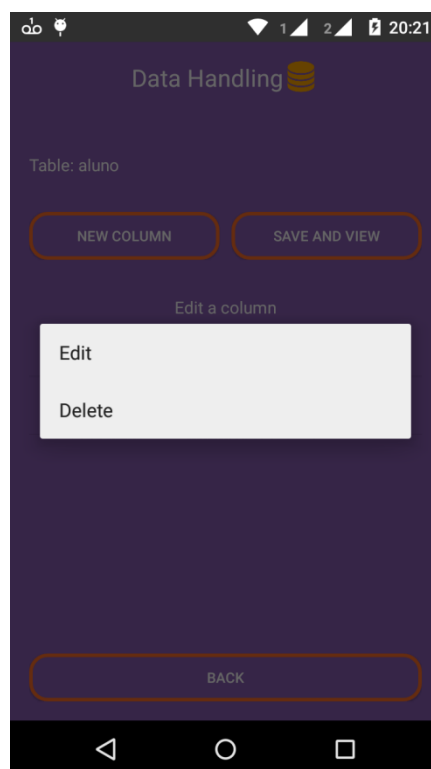
Mais uma vez o objeto Alteração é utilizado junto a essas funcionalidades, contendo os campos necessários para atualização ou exclusão da coluna, para ser tratado futuramente. Este processo será explicado mais detalhadamente nesta seção, pois existem diferenças no tratamento em relação a tabelas e colunas.

Figura 4.4- Listagem de colunas.



Fonte: O Autor.

Figura 4.5- Opções de alteração para colunas.



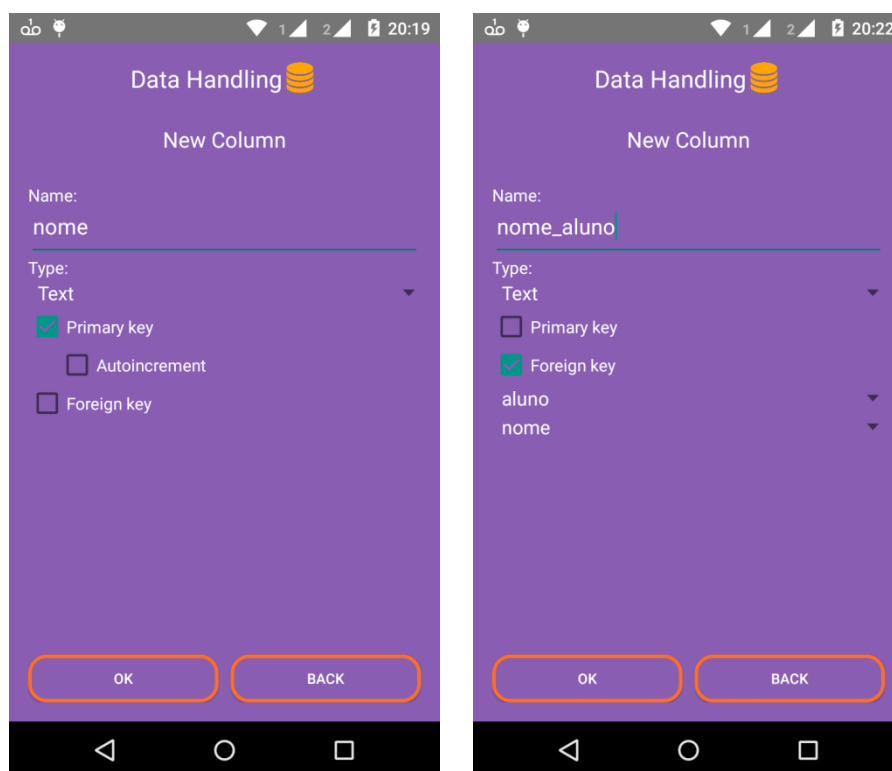
Fonte: O Autor.

Na tela de criação de novas colunas, apenas os dados necessários são exibidos, e em caso de escolha de certas opções, novas possibilidades aparecem, como ilustra a Figura 4.6. Aqui se pode notar que o usuário deve colocar o nome da coluna e escolher o tipo dos dados que serão armazenados nela. Em caso de escolha do tipo “BLOB”, ele ainda deverá selecionar qual dos formatos multimídia essa coluna aceitará, fazendo o campo escolhido aparecer dinamicamente com o clique na opção desejada. Ainda levando em conta a simplicidade visual, as opções abaixo da escolha do tipo também reservam comportamentos diferenciados ao serem selecionados.

Ao selecionar o campo “*primary key*” uma nova opção surgirá, e ela aponta a necessidade ou não deste campo em criação ser uma chave primária auto incremental, ou seja, uma chave do banco de dados em que não existe a necessidade de fazer inserções, pois o SQLite por si só alimenta esse campo, acrescentando seu valor em 1 a cada novo dado adicionado ao repositório. A aplicação permite ainda a criação de chaves compostas, desde que nenhuma chave auto-incremental tenha sido criada antes, apenas inserindo novos campos com a opção “*primary key*” selecionada, uma vez que o sistema verifica a quantidade de chaves e as trata no momento da criação.

Ainda nesta tela, existe o campo “*foreign key*”, que indica se a coluna em questão faz referência a alguma coluna do banco de dados. Em caso de escolha desta opção, é possível selecionar a tabela a qual ela pertence e, por fim, a coluna propriamente dita.

Figura 4.6- Opções durante a criação de colunas.



Fonte: O Autor.

Terminadas as configurações iniciais do banco de dados, o usuário já consegue ter acesso à visualização dos seus dados, como mostra a Figura 4.7. Nestas telas, os dados são exibidos no formato de tabela, com a sua primeira linha contendo os nomes das suas colunas. Uma característica importante da referida tabela é que ela tem a capacidade de, dinamicamente, saber se o usuário necessita de mais espaço para comportar seu conteúdo. Em caso positivo, o sistema habilita o deslize da tabela, tanto na vertical (no caso de muitas linhas inseridas) como na horizontal (caso existam muitas colunas e, portanto, não sendo a largura do dispositivo grande o suficiente para exibir todas elas).

Este comportamento só foi possível de ser implementado com uso de alguns tipos de *layouts* da plataforma Android. O primeiro deles é o *TableLayout*²², o qual cria um ambiente de visualização tabular para o conteúdo que se deseja trabalhar. Com ele, foi possível ajustar e inserir, célula a célula, o conteúdo vindo do banco de dados, mas ainda com a limitação de ser algo estático e que não conseguiria por si só acomodar-se na tela em caso de excesso de conteúdo. Para fornecer a dinamicidade necessária para a tabela, mais dois tipos de *layouts* foram utilizados: *ScrollView*²³ e *HorizontalScrollView*²⁴. O *TableLayout*, no sistema, está

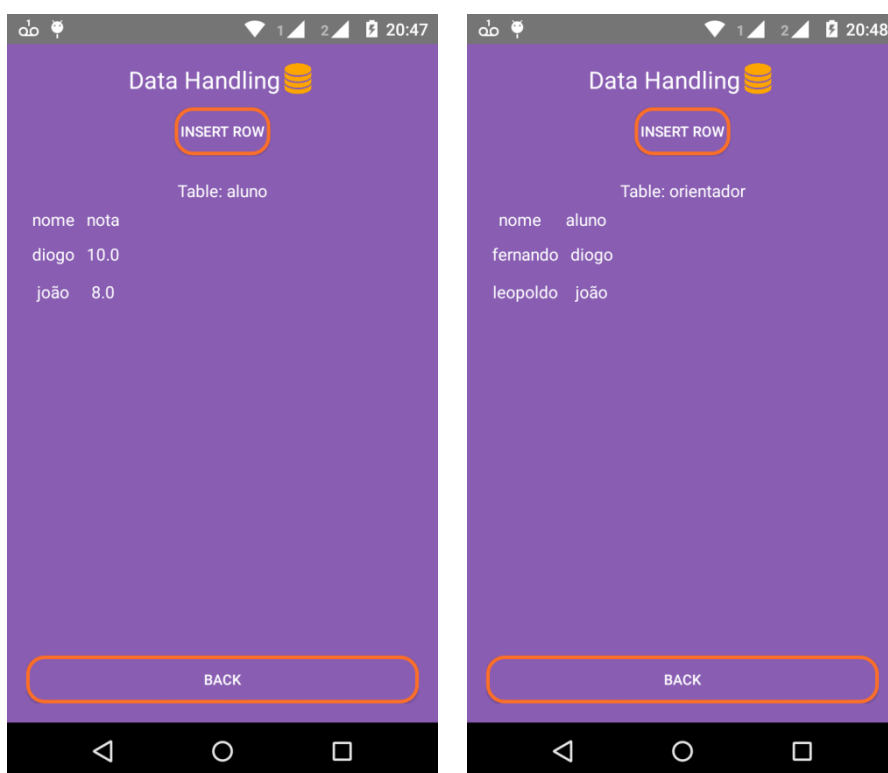
²² <http://developer.android.com/reference/android/widget/TableLayout.html>

²³ <http://developer.android.com/reference/android/widget/ScrollView.html>

inserido dentro do *ScrollView*, para assim permitir a rolagem vertical da tabela, mas apenas de seu conteúdo, ou seja, a primeira linha com o nome das colunas fica fixa para facilitar a visualização. Este processamento é permitido porque essa linha é tratada de forma diferente, de uma maneira que ela esteja separada do restante da tabela. Para a rolagem horizontal, toda a tabela (incluindo a primeira coluna) foram colocados dentro do componente *HorizontalScrollView*, que permite especificamente esse tipo de movimento na tela, de forma automática em caso de necessidade de mais espaço.

Ainda na tela de visualização de dados, outro ponto importante a ser observado é que além da indicação de qual tabela essa exibição pertence, ela também possui um botão para inserção de novos dados. Isso acontece mais uma vez por questão de espaço e da existência de diferentes telas para diferentes focos.

Figura 4.7- Tela de visualização de dados



Fonte: O Autor.

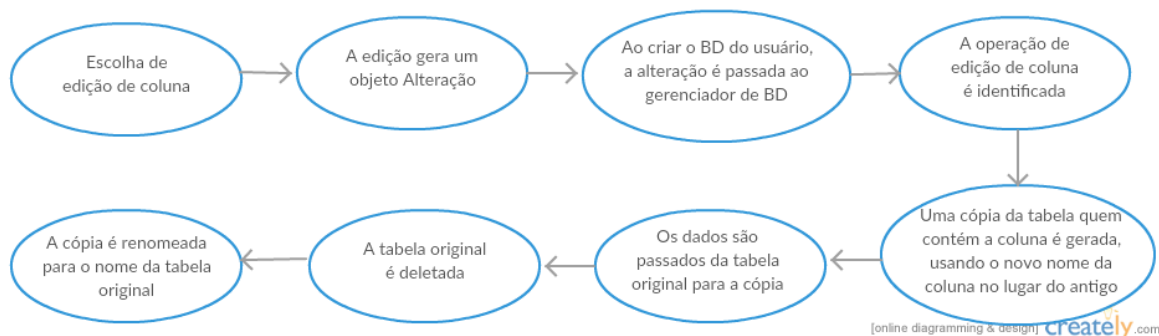
No momento da abertura da tela, é necessário garantir que o banco de dados do usuário seja finalmente criado, de acordo com as configurações passadas das telas anteriores. Existem duas abordagens neste ponto: a primeira vez que o banco de dados é criado (com suas tabelas e colunas) e as alterações no banco de dados, estando ele previamente criado. No

²⁴ <http://developer.android.com/reference/android/widget/HorizontalScrollView.html>

primeiro cenário, a classe responsável pelo gerenciamento do repositório identifica que não existe uma instância criada e a cria, montando os comandos SQLite dinamicamente para saber como se comportar quando se deparar com cada tipo diferente de configuração possível. Já no segundo caso é identificada a existência do banco de dados e, assim, são tomadas as providências necessárias para atualização do seu esquema, identificando quais alterações devem ser feitas por meio de uma lista de objetos Alteração, que fornece todo o necessário para que isto aconteça.

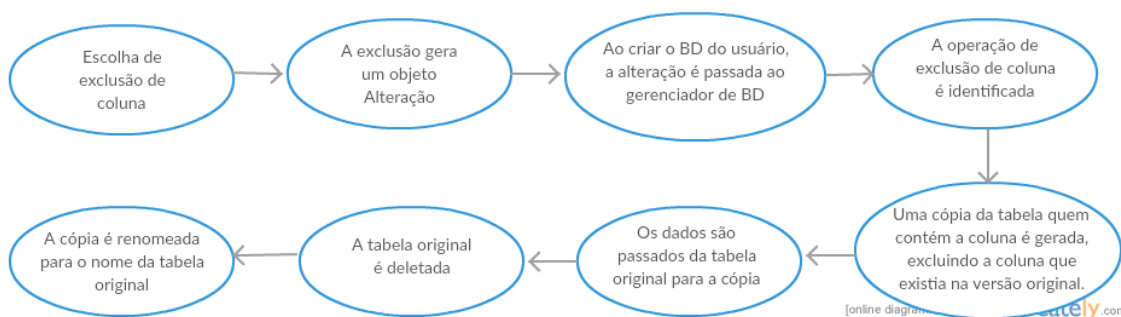
Uma preocupação em relação à alteração de esquema era a persistência de dados previamente inseridos no repositório, pois é interessante que o usuário continue com seus dados antigos para não ter que recomençar seu trabalho do início. As alterações de tabela não acarretam perda de conteúdo, pois elas foram executadas com comandos que propiciam sua manipulação mantendo as colunas intactas. Já nas alterações de coluna, apenas a operação de adição de nova coluna não gera modificações nos dados, pois para ela existe um comando isolado que propicia isso. Entretanto, as edições de esquema relacionadas à edição de nome e exclusão da coluna geram a necessidade de um tratamento especial, pois elas podem impactar onde já existe um conteúdo inserido. A medida adotada nesses casos está ilustrada nas Figuras 4.8 e 4.9, que mostra o fluxo para ambas as operações.

Figura 4.8- Fluxo de funcionamento de edição de colunas.



Fonte: O Autor.

Figura 4.9- Fluxo de funcionamento da exclusão de colunas.



Fonte: O Autor.

Na exibição dos dados, a maioria deles se adequa ao formato em que são apresentados, pois se tratam de textos e números. Porém, um tratamento especial foi necessário para os dados do tipo multimídia, porque a forma de exibição adequada deles é completamente diferente, tanto em relação às suas dimensões quanto ao seu conteúdo em si. No aplicativo PortoDB (que trata apenas imagens), as imagens eram colocadas diretamente na tabela de visualização, mas isso gerava um problema de tamanho das linhas inseridas, pois exigia muito espaço para essas células específicas e desperdiçava espaço nas que não tinham essas mesmas dimensões. Para cada um dos formatos de mídia com suporte no sistema (imagem, vídeo e música) foi preciso buscar formas diferentes de execução, e, para isso, mais uma vez foi realizada uma busca ao que o Android fornecia. Essa plataforma possui aplicações básicas de reprodução desses tipos de dados, e existem meios de abri-las dentro de outras aplicações que necessitem de seus serviços. Por meio dessas chamadas a outros sistemas, foi possível colocar apenas nas células destinadas às colunas do tipo “BLOB”. Há um botão representando esse tipo de dado e, ao clicar nele, o usuário será levado a outra aplicação onde poderá reproduzir sua mídia específica para aquela linha da tabela. A Figura 4.10 mostra a forma de visualização explicada acima.

Figura 4.10- Tela de visualização de dados com coluna do tipo BLOB.



Fonte: O Autor.

Clicando no botão relacionado à inserção de um novo dado, a tela referente a esta funcionalidade é exibida (Figura 4.11). Neste ponto, se fez mais uma vez necessário o tratamento dinâmico em relação à construção da tela, pois, como são colunas escolhidas pelo usuário (em relação aos seus nomes e tipos de dados) o sistema precisa saber quando deve possibilitar certos tipos de entrada para cada uma delas.

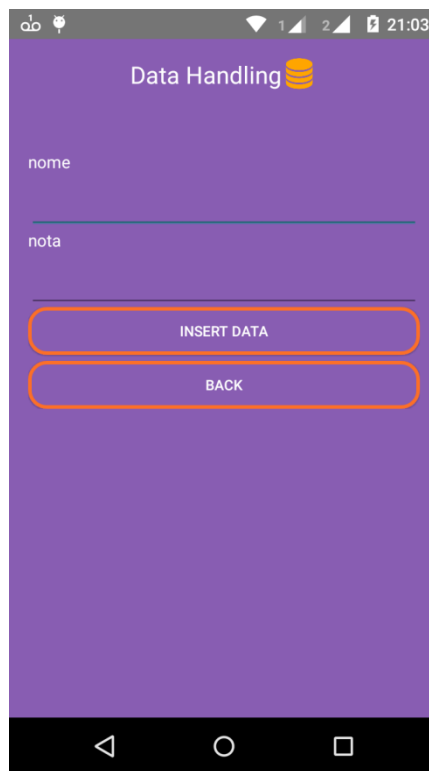
Para tornar a aplicação mais simples, tanto do ponto de vista do desenvolvimento quanto da interface, novos componentes do Android foram explorados. O primeiro deles foi o teclado da plataforma, e seus tipos diferentes de dados de entrada, pois como as colunas podem receber dados textuais ou numéricos, o atributo *InputType*²⁵ do componente *EditText*²⁶ foi escolhido para esses casos específicos. Outro componente utilizado foi o *DatePicker*²⁷, que é responsável pela escolha da data desejada por meio de uma interface similar a um calendário comum, tornando simples o aprendizado de utilização. Por fim, o tratamento feito para a inserção de dados multimídia foi específico para cada possibilidade, ou seja, fotos, vídeos e músicas, porque para cada um deles o local de onde o usuário pode buscar seu dado de entrada é diferente, como mostra a Figura 4.12. Para fotos, a galeria do Android é aberta ao usuário clicar no botão relacionado à escolha (com restrição de exibição de conteúdo apenas para as extensões de imagens). Para vídeos, essa mesma galeria é usada, mas com a restrição de exibição configurada para permitir apenas arquivos com extensões de vídeo. Sobre as músicas, a pasta do sistema que armazena os arquivos de som é exibida, possuindo a opção de ouvir o que está sendo selecionado. Após cada escolha de tipo multimídia o usuário pode ver ou ouvir o que foi escolhido, para oferecer conforto na utilização da funcionalidade em caso de ele esquecer ou desejar rever o que será inserido.

²⁵ <http://developer.android.com/intl/pt-br/reference/android/text/InputType.html>

²⁶ <http://developer.android.com/intl/pt-br/reference/android/widget/EditText.html>

²⁷ <http://developer.android.com/intl/pt-br/reference/android/widget/DatePicker.html>

Figura 4.11- Tela de inserção de dados.



nome

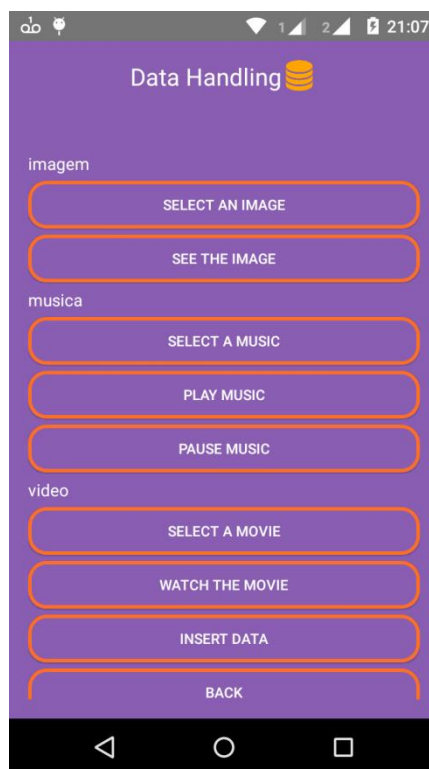
nota

INSERT DATA

BACK

Fonte: O Autor.

Figura 4.12- Tela de inserção de dados com colunas do tipo BLOB.



imagem

SELECT AN IMAGE

SEE THE IMAGE

musica

SELECT A MUSIC

PLAY MUSIC

PAUSE MUSIC

video

SELECT A MOVIE

WATCH THE MOVIE

INSERT DATA

BACK

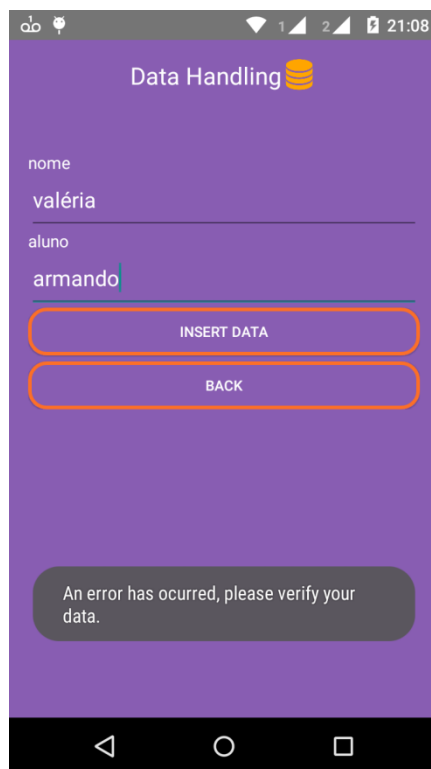
Fonte: O Autor

Ao clicar no botão para inserir os dados, o sistema coleta o conteúdo de cada campo preenchido pelo usuário e passa para o gerenciador de banco de dados, para que ele possa construir os comandos de inserção e executá-los. Algumas particularidades foram adicionadas para garantir o bom funcionamento da aplicação, dentre elas as conversões de dados, que são requeridas para que os dados que serão armazenados não sejam passados como apenas texto (caso comum nos casos de preenchimento de valor numérico em campos para digitação do usuário ou na coleta da data gerada pelo componente *DatePicker*), e a utilização do caminho dos arquivos multimídia para armazenamento do mesmo, por conta de uma limitação do componente *Cursor*²⁸ do Android. Um *Cursor* é responsável por armazenar em memória os dados fornecidos como resultado de uma consulta *SQLite*, e possui como uma de suas características uma capacidade máxima de armazenamento de 2MB e não foram encontrados meios de aumentá-la. Isso se torna um empecilho para recuperar imagens, vídeos e músicas mesmo que inseridas com sucesso, pois normalmente, por conta da melhoria das tecnologias de captura e processamento, eles acabam ultrapassando esse limite de capacidade. Por essa razão, foi optado por realizar a inserção utilizando o caminho do arquivo ao invés dele como um todo, reduzindo o processamento de tratamento desses arquivos. Uma problemática que essa estratégia pode gerar é a exclusão da mídia no caminho especificado para o banco, por fora da aplicação, ocasionando uma falha visto que o sistema não atenta para ações realizadas fora de seu contexto.

Para evitar que o sistema falhe gravemente, caso o usuário insira algum tipo de dado problemático (como chaves repetidas ou referências inexistentes para outras colunas), a aplicação identifica onde deve ter um cuidado com os dados, e, ao gerar um erro, ele pede para que o usuário verifique os dados a fim de alertá-lo sobre algum problema cometido, como mostra a Figura 4.13.

²⁸ <http://developer.android.com/intl/pt-br/reference/android/database/Cursor.html>

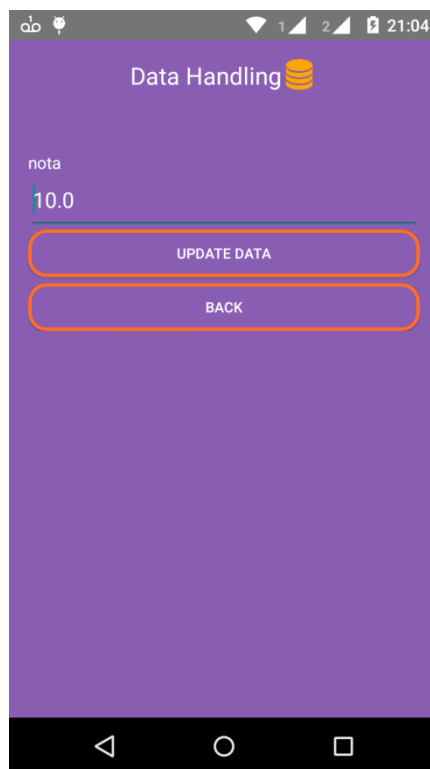
Figura 4.13- Mensagem alertando o usuário sobre problemas na inserção.



Fonte: O Autor.

Na tela de visualização de dados, ao pressionar por algum tempo uma linha de dados, um *popup* aparece, da mesma forma que em outras listas comentadas anteriormente, com as opções de edição e exclusão. Ao selecionar a edição de uma linha da tabela, uma tela semelhante à de inserção aparece, com a diferença que todos os campos relativos às colunas aparecem preenchidos com os dados que o usuário estava visualizando na tela anterior (Figura 4.14), para que ele possa editar como desejar quaisquer trechos da linha. Ao finalizar a edição, as mesmas verificações de dados problemáticos realizados na inserção se repetem aqui, para garantir que não haja inconsistência. Ao clicar na opção de exclusão, a linha é diretamente excluída, caso ela não seja referenciada em outra tabela, pois, nesse caso, o sistema avisa que não pode processar a requisição por essa razão.

Figura 4.14- Tela de edição de dados.



Fonte: O Autor.

Apresentadas todas as características e possibilidades do aplicativo é possível realizar as atribuições de notas que foram aplicadas aos sistemas analisados no Capítulo 2, a fim de descobrir se o desempenho dessa nova aplicação é aceitável. Elas serão realizadas e descritas na próxima seção.

4.4. Análise da aplicação

Com todo o sistema devidamente descrito, pode-se mostrar, em relação aos critérios, a nota adquirida em cada categoria. O Quadro 4.1 ilustra o resumo da análise.

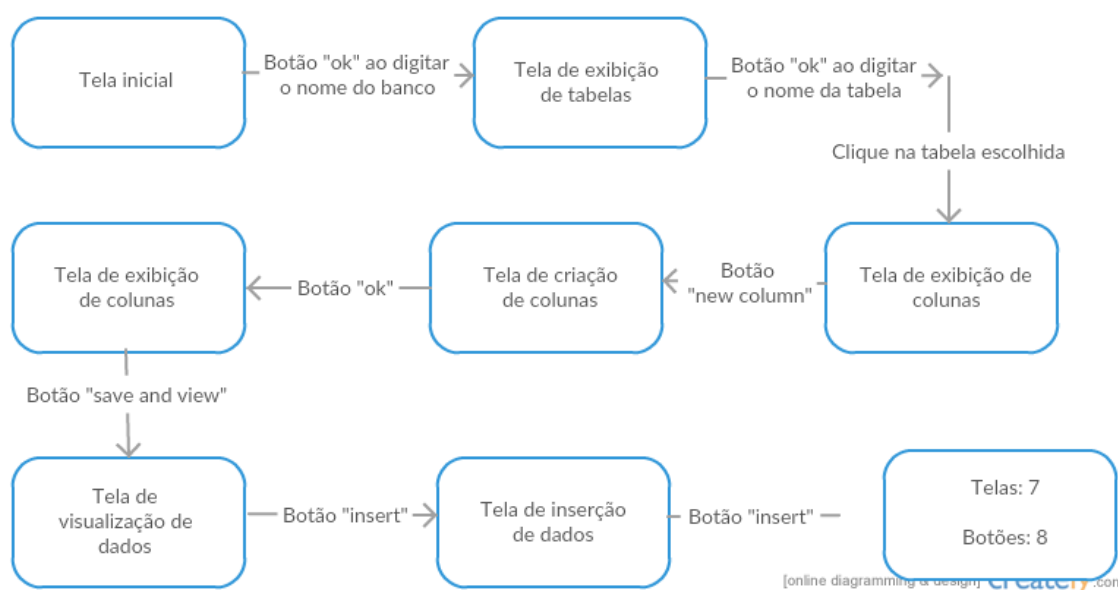
Quadro 4.1- Avaliação final do Data Handling.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
Nota	5	5	5	5

Fonte: O Autor.

Pode-se notar que todos os critérios foram satisfeitos, tornando, como esperado, a aplicação proposta algo que veio para satisfazer as necessidades encontradas em outros sistemas já disponíveis no mercado. Em relação à sua interface gráfica, pode-se ainda ver que ele se equipara aos que tiveram melhor desempenho, ilustrado pelo fluxo, na Figura 4.15, e pelo Quadro 4.2, que mostra a quantidade de telas e botões diferentes utilizados no cenário de utilização inicial até a primeira inserção de um dado.

Figura 4.15- Fluxo de utilização para o Data Handling.



Fonte: O Autor.

Quadro 4.2- Padrão do aplicativo baseado em telas e botões

	Quantidade de telas diferentes utilizadas	Quantidade de botões diferentes utilizados
Data Handling	6	5

Fonte: O Autor.

4.5. Considerações finais

Conclui-se, então, que a aplicação se mostrou satisfatória para o seu propósito, pois, de forma simples e direta, conseguiu trazer o conteúdo desejado para o seu público alvo e corrigiu deficiências presentes em aplicações que são utilizadas no mercado atualmente.

Contudo, ela não é uma aplicação completa em todos os sentidos para o contexto de bancos de dados e, por isso, o próximo capítulo traz, além de uma visão geral sobre o conteúdo apresentado durante todo o trabalho e as limitações da aplicação desenvolvida, sugestões de trabalhos futuros, com o objetivo de aumentar o potencial deste aplicativo.

5. CONCLUSÃO

Ao apresentar um cenário de grande crescimento do volume de dados, é importante que as variáveis que impactam diretamente neste fato sejam conhecidas, pois é a partir desse conhecimento que medidas podem ser tomadas para tratar da melhor forma tamanha quantidade de informação e as suas necessidades envolvidas.

Neste trabalho, ficou claro que, no contexto das aplicações móveis, os usuários estão diretamente ligados ao fato de que grandes quantidades de dados são transferidas todos os dias. Por isso, é importante que os desenvolvedores das aplicações que eles utilizam conheçam a forma mais adequada de montar uma estrutura de dados capaz de satisfazer totalmente os sistemas envolvidos.

Porém, a carência em alguns aspectos em relação aos sistemas disponíveis para o propósito da visualização e manipulação de dados em bancos de dados móveis é um fato. Alguns sistemas estão disponíveis atualmente, mas acabam apresentando problemas como a facilidade de aprendizado e a manipulação de certos tipos de dados específicos. A ideia de desenvolver uma aplicação que suprisse esses pontos de dificuldades foi viabilizada por conta desses fatos citados, no intuito de trazer mais facilidade para os desenvolvedores poderem enxergar o que estavam construindo, criando maior comodidade e podendo trazer resultados mais satisfatórios para eles.

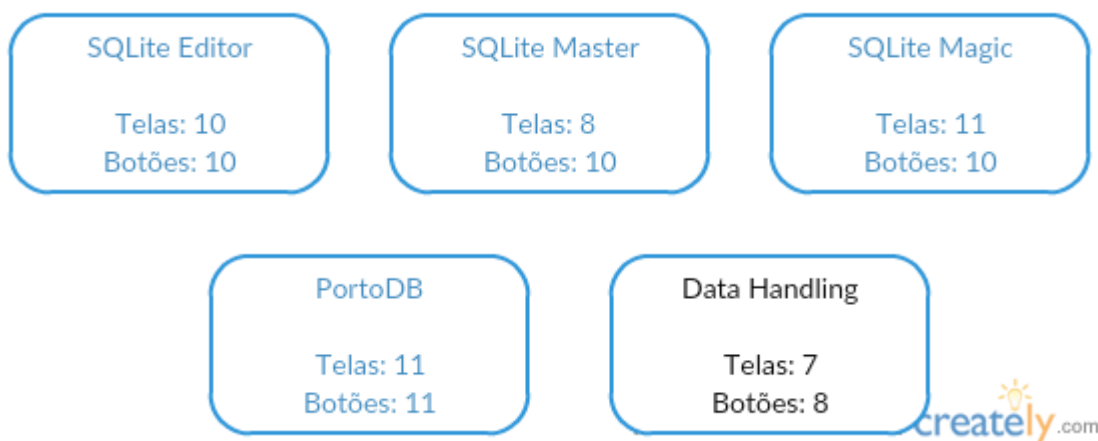
O Data Handling foi concebido sobre a proposta de ter uma interface simples e direta, mas que conseguisse fornecer funcionalidades básicas em se tratando de construção de bancos de dados. Além disso, trazer a possibilidade de utilização de dados do tipo multimídia foi uma meta imposta, para permitir gerar repositórios mais ricos em relação ao seu conteúdo, e com a mesma facilidade de manipulação para os desenvolvedores. Ao desenvolver a ferramenta, métricas puderam ser utilizadas para trazer um comparativo entre esta aplicação e as demais analisadas, como mostram a Figura 5.1 e os Quadros 5.1 e 5.2.

Quadro 5.1- Todas as avaliações realizadas no trabalho.

	Interface Intuitiva	Regras Básicas de Bancos de Dados	Visualização e Manipulação dos dados	Utilização de dados Multimídia
SQLite Editor	3	3	4	1
SQLite Master	3	1	3	1
SQLite Magic	4	1	3	1
PortoDB	5	3	4	3
Data Handling	5	5	5	5

Fonte: O Autor.

Figura 5.1- Resumo do diagrama criado no Capítulo 3 aplicado a todos os aplicativos.



Fonte: O Autor.

Quadro 5.2- Padrão de uso dos aplicativos baseados em telas e botões

	Quantidade de telas diferentes utilizadas	Quantidade de botões diferentes utilizados
SQLite Editor	8	7
SQLite Master	6	8
SQLite Magic	10	9
PortoDB	8	6
Data Handling	6	5

Fonte: O Autor.

No comparativo, o Data Handling se mostrou melhor que os demais, e assim poderá gerar uma melhor utilização por parte do seu público alvo, por não apresentar as carências

citadas anteriormente. Porém, existem alguns pontos de melhorias para ele, em caso de se desejar ter uma quantidade a mais de funcionalidades ou de facilidades para os desenvolvedores ou ainda uma certeza maior de que realmente ele se mostra uma aplicação mais completa.

5.1. Limitações

O Data Handling possui certas limitações que precisam ser descritas, para trazer uma maior transparência para esse estudo. Isto é importante para que se tome o conhecimento necessário da capacidade da aplicação e de tornar mais fácil a visualização de possíveis trabalhos futuros.

Em relação aos tipos de dados multimídia, apenas três deles foram contemplados, limitando as possibilidades de dados caso se deseje utilizar o sistema para reproduzir um banco de dados de uma aplicação mais complexa.

Sobre a interface de visualização de dados, é importante citar que quanto mais colunas e dados são colocados, mais difícil fica o alinhamento entre a linha de descrição das colunas para as linhas de conteúdo, isso por conta das dimensões dos dispositivos que podem executar o sistema, gerando assim dificuldades para repositórios muito complexos serem apresentados de uma forma agradável.

Outra limitação é sobre o tamanho total dos dados por linha em um banco de dados SQLite. Esse problema foi citado no Capítulo 4 quando foi tratada a dificuldade em armazenar dados multimídias diretamente no repositório, mas ainda que sem colunas desse tipo no banco deve-se ter um cuidado com a quantidade de dados armazenados, que não podem exceder o limite de 2MB. Isso por conta do objeto Cursor, responsável pelo retorno dos dados de uma consulta, que possui este tamanho máximo.

5.2. Trabalhos futuros

Como sugestão de trabalhos futuros, primeiramente fica a questão da verificação mais precisa sobre a real melhora de desempenho do Data Handling. A forma como foi realizada essa inferência neste trabalho foi baseada apenas no comparativo com outras aplicações disponíveis no mercado, mas não foram realizados experimentos com o público alvo. Um estudo sobre a utilização da aplicação por parte dos usuários é viável e interessante no nível tanto estatístico, com novas métricas como tempo de realização de tarefas, quanto sobre o objetivo do sistema, recebendo *feedback* de quem utilizou o sistema.

Outra parte interessante a ser trabalhada é a questão de trazer mais facilidades para o usuário. Em se tratando da manipulação dos tipos de dados multimídia, é interessante permitir que o usuário possa não só buscar o conteúdo que deseja armazenar na memória do aparelho, estando ele previamente salvo, mas também que possa gerar o conteúdo durante a utilização da aplicação. Isso é possível pegando dados diretamente a partir da geração de conteúdo da câmera (para fotos e vídeos) e do microfone (para áudio) do aparelho. Outro ponto importante como alvo de melhorias é a questão de aceitação de mais tipos de dados, além de apenas imagens, músicas e vídeos, para dar maior liberdade dentro da aplicação.

Como comentado durante o trabalho, não existem SGBD para dispositivos móveis no mesmo nível de complexidade que existem para outros dispositivos, como computadores *desktop*. Para conseguir alcançar esse nível de aplicação, uma maior robustez precisa ser adicionada ao Data Handling, como por exemplo permitir importar e exportar as bases de dados, dar suporte a triggers²⁹ e a consultas mais complexas (além de apenas visualização de dados por tabelas) via uso da linguagem para o SQLite.

²⁹ http://www.tutorialspoint.com/sqlite/sqlite_triggers.htm

REFERÊNCIAS

- [1] IMASTERS. **Aplicações mobile para quem usa SQL**. Disponível em <<http://imasters.com.br/banco-de-dados/sql-server/aplicacoes-mobile-para-quem-usa-sql>>. Acesso em 14/09/2015.
- [2] RABELO, I., **Usabilidades e suas metas**. Disponível em <<https://irlabr.wordpress.com/apostila-de-ihc/parte-1-ihc-na-pratica/6-usabilidade-e-suas-metas>>. Acesso em 03/10/15.
- [3] PAGANI, T. **O que é usabilidade?**. Disponível em <<http://tableless.com.br/o-que-e-usabilidade/>>. Acesso em 03/10/15.