
Pós-Graduação em Ciência da Computação

GIL CHRISTIANO GUEDES DOS SANTOS

# UM ESTUDO ALGÉBRICO RELACIONAL E DE INTERFACEAMENTO EM APLICAÇÕES SGBDR PARA DISPOSITIVOS MÓVEIS NA PLATAFORMA SQLITE


Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE
2017

**Gil Christiano Guedes dos Santos**

# Um Estudo Algébrico Relacional e de Interfaceamento em Aplicações SGBDR para Dispositivos Móveis na Plataforma SQLite

Este trabalho foi apresentado à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre Profissional em Ciência da Computação.

ORIENTADOR: **Prof. Dr. Fernando da Fonseca de Souza**

RECIFE
2017

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

S237e Santos, Gil Christiano Guedes dos
Um estudo algébrico relacional e de interfaceamento em aplicações SGBDR para dispositivos móveis na plataforma SQLite / Gil Christiano Guedes dos Santos. – 2017.
92 f.: il., fig., tab.

Orientador: Fernando da Fonseca de Souza.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2017.
Inclui referências e apêndice.

1. Banco de dados. 2. Álgebra relacional. I. Souza, Fernando da Fonseca de (orientador). II. Título.

025.04 CDD (23. ed.) UFPE- MEI 2018-077

**Gil Christiano Guedes dos Santos**

**Um Estudo Algébrico Relacional e de Interfaceamento em Aplicações SGBDR para Dispositivos Móveis na Plataforma SQLite**

Dissertação apresentada ao programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre Profissional em 22 de dezembro de 2017.

Aprovado em 22/12/2017.

**BANCA EXAMINADORA**

---

Prof. Dr. Fernando da Fonseca de Souza
Centro de Informática / UFPE
(Orientador)

---

Profa. Dra. Maísa Soares dos Santos Lopes
Universidade Estadual do Sudoeste da Bahia

---

Prof. Dr. Roque Mendes Prado Trindade
Universidade Estadual do Sudoeste da Bahia

*Dedico esse trabalho a meus pais, Gildásio Barreto dos Santos e Maria Cristina Guedes Oliveira, aos meus irmãos Shirley Cristina, Gil Adriano e Gil Luciano e a toda minha família, aos amigos do Mestrado Profissional e do IFBaiano os quais colaboraram especialmente com sugestões valiosas.*

# Agradecimentos

Agradeço em primeiro lugar a Deus pela oportunidade que me concedeu para realizar este grande estudo. Aos meus pais e irmãos por estarem ao meu lado nos momentos mais difíceis desta jornada. Ao meu orientador Professor Doutor Fernando da Fonseca de Souza pela compreensão e disponibilidade diante dos desafios encontrados ao longo deste período.

*A introdução de abstrações adequadas é a nossa única ajuda mental para reduzir o apelo à enumeração, para organizar e dominar a complexidade.*

—EDSGER WYBE DIJKSTRA

# Resumo

As aplicações para *smartphones* estão presentes no cotidiano das pessoas, as quais utilizam-se de inúmeros dados na forma de caracteres e de multimídia para efetivarem suas comunicações. Essas informações, quando persistentes, geralmente ficam disponibilizadas no dispositivo em gerenciadores de bases de dados relacionais. O Sistema Gerenciador de Banco de Dados Relacional (SGBDR) SQLite é atualmente o mais utilizado para aplicações móveis. Destacam-se suas justificativas de usos pela sua gênese no desenvolvimento a partir da estrutura algorítmica fundamental de banco de dados relacionais para servidores de médio e grande porte, e por ser um projeto de código aberto consolidado e com ampla comunidade atuante em melhorias contínuas. Este trabalho aborda a estrutura e a arquitetura em SQLite e os modelos algébricos relacionais em linguagem formal, representando-os conceitualmente por interações das funcionalidades gerais de um SGBDR e como são reproduzidas por modelagem de interfaceamento para a prototipação de uma solução visual para ambiente *mobile*. Como prova de conceito é realizada uma prototipação de interfaces de usuário correlacionadas com seus requisitos funcionais, seus diagramas, eventos reativos e os recursos básicos de interfaces atendidos pelo XCODE do iOS/Apple, no âmbito descritivo.

**Palavras-chave:** Sistema Gerenciador de Banco de Dados Relacional SQLite. Álgebra Relacional. Modelagem de Interfaceamento *Mobile*.

# Abstract

*The applications for smartphones are in the daily lives of people who use numerous data to communicate. This information, when persistent, is usually available on the device in relational database managers. The SQLite Relational Database Management System (RDBMS) is currently the most widely used in mobile application. Its justifications of uses for its genesis in the development from the fundamental algorithmic structure of relational database for medium and large servers stand out, as well as being a consolidated open source project with a broad community active in continuous improvements. This work focusses on structure and architecture in SQLite and relational algebraic models in formal language, represents them conceptually by interactions of the general functionalities of a Relational Database Management System (RDBMS) and how they are reproduced by interfacing modeling for the prototyping of a visual solution for mobile environment. As proof of concept it is proposed a prototyping of user interfaces correlated with their functional requirements, their diagrams, reactive events and the basic interface resources served by the iOS / Apple XCODE, in the descriptive scope.*

**Keywords:** *SQLite Relational Database Manager System. Relational Algebra. Mobile Interface Modeling*

# Lista de Figuras

# Lista de Quadros

# Lista de Tabelas

# Lista de Acrônimos

**ACID** Atomicidade, Consistência, Isolamento e Durabilidade . . . . . 18

**ANSI** *American National Standards Institute* ou Instituto Nacional Americano de Padrões . . . . . 18

**API** *Application Programming Interface* . . . . . 21

**Apps** *Aplicativos* . . . . . 23

**BCNF** *Forma Normal de Boyce-Codd* . . . . . 33

**BLOB** *Binary Large Object* . . . . . 24

**CASE** *Computer-Aided Software Engineering* . . . . . 32

**CVS** *Concurrent Versions System* . . . . . 21

**CRUD** *Create, Read Update e Delete* . . . . . 56

**CSV** *Comma Separated Values* . . . . . 54

**DCL** *Data Control Language* . . . . . 55

**DDL** *Data Definition Language* . . . . . 18

**DQL** *Data Query Language* . . . . . 48

**DML** *Data Manipulation Language* . . . . . 32

**DCL** *Data Control Language* . . . . . 55

**GCC** *GNU Compiler Collection* . . . . . 49

**GCF** *Generic Connection Framework* . . . . . 31

**GUI** *Graphical User Interface* . . . . . 51

**HTTP** *Hypertext Transfer Protocol* . . . . . 85

**HTTPS** *Hyper Text Transfer Protocol Secure* . . . . . 50

**IDE** *Integrated Development Environment* . . . . . 79

**IEEE** *Institute of Electrical and Electronic Engineers* . . . . . 35

**iOS** *Sistema Operacional da Apple Incorporation* para os dispositivos iPhone, iPod e iPad . . . . . 18

**IoT** *Internet of Things* . . . . . 50

**IP** *Internet Protocol* . . . . . 66

**Java ME** Java Micro Edition . . . . . 31

**JSON** *JavaScript Object Notation*. Notação de Objetos Javascript . . . . . 18

**MIDP** *Mobile Information Device Profile* . . . . . 85

**MVC** *Model View Controller* . . . . . 49

**NF** *Normal Form* . . . . . 33

**PaaS** *Platform as a Service*. Plataforma como Serviço . . . . . 25

**PDAs** *Personal Digital Assistants* . . . . . 18

**PDF** *Portable Document Format* . . . . . 54

**QIP** *Quality Improvement Paradigm* . . . . . 50

**RAD** *Rapid Applications Development* . . . . . 45

**RAM** *Random-Access Memory* . . . . . 24

**REST** *Representational State Transfer* . . . . . 21

**RMS** *Record Management Store* . . . . . 88

**SaaS** *Software as a Service* . . . . . 50

**SDK** *software development kit* . . . . . 49

**SFTP** *Secure File Transfer Protocol* . . . . . 66

**SGBDR** Sistema Gerenciador de Banco de Dados Relacional . . . . . 17

**SMS** *Short Message Service* . . . . . 27

**SNS** *Social Networking Service* . . . . . 28

**SQL** *Structured Query Language* ou Linguagem de Consulta Estruturada . . . . . 17

**SSD** *Solid-State Drive* . . . . . 27

**SSH** *Secure Shell* . . . . . 50

**TDD** *Test Driven Development* . . . . . 49

**UFPE** *Universidade Federal de Pernambuco* . . . . . 32

**URL** *Uniform Resource Locato* . . . . . 27

**UTF** *Unicode Transformation Format* . . . . . 35

**UML** *Unified Modeling Language* . . . . . 47

**UnQL** *Unstructured Query Language* . . . . . 28

**XML** *eXtensible Markup Language* . . . . . 21

# Sumário

# 1 INTRODUÇÃO

Os dispositivos móveis atuais com recursos de acessos à Internet – *smartphones* – e seus inúmeros aplicativos se difundiram como soluções para os atendimentos às necessidades humanas quanto às comunicações formais e interações cotidianas em hipermídias de textos, vídeos, áudios e jogos. A maioria dessas soluções, em seus requisitos para os processos de desenvolvimento, requer o armazenamento persistente de dados. O acesso à camada do *Sistema Gerenciador de Banco de Dados Relacional (SGBDR)*, em execução no próprio dispositivo, ocorre por meio de conexões ponto a ponto em estações *desktops* – computadores – o que torna o acesso custoso quanto ao deslocamento e dispendioso de recursos computacionais. Essa forma de acesso *in situ*, ou conectada, expurga a camada de gerência da base de dados do paradigma móvel.

## 1.1 Justificativas

As soluções *mobile* atuais para os *SGBDR*, quando em produção, não apresentaram todas as funcionalidades requeridas para a gestão de bases de dados. A modelagem e a solução de aplicação apresentada por Cabral (2017) conseguiu cumprir as demandas funcionais ausentes nos aplicativos abordados e identificados no decorrer dos comparativos. A pesquisa em laboratório do estudo utilizou-se do *SGBDR* SQLite[1] na camada base para a aplicação de gerência, em produção no Sistema Operacional Android (Informações sobre o SO Android, 2017), deixando-se ausente a dissertação sobre a completude para todas as condições do *SGBDR* SQLite e o respectivo atendimento nas funcionalidades do protótipo, conforme observado *"necessidade de uma quantidade a mais de funcionalidades ou de facilidades para o desenvolvedor"* (Cabral, 2017). Assim, para que seja completa, a aplicação deve esgotar as principais condições das linguagens formais de consulta com suporte pelo SQLite. As aplicações disponíveis para gerência das bases de dados em SQLite são carentes de análises quanto à ergonomia de interfaceamento no âmbito de pesquisa para uma metodologia apropriada e do atendimento às funcionalidades. Como

[1] SQLite é um sistema de banco de dados *Structured Query Language* ou Linguagem de Consulta Estruturada (SQL) de domínio público, independente para aplicar em qualquer propósito, comercial ou privado, de alta confiabilidade, incorporado, completo. Encontra-se disponível em https://sqlite.org e mais detalhes em https://www.sqlite.org/about.html. Último acesso em 24 de outubro de 2017

existem poucas soluções para gerenciamento de bases de dados para o *Sistema Operacional da Apple Incorporation para os dispositivos iPhone, iPod e iPad (iOS)*[2], faz jus a necessidade de um estudo em interfaceamento, a implementação e avaliação por especialista em Tecnologia da Informação, como proposta de solução para esta plataforma, atendendo aos conceitos da literatura acadêmica em banco de dados relacionais e das interações humano-computador.

## 1.2 Motivações

As soluções de persistências de dados em dispositivos móveis evoluem unilateralmente aos gradativos aumentos dos espaços de armazenamentos não voláteis, da velocidade e do paralelismo nos processamentos de dados e das técnicas algorítmicas em banco de dados. Os exemplos das soluções em Sistemas Gerenciadores Palm Pilot-DB (Sobre o Pilot-DB em Referências., 2017) para dispositivos *Personal Digital Assistants (PDAs)* (Palmtops - Personal Digital Assistants), HSQLDB HyperSQL (HSQLDB HyperSQL em Referências, 2017) – dentro dos padrões da *American National Standards Institute ou Instituto Nacional Americano de Padrões (ANSI)*-92[3] SQL com proposta bem completa, escrito na linguagem de programação Java (HyperSQL em Referências, 2017), o qual possui recursos para atender aplicações embarcadas – e o FileMaker (FileMaker em Referências, 2017), como um gerenciador *SGBDR* proprietário, para os produtos da Apple – iPad e iPhone, são amostras distintas das evoluções das técnicas de armazenamento até as atuais estruturas sintáticas *Data Definition Language* (DDL) relacional. Aplicativos para dispositivos móveis com registros de dados embarcados necessitam de suas bases em sistemas gerenciadores seguros e integrados às aplicações, os quais atendam aos requisitos de *Atomicidade, Consistência, Isolamento e Durabilidade (ACID)*[4].

A solução de gerência SQLite – de domínio público[5] – é o gerenciador com notório reconhecimento (*Google-O'Reily Open Source – Awards 2005*[6]), o qual cumpre todos os requisitos *ACID* citado[7], incluindo os principais padrões para indexação, sincronização *JSON*, ocupando pouco mais de 500 kilobytes. SQLite é interoperável para multiplataformas, é indicado para uso na Internet das Coisas[8] e está difundido como o principal gerenciador de banco de dados para o Sistema Operacional Android. SQLite também possui recursos para integrar ao

[2]Apple Inc. Mais informações em `http://www.apple.com/`. Último acesso em 20 de março de 2017.

[3]ANSI-92 SQL – Disponível em `http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt`. Último acesso em 23 de fevereiro de 2017

[4]Informações sobre a ACID no SQLite encontram-se disponíveis em `http://hwaci.com/sw/sqlite/features.html`. Último acesso em 20 de janeiro de 2017.

[5]SQLite – Detalhes da licença de Domínio Público em `http://www.sqlite.org/copyright.html`. Último acesso em 20 de janeiro de 2017.

[6]*Google-O'Reily Open Source Awards* disponível em `https://developers.google.com/open-source/osa/`. Último acesso em 20 de janeiro de 2017

[7]Lista com todas as especificações *SQL* do SQLite, constando o suporte às garantias *ACID* e do *JavaScript Object Notation. Notação de Objetos Javascript (JSON)* para troca (*swap*) de dados, disponível em `http://www.sqlite.org/fullsql.html`. Último acesso em 20 de janeiro de 2017.

[8]Definido como uma solução de Banco de Dados para Internet das Coisas – IoT (*Internet of Things*) em `https://www.sqlite.org/features.html`. Último acesso em 20 de janeiro de 2017

desenvolvimento de aplicações para *iOS* por meio da linguagem de programação Swift3 (Languedoc, 2016), como exemplos das bibliotecas para integrações tem-se SwiftyDB[9] e o FMDB[10]. No escopo dos recursos ofertados no SQLite, observa-se que não há gerenciador nativo para acesso visual *frontend* para *smartphones*. Os sistemas operacionais Android e *iOS* possuem soluções *frontends* isoladas (de terceiros) para gerência de dados, porém não asseguram que suas funcionalidades atendam todas as condições algébricas relacionais cobertas pelo SQLite e que se utilizem das melhores práticas no desenvolvimento das interfaces. Os espectros acadêmicos das fundamentações teóricas em Banco de Dados e as melhores práticas em interfaceamento serão premissas nesta justificativa para o desenvolvimento de uma solução de interfaceamento que atenda aos requisitos funcionais e de modelagem ergonômica visual[11].

## 1.3 Objetivos

Os recursos do *SGBDR* SQLite permitem separar as persistências de dados em uma camada à parte da estrutura de cada aplicação - disponível no mesmo dispositivo - fazendo uso como um gerenciador auto-suficiente. As possibilidades e as restrições deste gerenciador, suas bases algébricas e relacionais e o estudo em interfaceamento serão subsídios teóricos para se aplicarem em desenvolvimento de SGBDR para iPhone.

### 1.3.1 Objetivo Geral

Desenvolver uma estudo algébrico dos recursos do Banco de Dados Relacional SQLite para dispositivos móveis no âmbito do atendimento às funcionalidades e à usabilidade, analisando os recursos funcionais para aplicações gerenciadoras e propor uma solução de interfaceamento para iPhone.

### 1.3.2 Objetivos Específicos

Os objetivos específicos são:

- Pesquisar e discorrer sobre a arquitetura e a estrutura disponibilizadas pelo SGBDR SQLite e seus requisitos funcionais utilizados para o acesso e a manipulação de bases de dados em *smartphones*;
- Analisar as fundamentações algébricas para consultas relacionais que podem ser ofertadas pelo SQLite;

[9] Referências disponíveies em `http://oyvindkg.github.io/swiftydb/` e em `https://www.appcoda.com/swiftydb/`. Últimos acessos em 02 de novembro de 2017.

[10] Referência disponível em `https://github.com/ccgus/fmdb`. Último acesso em 02 de novembro de 2017.

[11] A Modelagem Ergonômica em Interface abordará propostas de *design* e da *iOS Human Interface Guidelines* da Apple – `https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/`. Último acesso em 20 de fevereiro de 2017.

- Apresentar as melhores práticas de interfaceamento de SGBDR para iPhone, valendo-se de métricas de *design* de interfaces e de usabilidade;
- Apresentar um estudo descritivo de interfaceamento do SGBDR SQLite vinculado às modelagens dos requisitos funcionais e sua prototipação para o *smartphone* iPhone, destacando os principais recursos de criação de interface ofertados pela plataforma de desenvolvimento XCODE[12]; e
- Realizar a avaliação qualitativa do interfaceamento por meio de especialistas em Tecnologia da Informação.

## 1.4 Estrutura da Dissertação

Além deste capítulo, esta dissertação está organizada da seguinte forma: No Capítulo 2 será apresentada a revisão da literatura contemplando a estrutura e a arquitetura SQLite e as fundamentações conceituais em interfaceamento. O Capítulo 3 descreve os materiais, métodos e procedimentos adotados para realização deste trabalho. O Capítulo 4 apresenta a avaliação qualitativa do método de interfaceamento por especialistas e as análises dos resultados. O Capítulo 5 apresenta as considerações do estudo, a avaliação do método de interfaceamento por especialistas em Tecnologia da Informação, as contribuições, as limitações e os trabalhos futuros. Referências. Apêndice A apresenta uma modelagem conceitual de dados para uma seção de Ambiente Acadêmico.

[12]Informações sobre a plataforma XCODE estão disponíveis em `https://developer.apple.com/xcode`. Último acesso em 02 de novembro de 2017.

# 2

# FUNDAMENTAÇÕES E CONCEITOS

Neste capítulo serão abordados as referências de trabalhos relacionados, a estrutura, a arquitetura, os conceitos elementares em linguagens de definições, manipulações e de consultas de dados do objeto de estudo SQLite. Também serão tratadas as fundamentações dos recursos e das restrições da álgebra relacional para a linguagem SQL do Gerenciador SQLite. Em complemento serão discutidos os conceitos em interfaceamento para embasar as disposições dos *layouts* e as funcionalidades em atendimento aos requisitos.

## 2.1 Trabalhos Relacionados

As camadas visuais para gerência nos *smartphones* são importantes para acessos concomitantes às informações persistentes das aplicações. Essas camadas são essenciais para visualizações, manipulações ou edições de relatórios, *backups*, migrações via *Concurrent Versions System (CVS)* (Sistema de Controle de Versões) (CVS, 2017) e sincronia em tempo real com uso da *Representational State Transfer* (REST) (REST, 2017) *Application Programming Interface* (API), uso da linguagem de marcação *eXtensible Markup Language (XML)* (XML, 2017) ou *JSON* (*JavaScript Object Notation*)[1], com uma base de dados replicada em nuvem – a exemplo das soluções proprietárias *Couchbase* (Couchbase, 2017) *Heroku* (HEROKU, 2017), e *OpenShift* (Openshift, 2017) as quais dão suporte à sincronização de dados entre uma aplicação embarcada e um serviço hospedado na nuvem. Destacam-se nas seções 2.1.1 e 2.1.2 algumas produções acadêmicas em *design* de interface para visualização e manipulação de dados para dispositivos móveis e de técnicas de armazenamento em Banco de Dados locais.

[1]Referências de sugestões de usos para SQLite em `https://www.sqlite.org/features.html`. A notação JSON possui um formato leve de texto para troca de dados, independente e interpretável por linguagens de programação. Encontra-se disponível em `https://www.json.org/json-pt.html`. Último acesso em 15 de novembro de 2017.

### 2.1.1 Visualização e Manipulação de Dados em Dispositivos Móveis

O trabalho de Cabral (2017) apresentado ao Centro de Informática da Universidade Federal de Pernambuco fez a análise sobre as principais aplicações disponíveis no mercado, comparando-as quanto às limitações e à modelagem de interfaces utilizadas para cumprirem as funcionalidades de maneira facilitada e objetiva. A partir do trabalho de Cabral (2017) – permitir uma melhora no desenvolvimento de aplicações de visualização de dados em *smartphones* no contexto de banco de dados, surge o fundamento teórico em *design* de interface, a ser seguido como proposta disponibilizada no repositório GitHub por este trabalho. Foram utilizados no estudo de Cabral (2017) os critérios de escolhas de soluções: quantidade de downloads feitos no *Play Store*, quantidades de avaliações dos usuários e as pontuações médias atribuídas em uma escala de 1 até 5. Os aplicativos pesquisados foram o SQLite Editor[2] (chamado aSQLiteManager, desenvolvido por Andsen[3]), SQLite Editor Master[4] (Desenvolvido por Amos *Mobile*[5]), SQLite Magic[6] (desenvolvido por iPoint Slovakia[7]) e PortoDB[8] (desenvolvido por PortoFarina)[9]. O autor abordou os atendimentos às funcionalidades: 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 cada critério foi estabelecida uma métrica própria do autor, variando no intervalo de 1 a 5, sendo 1 a nota mais baixa e 5 a mais alta. Cabral (2017) analisou as configurações básicas:

1. Chave primária
    (a) Criar tabela sem chave primária
    (b) Criar tabela com duas chaves primárias
    (c) Criar tabela com chave primária composta
2. Chave estrangeira
    (a) Criar coluna com chave estrangeira

[2]Disponível em `https://play.google.com/store/apps/details?id=dk.andsen.asqlitemanager&hl=pt_BR`. Último acesso em 29 de agosto de 2017.

[3]Mais informações e aplicativos em `https://play.google.com/store/apps/developer?id=Andsen`. Último acesso em 29 de agosto de 2017.

[4]Disponível em `https://play.google.com/store/apps/details?id=com.dundastech.sqlitemasterlight&hl=pt_BR`. Último acesso em 29 de agosto de 2017.

[5]Amos *Mobile*: Mais informações em `https://play.google.com/store/apps/developer?id=Amos+Mobile`. Último acesso em 29 de agosto de 2017.

[6]Disponível em `https://play.google.com/store/apps/details?id=air.SQLite.Magic&hl=pt_BR`. Último acesso em 29 de agosto de 2017.

[7]iPoint Slovakia: Mais informações disponíveis em `https://play.google.com/store/apps/developer?id=iPoint+Slovakia` Último acesso em 29 de agosto de 2017.

[8]Disponível em `https://play.google.com/store/apps/details?id=com.portofarina.portodb&hl=pt_BR` Último acesso em 29 de agosto de 2017.

[9]PortoFarina: Mais informações em `https://play.google.com/store/apps/developer?id=PortoFarina&hl=pt_BR`. Últimos acessos em 29 de agosto de 2017

(b) Inserir linha com chave estrangeira sem referência

(c) Editar linha para uma chave estrangeira sem referência

(d) Remover uma linha que é referenciada por outra tabela

3. Campos auto-incrementais

(a) Alterar um dado auto-incremental

(b) Inserir dados num campo auto-incremental

Avaliou, ainda, as condições funcionais para acessos as bases de dados existentes no dispositivo e a capacidade de se criar novas e, após acessos, realizar operações de manipulações e consultas. As interfaces e funcionalidades foram descritas respectivamente como:

1. Tela inicial (tela *default* ou interface de primeiro acesso), botões e funções associadas - acessos aos repositórios, consultas, editores de bases de dados, configurações e *backups*;

2. Tela com a listagem dos repositórios e cada botão para acessos (abrir ou excluir);

3. Tela de manipulação de base de dados, geradores de tabelas, acessos às entidades e as possibilidades de edição de campo ou exclusão;

4. Tela ou caixa em destaque (*pop-up*) com possibilidade de editar cada campo, atualizar o nome do campo, o tipo e as regras, como os valores padronizados ou obrigatoriedade de uso (de ser populado);

5. Tela para comandos de inserções ou consultas de dados, contendo orientações para cada operação SQL; e

6. Tela dos resultados das consultas.

O trabalho abordou as formatações das interfaces gráficas, a quantidade de diferentes telas e botões utilizados e as experiências de usos para os aplicativos pesquisados; esses parâmetros de *design* de interfaces são direcionados a facilitarem as aprendizagens ou apreensibilidade[10]. Quanto às funcionalidades, as interfaces devem ser adequadas às necessidades exigidas quando em produção - em uso - e devem atender a cada fluxo de interação. O estudo apresentou uma aplicação intitulada *Data Handling* como proposta de uma solução abrangente aos atendimentos dos pontos abordados e resolvidos, citados na Tabela 2.1, a partir das análises dos *Aplicativos* (Apps). Foi realizada uma ideação no escopo das funcionalidades e do *design*, gerando um conjunto de ações abordadas nas análises anteriores, listadas como: abrir, criar ou excluir uma

[10]Documento constando o padrão ISO IEC 9126-1 (2001) disponível em `https://www.iso.org/files/live/sites/isoorg/files/archive/pdf/en/isoupdate_september_2015.pdf`. Último acesso em 01 de setembro de 2017.

base de dados; acessar uma base de dados e realizar operações para criação ou exclusão de colunas (em uma interface à parte) com definição do tipo (ênfase ao tipo multimídia *Binary Large Object* (BLOB)), se chave primária ou estrangeira (com ou sem auto-incremento), a condição de obrigatoriedade (nula ou não-nula) e salvar mudanças; visualizar dados de consultas e alterá-las.

**Tabela 2.1:** Avaliação Final obtida pelo *Data Handling*.

| | Interface intuitiva | Regras Básicas de Banco de Dados | Visualização e Manipulação dos Dados | Utilização de Dados Multimídia |
|---|---|---|---|---|
| Nota | 5 | 5 | 5 | 5 |

Fonte: Cabral (2017)

Pode-se concluir que os conceitos do estudo realizado em visualização e manipulação e as métricas com pontuações, para o atendimento às funcionalidades, apesar de não homologadas (até esta data), são bem consistentes e válidas nas fases de implementações e apreciações nos ciclos de versionamentos para interfaceamento *mobile*. Os conceitos descritos em definições algébricas das operações relacionais e as abordagens em interfaceamento são fundamentos para as prototipações das interfaces bem como da usabilidade.

### 2.1.2 Técnica de Armazenamento de Dados Locais em Ambiente Android

O trabalho de Rosa (2015) discriminou as diferentes técnicas de armazenamento local em *smartphone* conseguinte com a análise do desempenho de três soluções: SQLite, *Shared Preferences* e arquivos de textos. O estudo demonstrou no resultado do comparativo por *benchmarking* que, para grandes quantidades de dados, o SQLite se mostrou mais eficiente uma vez implementado sobre o algoritmo da Árvore-B[11], separado por importação para cada tabela e índices do banco de dados. Analisou-se a eficiência energética – consumos – concomitante ao processamento, os resultados contribuíram para a escolha do recurso de persistência SQLite, pois conforme Newman (2004) é o adequado para aplicações com maior volume de dados. *"O mecanismo SQLite pode endereçar arquivos de banco de dados de até 2 Terabytes; entretanto, a restrição do tamanho de um banco de dados é mais provável que seja aplicada pelo seu sistema operacional. Em muitos casos, o limite de tamanho em um único arquivo é de 2 Gigabytes."* A eficiência do SQLite com reduzido consumo energético de processamentos em indexações e as extrações de dados, e a maturidade do projeto, atendendo aos requisitos não funcionais – arquitetura do hardware[12] para *smartphones* – são condições para a escolha do SQLite.

[11] Algoritmo Árvore-B escrito em *ANSI* C – Arquivo de código fonte btree.h e btree.c disponíveis em `https://github.com/endlesssoftware/sqlite3`. Descrição conceitual disponível em `http://www.cs.au.dk/~gerth/emF03/slides/b-trees.pdf` e `http://www.mccreight.com/people/ed_mcc/index.htm`, página pessoal do criador do algoritmo B-Tree. Últimos acessos em 15 de novembro de 2017.

[12] Arquitetura refere-se aos valores dos recursos de frequência e paralelismo de processamento de dados, volume de espaço e de frequência da memória *Random-Access Memory (RAM)* e os recursos de persistências de dados. Informações sobre o iPhone 7 em `http://www.apple.com/br/iphone-7/specs/`. Último acesso em 15 de novembro de 2017

### 2.1.3 Análise Comparativa com os Trabalhos Relacionados

Em suma, o trabalho de Cabral fez uso do comparativo de quatro gerenciadores SQLite (conjunto amostral reduzido). Estes são executados no Sistema Operacional Android e, a partir das avaliações funcionais, obteve-se suas referências para gerar uma aplicação protótipo intitulada *Data Handling*, a partir das interseções das funcionalidades pesquisadas. A proposta deste estudo dissertativo é apresentar o interfaceamento dentro do conjunto das funcionalidades de um SGBDR abrangente ao ambiente *mobile*; realizar o estudo algébrico do SQLite para dimensionar as saídas visuais das consultas relacionais *mobile*, valendo-se das métricas de ambientação de interfaces: as recomendações da empresa Apple e dos pontos essenciais citados por Cabral (2017). Como diferencial, a proposta iDBLite tem caráter conceitual e se valerá do método de refinamentos – nos Ciclos de *Design* da Engenharia de Software Experimental – para os moldes das interfaces. O estudo e a proposta são direcionados para fomentar o desenvolvimento de um SGBDR SQLite *Mobile* com enfoque (a recomendação para desenvolvimento) à linguagem Swift na plataforma XCODE. A Tabela 2.2 apresenta uma comparação dos pontos essenciais abordados, por Cabral (2017) e por Rosa (2015).

**Tabela 2.2:** Correlações entre os Trabalhos

| Critérios / Trabalhos | Técnicas de Armazenamento Rosa (2015) | *Data Handling* Cabral (2017) | Interfaceamento iDBLite |
|---|---|---|---|
| Abordou o Estudo de um Método para o Interfaceamento | NÃO | SIM | SIM |
| Utilizou um critério não-funcional para a escolha do SGBDR SQLite | SIM | NÃO | SIM |

Fonte: Elaborado pelo Autor (2017)

## 2.2 A Estrutura do SQLite

SQLite é um software de código aberto, multiplataforma, escrito em C (Linguagem C, 2017) ANSI (ANSI, 2017), específico para gerência de dados, o qual implementa um sistema de banco de dados SQL embarcado. Programas que usam a biblioteca do Gerenciador SQLite têm acesso a banco de dados SQL sem executar um processo SGBDR separado. O SQLite não é só uma biblioteca cliente usada para se conectar com um grande servidor de banco de dados (atualmente um grande servidor possui um ou mais bancos de dados em sistemas operacionais virtualizados fornecidos por um *Platform as a Service*. Plataforma como Serviço (PaaS) (PaaS Platform-as-a-Service, 2017) geralmente por uma empresa contratada), mas ele - SQLite - é o próprio servidor (Languedoc, 2016). A biblioteca SQLite dispensa configurações, a exemplo das importações de pacotes e *plugins* para sincronizações. Outra vantagem é a velocidade, uma vez que faz leituras e escritas diretamente para o arquivo da base de dados no banco de dados, no disco ou em unidade sólida em *flash*. No contexto histórico, o primeiro *parse* SQL para a

solução SQLite foi desenvolvido por *Dwayne Richard Hipp*[13] e apresentado em 17 de agosto de 2000. A ideia original era suprir as demandas de software embarcado que não necessitavam SGBDR de grande porte.

O SGBDR SQLite se mostrou um dos gerenciadores de código aberto mais difundido[14], estável (versão atual 3.2.10)[15], em uma comunidade de desenvolvimento ativa[16], com arquitetura leve e que atende aos requisitos da ACID. Para elucidar a importância do projeto SQLite, só no decorrer de 2017 ocorreram 15 *releases* - até esta data - incluindo a última com uma melhoria significativa de 2.1% na carga de uso do processador.

> *"O SQLite é uma biblioteca de processo interno que implementa um mecanismo de banco de dados SQL autônomo, sem configuração, sem servidor e transacional. O código-fonte do SQLite existe no domínio público e é gratuito para fins privados e comerciais. SQLite tem suportes para várias linguagens de programação como C, C++, BASIC, C Charp, Python, Java e Delphi. O wrapper COM (ActiveX) que torna o SQLite mais acessível a linguagens de script no Windows, como VB (Visual Basic) Script e JavaScript, adicionando recursos a aplicativos HTML. Ele também está disponível em sistemas operacionais incorporados como iOS, Android, Symbian OS, Maemo, Blackberry e WebOS por causa de seu pequeno tamanho e facilidade de uso"* (Bhosale, 2015) [17].

Desenvolvedores de aplicações *mobile*, seja para Sistema Operacional iOS ou Android, devem fornecer a biblioteca[18] SQLite instalada junto com suas aplicações (*cross-platform*), entre as vantagens desta inclusão estão: não necessitar de dependências externas ao dispositivo e ter compatibilidade com os principais aplicativos[19] e sistemas operacionais[20], a exemplo do UNIX (SQLite para UNIX, 2017), do LINUX (SQLite para Linux, 2017), do Mac OSX (SQLite no Mac OSX, 2017), do Android (SQLite no Android, 2017), do iOS (iOS, 2017) (Languedoc, 2016), do Windows (SQLite no Windows, 2017), WinCE (SQLite para WinCE, 2017) e WinRT (SQLite para WinRT, 2017). Devido a estabilidade e a maturidade de software (CMMI - Sobre Maturidade de Software, 2017), as principais (atuais) empresas (Principais empresas que usam SQLite, 2017) desenvolvedoras de aplicativos para pessoas físicas e soluções corporativas inseriram amplamente a biblioteca SQLite em seus aplicativos[21]. O iPhone[22] – aparelho de referência para o protótipo

[13] Mais informações em `http://www.hwaci.com/drh/`. Último acesso em 10 de março de 2017.

[14] Referência em `https://db-engines.com/en/ranking`. Último acesso em 21 de novembro de 2017.

[15] Histórico das releases disponível em `https://sqlite.org/changes.html`. Último acesso em 21 de novembro de 2017.

[16] A relação das últimas 50 contribuições está disponível em `http://www.sqlite.org/src/timeline`. Último acesso em 22 de novembro de 2017.

[17] *Bhosale*: Mais informações em `https://www.researchgate.net/profile/Satish_Bhosale2`. Último acesso em 01 de fevereiro de 2017.

[18] A biblioteca principal para o desenvolvimento é *sqlite3.c*. Encontra-se disponível em `https://www.sqlite.org/getthecode.html`. Último acesso em 21 de fevereiro de 2017.

[19] Referência do uso do SQLite para os principais softwares para computadores pessoais estão em `https://www.sqlite.org/famous.html`. Último acesso em 01 de outubro de 2017.

[20] Sistemas Operacionais com suportes SQLite citados explicitamente em `https://sqlite.org/download.html`. Último acesso em 01 de outubro de 2017.

[21] Aplicativos com SQLite: http://www.sqlite.org/mostdeployed.html. Último acesso em 01 de outubro de 2017.

[22] Apple iOS: Escolhido por critério de estabilidade do Sistema Operacional e a ampla disponibilidade.

de interfaceamento - produto da empresa Apple, é um *smartphone* que dá suporte ao SQLite como sistema gerenciador de informações - recurso de acesso SQLite disponível na plataforma de desenvolvimento XCODE (Languedoc, 2016), como exemplo de uso no registro de dados nos *Short Message Service* (SMS) (Serviço de Mensagens Curtas, 2017), gravar os registros de chamadas ou informações do emissor/receptor e seus anexos (JEON, 2011). Outra referência é o navegador Safari[23], destinado aos acessos às páginas de Internet, a qual salva seus dados de pesquisas da *cache* no SQLite (Núcleo de Dados para Programação, 2017) – antes fazia-se uso do *Mork Database*[24] – dimensionado conforme aumento das demandas de acessos. Os volumes de históricos e das seções de favoritos aumentavam as cargas de gerenciamentos. A correção para este aumento na carga de processamento se faz por otimização do banco de dados do navegador. As melhorias de desempenho encontram-se divulgadas no *review* da versão Mozilla Firefox 3; para este *browser* fez-se também a troca do método algorítmico *Mork* para o formato de armazenamento em uma base de dados no SQLite. Este Sistema Gerenciador implementa o recurso *mozIStorageService*[25]. Esta substituição resultou em melhorias significativas quanto à responsividade e à redução do uso de memória.

Nos acessos constantes aos *browsers*, qualquer SGBDR que faça uso contínuo incorrerá nos problemas decorrentes dos *swaps* (trocas) de dados, os quais geram espaços vazios. Essas falhas nascem das periódicas demandas de acessos, comuns a um SGBDR e são conhecidas como *data overhead* (sobrecarga de dados). As trocas de dados têm origens nos processos de incluir e remover objetos (índices, registros, tabelas e visualizações). Os ajustes de alocações em espaços vazios ou desfragmentações, apesar de aperfeiçoarem o tempo de novas inserções, criam fragmentações prejudicando o acesso aos dados persistentes (disco, memória *flash* ou *Solid-State Drive* (SSD) *(Solid-State Drive, 2017)*), e ao conteúdo. São similares às lógicas de fragmentação em sistemas de arquivos. O Mozilla Firefox 3, em sua implementação de persistência usa várias bases de dados SQLite com finalidades distintas, destas as principais são: histórico dos endereços *Uniform Resource Locato* (URL)s e favoritos (*places.sqlite*), armazenamento em *cookies* (*cookies.sqlite*), *downloads* (*downloads.sqlite*), buscas e preferências de conteúdos (Pereira, 2009) [26] por usuário. As principais bases que podem levar à estabilidade (ou falta dela) – fragmentações – na aplicação são apenas duas: *places.sqlite*[27] (onde se armazena os favoritos e histórico) *urlclassifier3.sqlite*[28] (dados do filtro *anti-phishing*). *O navegador do Google Chrome* (Suporte no Google Chrome para Web SQL Database, 2017) *faz uso do Gerenciador SQLite para salvar dados na cache, históricos e cookies* (JEON, 2011).

[23] Informações em `http://www.apple.com/br/safari/`. Último acesso em 08 de março de 2017.

[24] Estrutura do Mork: `https://developer.mozilla.org/en-US/docs/Mozilla/Tech/Mork/Structure` e `https://github.com/KevinGoodsell/mork-converter`. Último acesso em 08 de março de 2017.

[25] Referência em `https://developer.mozilla.org/en-US/docs/Mozilla/Tech/XPCOM/Storage`. Último acesso em 25 de março de 2017.

[26] Referência em `http://sqlite.org/mostdeployed.html`. Último acesso em 15 de maio de 2017.

[27] places.sqlite: Mais informações disponíveis em `https://goo.gl/P4nSKJ`. Último acesso em 08 de março de 2017.

[28] urlclassifier3.sqlite: Acesso em `https://goo.gl/tuCgHL`. Último acesso em 08 de março de 2017.

Outro exemplo é o Skype, que utiliza SQLite na versão 3. A base de dados SQLite do Skype encontra-se disponível no arquivo main.db, podendo ser acessada em sqlite3 (no diretório /Library/Application/Support/Skype/«SEU_NOME_DE_USUÁRIO»/main.db), contendo pequenas tabelas(.*schema*) para contatos e mensagens, respectivamente: .tables, .schema e .schema Messages[29] sendo um software de comunicação que inclui chamadas de voz, envios de arquivos e *chat* de texto. Os usuários do Skype usam uma lista com todos os dados de suas comunicações (históricos) os quais são armazenados em bases SQLite. O TweetDeck (TweetDeck - Localização dos dados em SQLite, 2017) é uma aplicação amplamente utilizada pelo Twitter (Rede Social Twitter, 2017) e o Facebook (Rede Social Facebook, 2017) – *Social Networking Service* (SNS) no qual o gerenciador SQLite armazena os contatos e os metadados. Em 2011, Dwayne Richard Hipp[30] anuncia a interface *Unstructured Query Language* (UnQL)[31] para o SQLite – uma nova linguagem de consulta para documentar banco de dados com JSON e inserida no CouchDB (Banco de dados não-relacional (NoSQL) CouchDB, 2017).

## 2.3 A Arquitetura do SQLite

A apresentação da visão esquemática de organização e de implementação da arquitetura[32] do SQLite é essencial para contextualizar (Figura 2.1) e apresenta uma visão didática da arquitetura SQLite. No núcleo do SQLite, os códigos da interface são herdados da linguagem C e são encontrados nos arquivos *main.c*, *legacy.c* e *vdbeapi.c*, embora outras rotinas estejam dispersas em outros arquivos em linguagem C, nos quais se tem acessos às estruturas nativas. A rotina *sqlite3_gettable()* está disponível em *table.c*, e a rotina *sqlite3_mprintf()* encontra-se em *printf.c*, entre outras. O processador de comandos reconhece os *parses*, as palavras reservadas *tokens*, os identificadores criados, as condicionais e os laços estruturados, seguindo os padrões de ordens de precedência e a direção na leitura dos comandos SQL das consultas, das construções ou modificações de tabelas e *views*, gerando os *bytecodes*, os quais são específicos para execuções na máquina virtual (Owens, 2006). O *backend* é a seção do SQLite independente quanto às informações dos algoritmos de acessos às unidades de persistência. Apesar do acesso ao código fonte *backend* está aberto (para desenvolvimento), a implementação dos acessos é facilitada ou transparente ao desenvolvedor pelas chamadas – via nomenclaturas convencionadas – aos métodos das classes. A estrutura esquemática da arquitetura SQLite deve ser conhecimento elementar sobre o objeto de estudo, porém os componentes do núcleo e da implementação do compilador, respectivamente acessos aos setores em hardware via programação C (*Assembler*) e

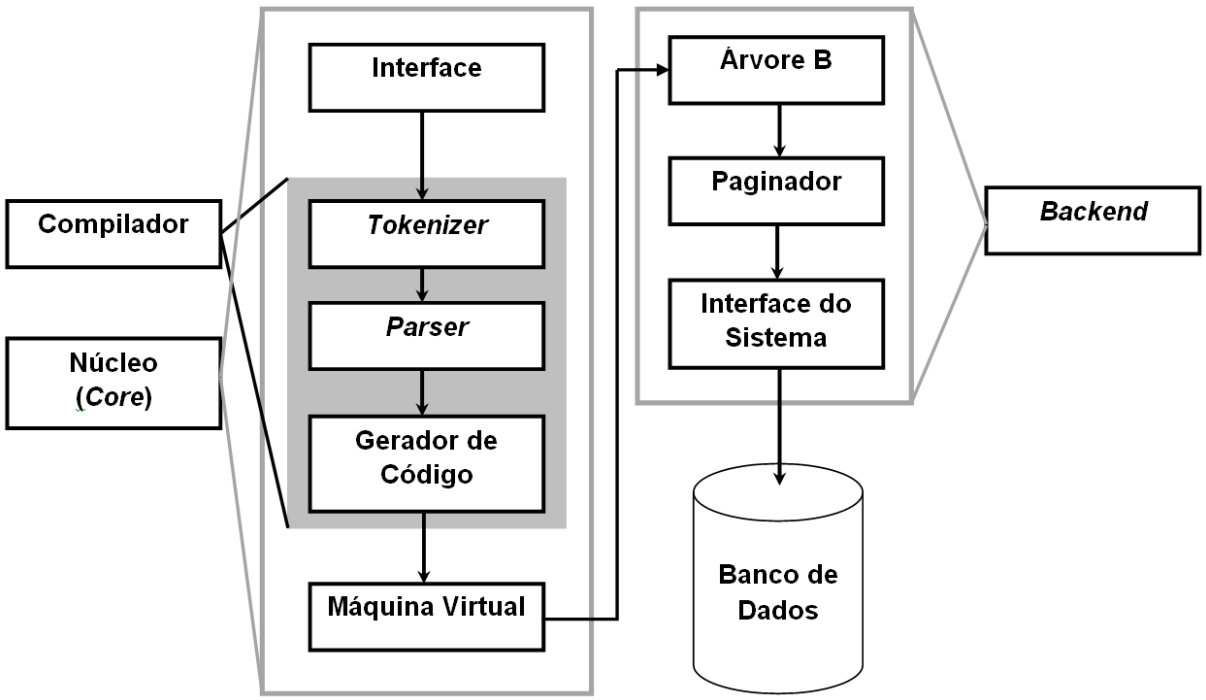
[29] Referências em `https://coolaj86.com/articles/searching-skypes-sqlite-database/`. Último acesso em 10 de novembro de 2017.

[30] A página ofical do Doutor *Dwayne Richard Hipp* encontra-se no endereço `http://www.hwaci.com/drh/`. Último acesso em 11 de novembro de 2017.

[31] Informações disponíveis em `https://www.infoq.com/news/2011/08/UnQL`. Último acesso em 11 de novembro de 2017.

[32] Disponível em `https://www.sqlite.org/arch.html`. Último acesso em 14 de fevereiro de 2017

**Figura 2.1:** Esquemático da Arquitetura SQLite




Fonte: `https://www.sqlite.org/arch.html`

a análises léxica[33] e semântica (Price, 2006), abrangem as estruturas em baixo nível em acessos a disco e da teoria da computação (Lewis, 2000), deixando margens para pesquisas futuras em incrementos e ou alterações nas bases algorítmicas SQLite, as quais estão além do escopo deste trabalho.

## 2.4 SQLite: Limitações dos Recursos Funcionais e Não-Funcionais

Os recursos ofertados pelo SQLite descendem do padrão SQL92[34] e, pela proposta de otimização do uso dos recursos de processamento de dados como garantia da portabilidade para aplicações embarcadas, seguem as operações relacionais - *queries* - nas bases de dados que não possuem suporte[35], como descrito no Quadro 2.1 com as *Limitações funcionais do SQLite*. No âmbito das limitações não-funcionais, atualmente há restrições às capacidades máximas em volumes de *bytes* para tipos e quantidade de tabelas, limites de linhas e colunas que não podem

[33] Análise Léxica: *"O analisador léxico ou scanner tem como função fazer a leitura do código fonte, caractere a caractere, e traduzi-lo para uma sequência de símbolos léxicos (tokens)". (Price, 2006)*

[34] Disponível em `http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt`. Citado também na referência 1.2. Último acesso em 22 de setembro de 2017

[35] Informações disponíveis em `http://www.sqlite.org/cvstrac/wiki?p=UnsupportedSql`. Último acesso em 10 de março de 2017.

ser excedidos, entre outros. Nas primeiras versões[36] do SQLite não havia um limite superior definido para volume de armazenamento, ficando restrito aos limites de espaço da memória persistente do dispositivo.

O Gerenciador SQLite, em sua concepção, foi definido sem ter uma medida de limites superiores de espaços em memória, uma vez que as máquinas possuem memórias finitas, este fato provocou erros por saturação de dados. Estes problemas de definições dos limites foram testados e encontram-se corrigidos nas versões atuais.[37] No decorrer da referência citada encontram-se mais 15 (quinze) tópicos com informações úteis sobre: o volume máximo de dados para variáveis do tipo *String* e tipo BLOB (Detalhes sobre a implementação do tipo BLOB, 2017), para o número máximo de colunas, para o comprimento máximo de uma instrução SQL, para o número máximo de tabelas em uma associação, para a profundidade máxima de uma árvore de expressão, para o número máximo de argumentos em uma função, para o número máximo de termos em uma instrução *SELECT* composta, para o comprimento máximo de um padrão *LIKE* ou *GLOB* (Detalhes sintáticos para os usos dos tipos LIKE e GLOB (2017); Owens (2006)), para o número máximo de parâmetros de *host* em uma única instrução SQL, para a profundidade máxima de Recursão de *triggers*, para o número máximo de bases de dados anexadas, para o número máximo de páginas em um arquivo de banco de dados, para o número máximo de linhas em uma tabela, para o tamanho máximo do banco de dados e para o número máximo de tabelas em um esquema.

Em termos da estrutura física, limita-se o espaço da quantidade de bytes para armazenamento de variáveis como exemplos dos tipos primitivos *String* e, com enfoque ao tipo *BLOB* - trata-se, além da quantidade, do espaço de alocação vetorial de bytes para armazenar caracteres de *strings* e multimídias como: figuras, áudios e vídeos; como alternativa foi implementado pelos desenvolvedores do SQLite a possibilidade de definir explicitamente um limite superior; a solução permite alterar em tempo real - via método sqlite3_limit() - por meio de uma interface para a categoria limite com enfoque às aplicações em múltiplas bases de dados. *Para customizar ou alterar a largura máxima de uma string ou de um BLOB em SQLite define-se o valor do macro de pré-processador SQLITE_MAX_LENGTH. A implementação atual suporta uma string ou BLOB com largura máxima de 231-1 ou 2147483647. A partir desta macro pode-se definir o limite máximo para o número de colunas - permissão para customizar a quantidade - os limites de uma declaração SQL, números de tabelas em uma JOIN - Junção, limitar explicitamente a profundidade máxima de uma árvore de expressão, a quantidade máxima de argumentos de uma função, definir o número máximo de termos em uma declaração SELECT, das extensão de um LIKE ou GLOB, definição do tamanho máximo de uma base de dados (140 terabytes) e o número máximo de tabelas de um Schema*[38], entre outros limites já citados nesta seção. Os

[36]Detalhamento das versões disponíveis em `https://sqlite.org/changes.html`. Último acesso em 10 de março de 2017

[37]Referências no *website* principal `http://sqlite.org/limits.html`. Último acesso em 10 de março de 2017.

[38]Limites Físicos em SQLite: Mais informações sobre os limites quantitativos binários que ocupam em memória

**Quadro 2.1:** Limitações Funcionais do SQLite

| Funcionalidade | Descrições |
|---|---|
| Suporte a *Trigger* | Triggers com suporte de linha – FOR EACH ROW – mas não os de comando – FOR EACH STATEMENT triggers. |
| RIGHT, OUTER JOIN | Somente *LEFT OUTER JOIN* é implementado |
| *FULL OUTER JOIN* | **Somente *LEFT OUTER JOIN* é implementado**. |
| ALTER TABLE | Os recursos para *RENAME TABLE* e *ADD COLUMN* e as variantes para os comandos *ALTER TABLE* podem ser usados. Comandos para *DROP COLUMN, ALTER COLUMN* e *ADD CONSTRAINT* não podem ser usados. |
| *VIEW* | Visualizações em SQLite são somente para leitura. O desenvolvedor não pode executar declarações *DELETE*, *INSERT* ou *UPDATE* sobre uma visualização. |
| *GRANT* e *REVOKE* | O único tipo de acesso permitido que pode ser aplicado é o de acessos normais aos diretórios de arquivos do sistema operacional |
| *STORED PROCEDURE* | Não há suporte para *Stored Procedure* (Procedimentos Armazenados). Trata-se de rotinas – procedimentos – que são armazenadas na base de dados. Uma vez que o SQLite é disponibilizado junto às aplicações, as vantagens dos *Stored Procedures* se descontextualizam. |

Fonte: Elaborado pelo Autor (2017)

recursos físicos não-funcionais são fundamentais quanto aos destinos dos projetos das aplicações, devendo o administrador de banco de dados dimensioná-las conforme a estimativa da quantidade de transações relacionais e da saturação em disco.[39]

## 2.5 Linguagem de Definição e Manipulação de Dados no SQLite

As persistências dos dados são premissas vitais dos sistemas gerenciadores de banco de dados, pois com o aumento dos recursos ofertados pelas aplicações embarcadas, exigiram-se novas técnicas de armazenamento e acessos. Como exemplos cronológicos das evoluções das técnicas, tem-se a solução de armazenamento em registros sequenciais utilizada pela API *Persistent Storage* da Java Micro Edition (Java ME) (Persistência de dados em JavaME, 2017) a qual faz uso de uma API de persistência orientada a registros que, por definição algorítmica em seu projeto, não possui os recursos avançados de um SGBDR, como exemplos dos atendidos pelo SQLite. No Java ME as informações podem persistir no dispositivo ou, com incremento de técnica via *framework* de Estrutura de Conexão Genérica (*Generic Connection Framework* (GCF)),

estão em `http://www.sqlite.org/limits.html`. Último acesso em 29 de setembro de 2017.

[39]Mais informações em `https://www.sqlite.org/fileformat.html`. Último acesso em 25 de março de 2017.

podem espelhar com um *backend* de um servidor pelo protocolo HTTP 1.1, via *HttpConnection* (Conexão Genérica Java ME (2017); Muchow (2004)); trata-se de um exemplo funcionalmente semelhante às trocas de dados em SQLite via JSON[40] usando REST API para sincronização com um servidor, a exemplo do Node.js (Sobre o Node JS, 2017). Para realizar a sincronização, o *plugin* Node responde aos acessos concorrentes de forma escalável uma vez que não necessita abrir novas conexões para as chamadas ao servidor. Para suportar as demandas de conexões de usuários, esse *plugin* troca os acessos via OS *thread*[41] por eventos do próprio *engine* do Node. Como não há bloqueios de processos, então nunca resultarão em *deadlocks*, assim pode-se garantir acessos às dezenas de milhares de conexões simultâneas. O *engine* do Node usa o motor *JavaScript V8*[42], tradutor escrito em linguagem C++ (Referências sobre a Linguagem C++, 2017) pela empresa Google[43] que fica em produção no lado servidor. Node.js usa a programação orientada a eventos ou seja, chamadas e retornos são geridos por eventos. No lado servidor os dados ficam disponíveis para acessos pelos usuários por meio de *frontends* em aplicativos em *smartphones*, em *browsers*, entre outros que tenham esses suportes.

As estruturas das bases de dados são os espaços pré-definidos por meio de tabelas fundamentais e de relacionamentos para inserções de registros. Estas estruturas podem ser implementadas com níveis de dependências hierárquicas, diante destas relações diretas o projetista de banco de dados pode declarar explicitamente as exclusões dos dados em cascatas (como exemplo, excluindo os dados de uma empresa, todas as tabelas associadas - informações sobre os insumos, bens patrimoniais, funcionários, entre outros - serão excluídos sucessivamente). Estas estruturas são definidas na fase de análise dos requisitos funcionais após o levantamento do domínio do estudo de caso. Os acessos às estruturas das bases de dados são implementados em Linguagem de Definição de Dados - DDL e são representadas a partir de diagramas formais - formação dos *parsers* - e obedecendo às regras convencionadas por modelos, estes são copiados e usados na memória volátil para gerarem os locais dos registros de dados (campos/tuplas). As manipulações das informações das bases de dados são feitas por Linguagem de Manipulação de Dados - *Data Manipulation Language* (DML). Para exemplificar as fundamentações algébricas da Seção 2.6, apresenta-se uma modelagem conceitual implementada na ferramenta EERCASE[44] intitulada **Sistema de Gestão Acadêmica**, com a sua modelagem disponível no **Apêndice A**. Para gerar as relações SQL para consultas e *views* é necessário seguir os conceitos definidos em três partes:

[40] Detalhes e orientações de uso no endereço eletrônico https://sqlite.org/json1.html. Último acesso em 11 de novembro de 2017.

[41] *Threads* são seções de códigos de uma mesma aplicação (não são seções independentes) as quais compartilham os espaços da memória volátil, recursos de sistemas e os dados entre si. (Carissimi , 2010)

[42] JavaScript V8 é a *engine* Javascript de alto desempenho e de código aberto da Google. Encontra-se disponível em `https://developers.google.com/v8/`. Último acesso em 20 de março de 2017

[43] Referência do *Google Cloud Platform* node.js em `https://cloud.google.com/nodejs/`. Último acesso em 20 de março de 2017.

[44] EERCASE é uma ferramenta *Computer-Aided Software Engineering* (CASE), desenvolvida no Centro de Informática da *Universidade Federal de Pernambuco* (UFPE) com o objetivo principal de construir modelos conceituais para banco de dados. Informações sobre EERCASE encontra-se em `https://sites.google.com/a/cin.ufpe.br/eercase/apresentacao`. Último acesso em 20 de setembro de 2017.

a forma (*form)*, função (*function*) e consistência (*consistency*); definidos por Owens (2006).

- *Forma* - A forma se refere a estrutura da informação. Existe uma estrutura de dados para representar todas as informações de um conjunto amostral restrito de dados. Esta estrutura é chamada de relação (conhecida em SQL como uma tabela volátil), neste caso é a tabela que forma dinamicamente tuplas (conhecidas como linhas) e atributos (conhecidos como colunas). A forma do modelo relacional é uma representação lógica da informação. Esta lógica representa uma conformação, ou uma visualização abstrata de informação inalterável por qualquer coisa fora dela, uma vez que são cópias de dados em tabelas literais persistentes;

- *Função* - A parte funcional ou, componente manipulativo, trata-se das maneiras como as informações são operadas nos níveis lógico. Este conceito foi formalmente introduzido por (Codd, 1970). A parte funcional do modelo relacional define a álgebra relacional e o cálculo relacional. Estes dois tratados formais, ou puros, são expressos em linguagem de consultas por meio de relações algébricas, como descrito na Seção 2.6; e

- *Consistência* - Significa a coesão entre a forma e a função, uma vez que a forma representa uma relação - definida na forma de um modelo de dados ou uma tabela - no contexto semântico (estrutura logicamente formada) e a função, representa os parâmetros, fundamentados nos modelos da álgebra relacional e cálculo relacional, a consistência do modelo relacional.

As estruturas relacionais são estratificadas em formas normais - *Forma Normal de Boyce-Codd* (BCNF), didaticamente convencionadas por 1NF - *Normal Form* (NF), 2NF, 3NF, 4NF e 5NF, representando a ordem de complexidade da organização do projeto de banco de dados para a redução das redundâncias (evitar duplicidades de tuplas em entidades distintas), organização conceitual-relacional (coerência e melhoria da estrutura), aumento da integridade dos dados (garantir atributos independentes e associados à chave primária - relação de pertinência) e aumentar o desempenho (expurgar os atributos avulsos - os quais não possuem relações diretas com a semântica e chave primária - em tabelas, garantir que sejam exclusivos da lógica conceitual de cada tabela) (Silberschatz, 2012). As estruturas e as sintaxes relacionais são definidas por meio de comandos - tradicionalmente em *prompt* - DDL e DML para o SQLite. Conforme o Quadro 2.2, tem-se respectivamente as sintaxes para: criar uma tabela (*CREATE*), alterar, renomear ou adicionar campo (*ALTER*), definição de um sinônimo para o nome de uma tabela (*ATTACH*), exclusão de sinônimo do nome de uma tabela (*DETACH*) e, exclusão de uma tabela (*DROP*) . Para definir um tipo de dado, o desenvolvedor da estrutura da base de dados em SQLite terá à disposição cinco tipos concretos: Integer, Float, REAL, Text, BLOB e NULL. Para que um projetista de banco de dados saiba qual tipo definir e dimensioná-lo conforme o espaço para

cada variável de entidade, de cada campo de cada tabela de uma base de dados[45]. No momento da definição do nome e do tipo de dado em cada campo pode-se aplicar restrições (*constraints*) atribuindo as regras:

- *PRIMARY KEY* (Chave primária) - Identifica um campo com um índice único o qual referencia toda a tabela;
- *NOT NULL* (Não nulo) - Assegura que não haverá valor nulo durante um *INSERT*;
- *CHECK* - Verifica uma condição definida previamente em um campo, no momento de definição de dados - construção de uma tabela - se valida ou não uma inserção (*INSERT*), exemplo: idade Integer CHECK(idade >= 18);
- *UNIQUE* - Evita duplicidade em um campo uma vez que ao inserir esta condição todos os valores serão distintos; e
- *DEFAULT* - Insere em um campo um valor padrão - pré-definido - no momento de definição de dados, desta forma será atribuído a este campo caso nenhum valor seja definido.

Após o projetista definir as estruturas da base de dados em DDL, têm-se as possibilidades de manipulações conforme segue no Quadro 2.3, via sintaxe DML[46] para o SQLite. Os tipos de dados declarados na implementação de SQL são convertidos e definidos em cinco dos seis tipos SQLite (exceto NULL, por ser um tipo literal o qual representa a ausência de valor), com breves descrições abaixo, como descritos nas Fundamentações Algébricas na Seção 2.6 e no Quadro 2.4, com as conversões - simplificações - por tipos equivalentes. As definições nos tipos de dados na DDL devem ser seguras pois quando uma base de dados encontra-se em produção, um único erro na mudança de tipo ou na definição de espaço reservado pode comprometer a consistência.

- Integer: Define-se inteiro para um número absoluto com comprimento variável de 1, 2 , 3, 4, 6 ou 8 bytes. O comprimento do espaço reservado truncará conforme o tamanho do inteiro atribuído. Um *Integer* tem uma variação de alocação de −9.223.372.036.854.775.808 até +9.223.372.036.854.775.807 ou, aproximados 19 dígitos. Um inteiro é representado por qualquer série pura de dígitos, desde que não possua ponto decimal ou expoente;
- Float: Um número em ponto flutuante é armazenado em um vetor de memória de 8 bytes, compatível ao formato nativo de um registrador do processador. Números float abrangem o intervalo ±1.40129846432481707e -45 a ±3.40282346638528860e+38.

[45] A versão original do detalhamento dos tipos de dados com suporte no SQLite na versão 3 está disponível em `https://sqlite.org/datatype3.html`. Último acesso em 25 de março de 2017

[46] Referência em `https://www.safaribooksonline.com/library/view/using-sqlite/9781449394592/ch04s06.html`. Último acesso em 26 de março de 2017.

**Quadro 2.2:** Sintaxes DDL para o SQLite

| Comandos | Definições |
|---|---|
| CREATE | CREATE TABLE table_name(<br>nome_da_coluna tipo_de_dado restrições_coluna,<br>...,<br>) |
| ALTER | ALTER TABLE basededados_nome.tabela_nome RENAME TO novo$_n$*ome_tabela*;<br>ALTER TABLE basededados_nome.tabela_nome ADD COLUMN nova_definicao...; |
| ATTACH | ATTACH DATABASE'nome_da_base.db' AS 'sinônimo' |
| DETACH | DETACH DATABASE'sinônimo' |
| DROP | DROP TABLE tabela_nome; |

Fonte: Elaborado pelo Autor (2017)

Um ponto flutuante possui de 6 a 7 dígitos significativos, delimitando mais infinito e menos infinito e NaN (Não é um Número). As definições técnicas para números float encontram-se na referência 754 do *Institute of Electrical and Electronic Engineers* (IEEE)[47];

- *REAL*: Define uma variável de ponto flutuante que armazena um número com 8 bytes com até dupla precisão;
- Text: Uma estrutura de dados formada por uma sequência de caracteres com comprimento variável, usando a codificação de banco de dados *Unicode Transformation Format* (UTF)-8, UTF-16BE (*Big-Endian*) ou UTF-16LE (*Little-Endian*);
- BLOB: Armazena um tipo de dado integralmente binário, usa-se para guardar imagens ou arquivos de mídia ; e
- NULL: Representa a ausência de um ponteiro para um local de memória e por consequência a nenhum tipo de dado.

Para orientar outras pesquisas a partir da vertente de estudo mais aprofundada na codificação do SGBDR SQLite em linguagem C, desde a definição da funcionalidade de cada palavra reservada, acessos aos registros em endereços físicos, as formações léxicas e as manipulações de dados pelas construções sintáticas (ordens de leituras e precedências) e suas funções, seguem-se as referências na página oficial do projeto SQLite[48].

[47]Padrão da aritmética binária para pontos flutuantes `http://grouper.ieee.org/groups/754/` e representação na linguagem Swift em `https://developer.apple.com/reference/swift/float`. Último acesso em 26 de março de 2017.

[48]As implementações da estrutura fonte em linguagem C, para a interface, os objetos e as funções encontram-se respectivamente em https://www.sqlite.org/c3ref/constlist.html, `https://www.sqlite.org/c3ref/objlist.html` e `https://www.sqlite.org/c3ref/funclist.html`. Últimos acessos em 10 de novembro de 2017.

**Quadro 2.3:** Sintaxes DML para o SQLite

| Comandos | Definições |
|---|---|
| ALTER | ALTER TABLE nome_do_banco_de_dados.nome_da_tabela<br>RENAME TO novo_nome_da_tabela; |
| ALTER | ALTER TABLE nome_do_banco_de_dados.nome_da_tabela<br>ADD COLUMN define_coluna...; |
| VIEW | Visualizações em SQLite são somente para leitura.O desenvolvedor não pode executar declarações DELETE, INSERT ou UPDATE sobre uma visualização.<br>CREATE [TEMP] VIEW nome_da_view<br>AS<br>SELECT declaração_ de_consultas |
| ALTER TABLE | Os recursos para *RENAME TABLE* e *ADD COLUMN* e as variantes para os comandos *ALTER TABLE* têm suporte.<br>Os comandos para *DROP COLUMN*, *ALTER COLUMN* e *ADD CONSTRAINT* não tem suporte. |
| *ENABLE TRIGGER* | *ENABLE TRIGGER* { [ schema_nome . ] trigger_nome [ ,...n ] \| ALL }<br>ON { objeto_nome \| *DATABASE* \| *ALL SERVER* } [ ; ] |
| *DISABLE TRIGGER* | *DISABLE TRIGGER* { [ schema_nome . ] trigger_nome [ ,...n ] \| ALL }<br>ON { object_nome \| *DATABASE* \| *ALL SERVER* } [ ; ] |
| *TRUNCATE TABLE* | *TRUNCATE TABLE*,<br>[ { database_nome .[ schema_nome ] . \| schema_nome . } ], table_nome<br>[ *WITH* ( *PARTITIONS* ( {,l,},[ , ...n ] ) ) ] [ ; ] |

Fonte: Elaborado pelo Autor (2017)

## 2.6 Fundamentações Algébricas do SGBDR SQLite

As operações em SQL para consultas relacionais nas bases de dados são regidas por formalizações algébricas em linguagens universais. Como referência descritiva e por se tratar de uma linguagem amplamente utilizada, a álgebra relacional é apresentada no discurso paramétrico no atendimento pelo SQLite. Esta é descrita no decorrer de cada operação fundamental.

> *"A álgebra relacional é uma linguagem de consulta procedural descrita sob a forma de operações fundamentais: São as operações de seleção, projeção, união, diferença de conjuntos, produtos cartesianos e renomeação; além destas, tem-se as operações complementares de inserção de conjuntos, junção natural e atribuição.".* (Silberschatz, 2012)

Os tipos de dados no Quadro 2.4 são convencionados por nomes reservados e seus espaços de memória para alocações. Os detalhes das arquiteturas de endereçamentos, os ponteiros para referências e arredondamentos ou truncamentos, a exemplo dos vetores de memória para seções menos significativas, os quais não fazem parte do escopo deste estudo.

**Quadro 2.4:** Conversões de Tipos de Dados para Tipos Básicos SQLite

| Os tipos de dados definidos na criação de tabela ou expressão CAST | Tipo de dado |
|---|---|
| INT<br>INTEGER<br>TINYINT<br>SMALL INT<br>MEDIUM INT<br>BIG INT<br>UNSIGNED BIG INT<br>INT2<br>INT8 | INTEGER |
| CHARACTER(20)<br>VARCHAR(255)<br>VARYING CHARACTER(255)<br>NCHAR(55)<br>NATIVE CHARACTER(70)<br>NVARCHAR(100)<br>TEXT CLOB | TEXT |
| BLOB<br>(Tipo de dado não especificado) | BLOB |
| REAL<br>DOUBLE<br>DOUBLE PRECISIO<br>NFLOAT | REAL |
| NUMERIC<br>DECIMAL(10,5)<br>BOOLEAN<br>DATE<br>DATE TIME | NUMERIC |

Fonte: Página oficial do projeto SQLite[49]

### 2.6.1 Estruturas Fundamentais

As operações fundamentais são didaticamente definidas por operações unárias e binárias, sendo estas compostas por relações entre conjuntos do mesmo contexto interoperável (no âmbito do mesmo contexto lógico). Os exemplos que se seguem, após cada definição algébrica, são obtidos por informações arbitrárias aferidas da base de dados da **Modelagem de um Sistema de Gestão Acadêmica**, disponível no **Apêndice A**. As consultas sobre o modelo proposto foram realizadas na ferramenta *DB Browser para SQLite* (Referências sobre DB Browser para SQLite, 2017)

#### *Operações Unárias*

As operações unárias ocorrem em apenas uma relação (de $r$ em $r$) - onde R é uma expressão da álgebra relacional cujo resultado é uma relação. Descreve-se a seguir as formali-

zações algébricas e os exemplos em SQLite para as operações unárias de seleção, projeção e renomeação.

- **Seleção**: A operação de seleção retorna tuplas que atendem a um predicado explícito; a relação resultante - subconjunto de tuplas - da operação SELEÇÃO tem os mesmos atributos de R (está contido em)[50] e é especificada na cláusula WHERE de uma consulta. A letra grega $\sigma$ (sigma minúsculo) é convencionada para obter os argumentos a partir do predicado. O predicado é posto subscrito a $\sigma$. A relação de argumento do predicado é descrito em parênteses após o sigma e faz uso de operadores de comparação (=, $\neq$, <, >, $\leq$, $\geq$); *usa-se condicionais para selecionar ou descartar as tuplas que atendam a uma determinada condição* (Navathe, 2011). A partir da **Modelagem de um Sistema de Gestão Acadêmica**, disponível no **Apêndice A**, faz-se a representação para seleção conforme se segue: A forma algébrica de seleção para o conjunto dos docentes os quais lecionam a disciplina biologia é expressa por:

$$\sigma_{(\text{docente.codigo} = \text{leciona.codigoDocente} \wedge \text{leciona.codigoDisciplina} = \text{disciplina.codigo} \wedge \\ \textit{disciplina.nome}=\text{"}\textit{biologia}\text{"})}(\text{docente x leciona x disciplina})$$

A representação da operação unária de seleção é atendida pelo tradutor do SQLite pela sintaxe *SELECT DISTINCT doc.* FROM docente doc, leciona lec, disciplina dis WHERE doc.codigo = lec.codigo_docente AND lec.codigo_disciplina = dis.codigo AND dis.nome = "Biologia"ORDER BY doc.nome;*, esta seleção para a operação na base de dados - disponível no Apêndice A - é expressa na Figura 2.2 abaixo: .

**Figura 2.2:** Resultado da Consulta dos Docentes que lecionaram a Disciplina Biologia.

| codigo | nome | matricula | CPF | data_nascimento | data_posse | situacao | formacao |
|---|---|---|---|---|---|---|---|
| 2 | Adréia Seixas | 235468 | 856.211.357-55 | 10/11/1989 | 15/03/2008 | Ativo | Licenciado em Biologia |
| 1 | Matheus da Silva Carvalho | 123456 | 226.764.190-60 | 20/05/1988 | 10/01/2007 | Ativo | Licenciado em Matemática |
| 3 | Silvana Teixeira Villas Boas | 58742365 | 772.544.651-70 | 05/09/1988 | 11/04/2008 | Ativo | Licenciado em Biologia |

Fonte: Elaborado pelo Autor (2017)

Pode-se combinar sucessivos predicados com a inserção de conectivos da lógica proposicional e ($\wedge$), ou ($\vee$) e não ($\neg$); estes acrescentam condições para filtros diretos em consultas. Como exemplo, utilizando-se a operação sobre a base de dados já citada, fazendo-se a consulta para todos os docentes que lecionaram biologia no ano letivo de 2016.2:

$$\sigma_{(\text{docente.codigo} = \text{leciona.codigoDocente} \wedge \text{leciona.codigoDisciplina} = \text{disciplina.codigo} \wedge}$$

[50] O número de tuplas a partir da relação resultante é sempre menor ou igual ao número de tuplas em R, chamada de seletividade da condição (Navathe, 2011).

disciplina.codigoTurma = turma.codigo ∧ turma.anoLetivo = "2016.2"∧ disciplina.nome = "biologia")

(docente x leciona x disciplina x turma)

A consulta em SQLite é obtida pela sintaxe *SELECT DISTINCT doc.* FROM docente doc, leciona lec, disciplina dis, turma tur WHERE doc.codigo = lec.codigo_docente AND lec.codigo_disciplina = dis.codigo AND dis.codigo_turma = tur.codigo AND tur.anoLetivo = "2016.2"AND dis.nome = "Biologia";*, esta seleção para a operação na base de dados - descrita no Apêndice A - retorna a Figura 2.3 abaixo:

**Figura 2.3:** Consulta Docentes que lecionaram a Disciplina Biologia no Ano Letivo 2016.2.

| codigo | nome | matricula | CPF | data_nascimento | data_posse | situacao | formacao |
|---|---|---|---|---|---|---|---|
| 1 | Andréia Seixas | 123456 | 856.211.357-55 | 10/11/1989 | 15/03/2008 | Ativo | Licenciado em Biologia |

Fonte: Elaborado pelo Autor (2017)

- **Projeção**: É um conjunto de relações de argumentos definidos explicitamente sobre o conjunto total dos argumentos possíveis de seleção. Como forma de fundamentação, a operação de SELEÇÃO escolhe e descarta as linhas de uma tabela a partir dos condicionais, o resultado pode ser representado na forma de uma tabela; a operação de PROJEÇÃO selecionará certas colunas da tabela e descartará outras, será uma restrição sobre o subconjunto resultante, reduzindo a um conjunto de atributos desejável. *O resultado da operação de PROJEÇÃO pode ser visualizado como uma partição vertical da relação em duas relações: uma tem as colunas (atributos) necessários e contém o resultado da operação, e a outra contém as colunas descartadas* (Navathe, 2011). A sintaxe da álgebra de projeção é representada pela letra grega pí maiúscula (Π), e a forma geral da operação PROJEÇÃO é $\Pi_{\text{lista de atributos}}(R)$. Fazendo-se a simples consulta para retornar as disciplinas do Primeiro ano, turma A, do curso técnico em agropecuária (nível médio).

$$\Pi_{\text{nivel, serie, sigla\_turma, curso.nome, disciplina.nome}}$$

$$\sigma_{\text{(Curso.id = Possui\_Turma.id\_curso ∧ Possui\_Turma.id\_turma = Turma.id ∧}}$$

$$\text{Turma.id = Possui\_Disciplina.id\_turma ∧}$$

$$\text{Possui\_Disciplina.id\_disciplina = Disciplina.id)}$$

((Curso)x(Possui_Turma)x(Turma)x(Possui_Disciplina)x(Disciplina))

A Figura 2.4 abaixo expressa a saída da projeção em SQLite para a consulta *SELECT Curso.nivel, Curso.serie, Turma.sigla_turma, Curso.nome, Disciplina.nome FROM*

*Curso, Possui_Turma, Turma, Possui_Disciplina, Disciplina WHERE Curso.id = Possui_Turma.id_curso AND Possui_Turma.id_turma = Turma.id AND Turma.id = Possui_Disciplina.id_turma AND Possui_Disciplina.id_disciplina = Disciplina.id* sobre a base de dados do Apêndice A: .

**Figura 2.4:** Projeção para obter as Disciplinas do 1º Ano (A) / Nível Médio.

| | nivel | serie | sigla_turma | nome | nome |
|---|---|---|---|---|---|
| 1 | Médio | 1 | A | Técnico em Agropecuária | Matemática |
| 2 | Médio | 1 | A | Técnico em Agropecuária | Português |
| 3 | Médio | 1 | A | Técnico em Agropecuária | Biologia |
| 4 | Médio | 1 | A | Técnico em Agropecuária | Física |
| 5 | Médio | 1 | A | Técnico em Agropecuária | Bioquímica dos solos |

Fonte: Elaborado pelo Autor (2017)

- **Renomeação**: Aplica-se a possibilidade de renomeação para identificar cada expressão da álgebra relacional. A referência da renomeação é denotada pela letra grega minúscula rho ($\rho$); dado uma expressão da álgebra relacional E, a forma $\rho_x$(E), resulta na expressão de E sob o novo nome x. A singularidade em $\rho_x$(E) pode ser estendida para a expressão em E com argumentos passados até n, sendo denotado por:

$$\rho_x(A_1, A_2, ..., A_n)E$$

A expressão E é renomeada para x e os atributos resultantes renomeados para $_1$, $_2$, ... até $_n$. Para exemplificar a operação de renomeação faz-se uma *consulta singular para obter os nomes dos alunos que estejam lotados na disciplina biologia, na aula da data "20/08/2017"*. Procede-se em duas definições de conjuntos: (B) a seleção de todos os alunos matriculados na disciplina biologia e que tiveram faltas; (A) seleciona todos os alunos matriculados na disciplina biologia os quais estejam presentes no dia "20/08/2017". Procede-se fazendo a operação de subtração de conjuntos (A - B), obtendo os nomes dos alunos. A consulta do exemplo de renomeação é expressa na forma:

$$\Pi_{\text{nome}}(\sigma_{\text{(disciplina = "biologia"} \wedge \text{ presenca = "TRUE"} \wedge \text{ ocorre\_aula.data = "20/08/2017")}}$$

$$\text{(aluno)x(matricula)x(disciplina)x(ocorre\_aula))}$$

$$\Pi_{\text{nome}}(\sigma_{\text{(disciplina = "biologia"} \wedge \text{ ocorre\_aula.presenca = "FALSE")}}$$

$$\rho_d\text{(aluno)x(matricula)x(disciplina)x(ocorre\_aula))}$$

A renomeação da relação: $\rho_d$ (aluno) x (matricula) x (disciplina) x (ocorre aula) evitará ambiguidades nas situações auto referenciáveis uma vez que ao renomear, será aplicado um nome diferente como atributo identificador da relação.

### *Operações Binárias*

As operações binárias ocorrem sobre pares de relações, a partir de uma expressão de **composição das relações**, e os resultados advém das suas entradas (argumentos), na forma de expressão da álgebra relacional. Estas são definidas pela diferença de conjuntos, pelo produtos cartesianos e as operações complementares de inserção de conjuntos, atribuição, junção natural (*natural join*) e junção externa (*outer join*).

- **Interseção de Conjuntos**: Define-se como a interseção entre duas relações distintas representada pelo símbolo ($\cap$). Considere as relações r(R) e s(S) e sua interseção na forma R $\cap$ S, pode ser exemplificada em uma expressão algébrica para obter todos os alunos que cursaram a disciplina biologia nos anos de 2017.

$$\Pi_{\text{id, nome}}(\sigma_{\text{(disciplina = "biologia"} \wedge \text{ matricula = "2017")}}$$

$$\text{(aluno)x(matricula)x(disciplina))} \cap$$

$$\Pi_{\text{id, nome}}(\sigma_{\text{(disciplina = "biologia")}}$$

$$\text{(aluno)x(matricula)x(disciplina))}$$

A representação da interseção de conjuntos pode ser expressa pela relação de diferença de conjuntos, $r \cap s = r - (r - s)$, obtendo o mesmo resultado.

- **Operação de Junção Natural**: Decorre de uma relação binária na forma de um produto cartesiano, referenciado pelo mesmo valor ID, fazendo comparações entre os atributos e garantindo a remoção de atributos duplicados na saída, a junção natural (*natural join*) é expressa pelo símbolo ($\bowtie$). Para ilustrar uma relação de junção natural tem-se o exemplo: "Consulte todos os nomes e identificadores (id) dos professores e todos as disciplinas as quais lecionam".

$$\Pi_{\text{id, nome}}(docente) \bowtie (disciplina)$$

- **Junções Externas** *Outer Join* - Operação semelhante a Junção Natural exceto por manter as tuplas em um dos lados ou nos dois lados. São definidos como *Left Outer Join* - Junção à Esquerda ($\sqsupset\bowtie$), *Right Outer Join* - Junção à Direita ($\bowtie\sqsubset$) ou *Full Outer Join* - Junção Completa ($\sqsupset\bowtie\sqsubset$). Considere o conjunto de todas as disciplinas e de todos os docentes, incluindo aqueles que não estejam lecionando nenhuma disciplina (Figura 2.5), faz-se a consulta SQLite da relação *SELECT Docente.nome,*

*Disciplina.nome FROM Docente, Leciona LEFT JOIN Disciplina ON (Docente.id = Leciona.id_docente AND Disciplina.id = Leciona.id_disciplina).*

**Figura 2.5:** Operação *Outer Left Join*

| | nome | nome |
|---|---|---|
| 1 | Andréia Seixas | Biologia |
| 2 | Andréia Seixas | NULL |
| 3 | Matheus da Silva Carvalho | NULL |
| 4 | Matheus da Silva Carvalho | Matemática |
| 5 | Silvana Teixeira do Amparo | NULL |
| 6 | Silvana Teixeira do Amparo | NULL |

Fonte: Elaborado pelo Autor (2017).

- **Operação de Atribuição**: Este recurso utiliza o artifício da atribuição (semelhante a atribuição em linguagem de programação), inserindo uma seção de uma expressão relacional, em uma variável temporária. Esta operação usa o símbolo (←) e, a partir da atribuição, pode-se utilizar a variável temporária no decorrer da expressão. Como exemplo, tem-se a atribuição aplicada na expressão:

$$temp1 \leftarrow (alunoxdisciplina)$$

$$temp2 \leftarrow \sigma_{\text{situacao = "matriculado}}(temp1)$$

$$result = \Pi_{\text{nome, data\_matricula}}(temp2)$$

## 2.7 Interfaceamento: Fundamentos, Visualizações e Propostas de Interações

Os requisitos de interface são os fundamentos das interações humano / computador e estes subsidiam o sentido a implementação de um sistema. São definidos desde a fase de projeto na concepção e existirão perenes nos processos formais para as correções das interfaces e atualizações. Em termos gerais, sistemas são definidos por entradas e saídas, receptores e atuadores, sinérgicos em aferições e respostas, e para sobreviverem os sistemas devem ajustar-se (ou migrarem de interfaces) constantemente aos avanços dos recursos tecnológicos.

> *"Os dispositivos móveis como os smartphones e iPhones possuem dimensões reduzidas, diferenciam-se dos PCs e laptops essencialmente por causa do seu tamanho reduzido e portabilidade. Eles cabem no bolso, em uma bolsa ou em um case preso ao cinto, de modo a estar à mão o tempo todo."* (Yvonne, 2013)

As interfaces para interações com as aplicações podem ser ordenadas cronologicamente em terminais analógicos por linhas de comandos ou em ambientes gráficos com elementos visuais, enfatiza-se neste último as entradas para interações *touch screen* como meio de manipulação atualmente mais difundido. A empresa Apple retirou os tradicionais teclados físicos dos seus *smartphones*, estes são comuns nos aparelhos Palms TREON, Moto Q, BlackBarry e no Nokia E62, entre outros, vide Figura 2.6; estes teclados físicos foram declarados por Steve Jobs[51] como obsoletos, quando o comparou ao apresentar o primeiro iPhone. As inovações em interações por

**Figura 2.6:** Interfaces dos *Smartphones* Principais Concorrentes do Primeiro iPhone.


Fonte: `https://goo.gl/iQJ5gQ`

meio de teclados em *layouts* virtuais – a interface *multi touch*[52] – permitiram que os usuários usassem seus dedos sobre a tela – antes se usavam canetas – em acessos às figuras, aos mapas e aos textos, podendo ser rotacionados quanto aos posicionamentos, girados com os movimentos circulares, expandindo ou fechando em movimentos do tipo pinça (Figura 2.7).

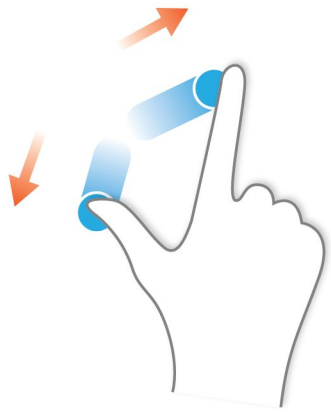
As melhores práticas em *design* de interfaces sustentam que as disposições dos elementos visuais, os modos de usar – usabilidade – e a sinergia das funcionalidades sejam fundamentos para o desenvolvimento de ambientes limpos. A *Apple* apresenta suas recomendações aos desenvolvedores de interfaces no documento *online* intitulado *Developer - UI Design Do's and Don'ts*[53], intuitivos, com recomendações das disposições de *layouts* e dos ícones, com recursos interativos e intuitivos – *uma premissa para o uso de ícones em vez de rótulos de texto é que eles são mais fáceis de serem operados e lembrados*. Como resultados, tem-se sucessivas versões de interfaces com cada vez menos percursos clicáveis e posicionamentos reduzidos para alcançarem seus resultados, são os recursos mínimos de interfaces (a menor distância para alcançar uma

[51] Disponível em `https://youtu.be/9ou608QQRq8?t=3m33s`, apresentação do primeiro iPhone em 09 de janeiro de 2007. Último acesso em 26 de março de 2017.

[52] Registro de patente em `https://goo.gl/gpLjki`, hoje conhecido como iTouch. Último acesso em 26 de agosto de 2017.

[53] Orientações disponíveis em `https://developer.apple.com/design/tips/`. Último acesso em 26 de agosto de 2017.

**Figura 2.7:** Movimento de Pinça em Interface *Multi Touch*.



Fonte: *Wikimedia Commons*
`https://commons.wikimedia.org/wiki/File:Gestures_Unpinch.png`

funcionalidade). Estas facilidades têm uma relação de primeira ordem com a aprendizagem intuitiva, como exemplos dos padrões de interfaces, disposições de *layouts*, menus, formulários e seus elementos, que *em termos gerais são as maneiras como uma pessoa interage com um produto ou aplicação*, resultam na aproximação da manipulação direta ou da semântica intuitiva baseada em padrões de interfaces – grau de usabilidades – que *permitem que usuários ocasionais lembrem como realizarem operações no decorrer do tempo ou usuários iniciantes aprendam rapidamente as funcionalidades básicas* (Yvonne, 2013).

## 2.8 Métricas de Interfaceamento e Usabilidade: Design de Interação, Visualização e Manipulação de Dados

As melhores práticas para construções de interfaces se baseiam no quanto os usuários se familiarizam com uma App (aplicativos para dispositivos móveis se chamam também de App) *como suas ações estão fazendo para seus objetivos serem alcançados* (Yvonne, 2013). Em cada aplicativo a ser posto em produção deve-se seguir as exigências funcionais dos meios de acessos pré-definidos pelos *stakeholders* ou análise do desejo do público alvo. Espera-se que os resultados, em cada ciclo de entregas versionadas, estejam em conformidade com as necessidades informadas no escopo dos requisitos. Tratando-se da seção de interface chama-se de Processos de *Design* de Interação; segundo Yvonne (2013) os processos podem ser subdivididos em quatro atividades básicas:

1. Identificação das necessidades dos usuários e estabelecer (delinear) os requisitos;

2. Desenvolver *designs* alternativos (projetar alternativas). Nesta etapa faz-se uso

de ferramentas de modelagens de interfaces para se ajustarem aos modelos mais semânticos possíveis;

3. Construir versões interativas dos *designs* (prototipar). Nesta etapa há um esquemático[54] do fluxo de funcionamento para adição e exclusão de colunas, conforme abordado por Cabral (2017) em fluxo de interações a partir da definição do diagrama de casos de usos; e

4. Avaliação do *design*, a qual refere-se à aceitação do sistema de informação.

As etapas supracitadas passam por verificações constantes quanto:

1. Ao foco no usuário no início de cada processo e avaliação do artefato;

2. Identificação, documentação e alcance de um acordo sobre os requisitos de qualidade (usabilidade, comunicabilidade, aprendizado e experiência do usuário); e

3. Os ciclos de interações tornam-se inevitáveis. No escopo da usabilidade, raramente se acertam a prototipação na primeira vez.

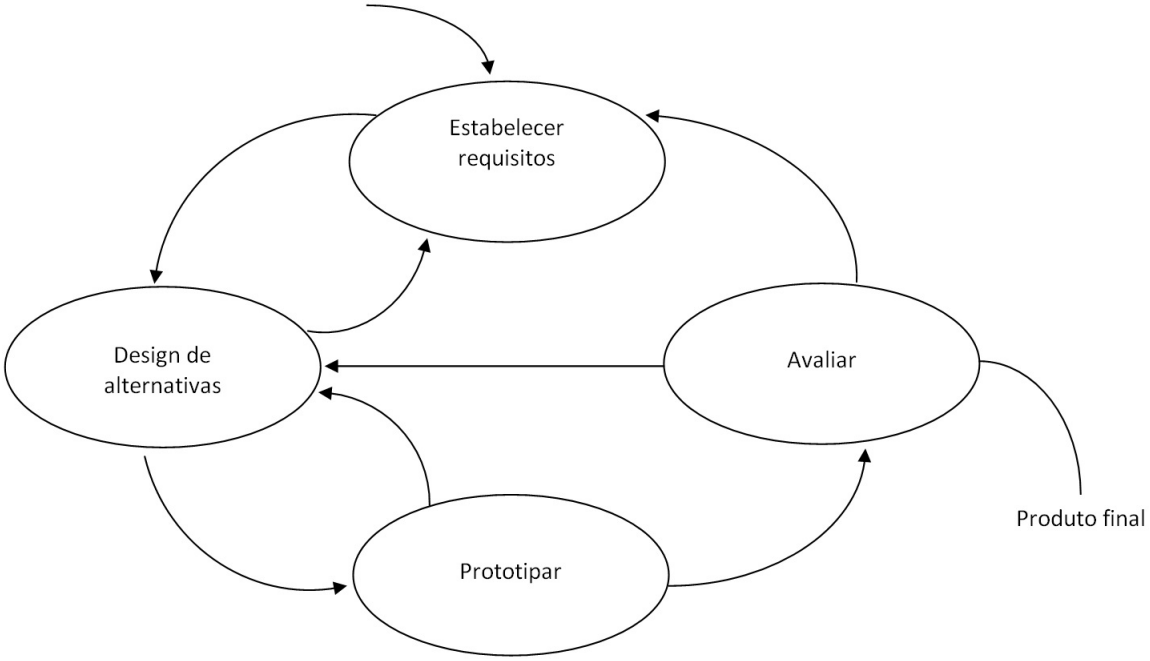
As atividades e as ações acima enumeradas constituem procedimentos para se obter os mais aproximados resultados quanto às interfaces. Estas devem seguir um roteiro ou Ciclo de Vida para o *design* de interações (atender às funcionalidades).A Seção 2.8.1 apresenta os ciclos para uma melhor produção dos resultados em conformidade com a modelagem.

### 2.8.1 Um Modelo de Ciclo de Vida para *Design* de Interação

As ações interativas de uma sistema devem atender aos Casos de Usos extraídos dos requisitos funcionais; as interações devem partir do estudo do usuário pois inicia-se e termina nele. As definições das atividades de desenvolvimento (Figura 2.8) se conformizam nas interações e resultados esperados; para cumprir as etapas dos caminhamentos das interfaces deve-se prever os ciclos de vida das ações. O usuário, a partir de um objetivo prévio, deve iniciar uma sequência de ações bem estruturada para alcançar o(s) resultado(s). Essa completude do interfaceamento é definida nos ciclos de vida das interações e seus objetivos devem ser sucintos e intuitivos, resultando no menor *stress* durante as interações. Esse contexto para a usabilidade antecipa os objetivos dos *designers* de interações mesmo a partir da construção das modelagens das interfaces.

> *A Engenharia de Software apresenta diversos modelos de ciclo de vida, como exemplo do modelo cascata (waterfall), modelo espiral e os modelos para aplicações de desenvolvimento rápido- Rapid Applications Development* (RAD) (Rogers, 2006) apud (Yvonne, 2013).

**Figura 2.8:** Um modelo simples de ciclo de vida de design de interação.




Fonte: (Yvonne, 2013).

O modelo de *design* de interação na Figura 2.8 descreve como as atividades estão inter-relacionadas, mesmo não tendo sido definido um critério para a saída de cada atividade, apesar de que os critérios de usabilidades devam ser especificados logo no início (fase de prototipação) e referidos em todas as atividades. Internamente, a prototipação está vinculada às prévias dos resultados antes de cada avaliação. O ciclo proposto para o *design* iterativo, as medidas empíricas de acertos - antes da seção de avaliação - baseadas nos resultados são as medidas de validade do sistema para gerar o produto final.

## 2.9 Considerações Finais do Capítulo

Este capítulo apresentou uma breve pesquisa quanto às tecnologias atendidas pelo SQLite para o trânsito de dados entre a aplicação e os serviços remotos. Analisou-se os trabalhos relacionados de Cabral (2017) e Rosa (2015), respectivamente quanto aos requisitos das funcionalidades do SQLite para a visualização e manipulação de dados e as referências para a estrutura de armazenamento. Também foram abordados o algoritmo de acesso à estrutura base e as vantagens do SQLite quanto à arquitetura. Apresentou-se a estrutura do SQLite, a origem, licença de uso, a Linguagem de Programação do código fonte do SGBDR SQLite. Elencou-se os suportes e restrições da DDL e DML. Discorreu-se sobre a Álgebra Relacional para dimensionar o suporte pelo SQLite. Fez-se um breve estudo em interfaceamento, formas de visualizações e interações nos *smartphones* iPhone. Por fim, abordou-se os ciclos e as avaliações de *design* de

[54] Utilizou-se a ferramenta Creately - disponível em `https://creately.com/` - para elaboração do diagrama. Último acesso em 03 de setembro de 2017.

interações para subsidiar o projeto de interfaceamento.

O próximo capítulo apresentará os principais *frameworks* disponíveis atualmente para o desenvolvimento de aplicativos mobile e as justificativas para a escolha da ferramenta XCODE para avaliar os seus recursos de interfaceamento. Serão aplicadas a Engenharia de Software Experimental e o uso da ferramenta *Unified Modeling Language* (UML) para descrever os requisitos funcionais para interfaceamento. Serão prototipadas as interfaces para as principais funcionalidades do SQLite estudadas neste capítulo. Serão apresentados os principais recursos de elaboração de interfaces do ambiente de desenvolvimento XCODE e a integração com o SGBDR SQLite.

# 3

# MATERIAIS E MÉTODOS PARA CONSTRUÇÃO DO PROTÓTIPO

Este capítulo discorre sobre as ferramentas de desenvolvimento para iPhone, as abordagens dos Processos de Engenharia Experimental, Diagramas de Casos de Uso, de Classes, de Estados e a prototipação de interfaceamento e sua usabilidade para tratar as funcionalidades e restrições nos subconjuntos das operações SQL: *DDL*, *DML* e *Data Query Language (DQL)*, aplicando-se as melhores práticas discutidas nas seções 2.7 e 2.8.

## 3.1 Recursos e Orientações

As soluções *mobile* nativas para iPhone - dizem-se nativas uma vez que co-existem atualmente as possibilidades de desenvolvimentos a partir de *frameworks*, como o exemplo do iOnic[1], do Xamarin[2] e o Adobe PhoneGap[3], entre outros que, em conjunto com o Apache Cordova[4] são plataformas para gerarem aplicativos híbridos e assim sendo, podem ser executados no iPhone, no Android e Windows *Mobile* (Sobre o Windows Mobile, 2017) são implementadas com as ferramentas: Objective-C[5] e Swift 3[6] (a atual versão é 4), ambas linguagens foram desenvolvidas pela empresa Apple. Objective-C é uma Linguagem de Programação de propósito geral, sendo a linguagem primária destinada a escrever software para Mac OS X[7], iOS[8], wat-

[1]Disponível em `https://ionicframework.com/`. Último acesso em 26 de agosto de 2017.

[2]Framework proprietário da empresa Microsoft; disponível em `https://www.xamarin.com/`. Último acesso em 26 de agosto de 2017.

[3]Disponível em `https://phonegap.com/`. Último acesso em 26 de agosto de 2017.

[4]Disponível em `https://cordova.apache.org/`. Último acesso em 26 de agosto de 2017.

[5]Disponível em `https://developer.apple.com/documentation/objectivec`. Documentações disponíveis em `https://goo.gl/ZjmDBX`. Último acesso em 26 de agosto de 2017.

[6]Swift 4: Mais informações no site oficial `https://developer.apple.com/swift/`. Último acesso em 22 de setembro de 2017.

[7]Mac OS X: Mais informações disponíveis no endereço oficial `https://developer.apple.com/macos/`. Último acesso em 22 de setembro de 2017.

[8]iOS: Mais informações disponíveis no endereço oficial `https://developer.apple.com/ios/`. Último acesso em 22 de setembro de 2017.

chOS[9] e tvOS[10]. É um subconjunto - com extensão do *GNU Compiler Collection* (GCC)[11] - da linguagem de programação C e fornece recursos de orientação a objetos e execução dinâmica, em tempo real. A linguagem Objective-C herda a sintaxe, tipos primitivos e as instruções dos fluxos de controle da linguagem C e adiciona a sintaxe própria para definir classes e métodos. Objective-C é a linguagem nativa do *Framework Cocoa Programming e do Cocoa Touch*, uma observação interessante é que o próprio *framework* foi escrito em Objective-C para servir de ambiente de desenvolvimento - *software development kit* (SDK) com API - para ele mesmo, este possui suporte a outras linguagens como Python e Ruby.[12] A ferramenta orientada para replicar - a partir da prototipação - os recursos de interfaceamento é a Switf3 (anotada quanto as mudanças para a versão 4). Justifica-se a escolha pois o Swift no XCODE apresenta suporte à metodologia *Model View Controller* (MVC) para separar e organizar o ambiente de visualização da lógica de programação, atendimento ao requisitos de *desenvolvimento Ágil incluindo técnicas de Desenvolvimento Guiado por Testes - Test Driven Development (TDD)* (Godfrey, 2017), além dos requisitos não-funcionais como a velocidade de compilação - até 2,6 vezes mais rápido que Objective-C, gerência automática de memória (Capacidade de Gerência Automática de Memória do Swift, 2017), verificações constantes das vulnerabilidades (Estatística de Vulnerabilidades do Apple Iphone OS, 2017) e mecanismos de seguranças disponíveis nos frameworks *(Sobre a Apple Security Framework, 2017)*. Swift3 e 4 encontram-se em versões estáveis e em produção pela Apple (Silveira, 2016).

> *A linguagem Swift foi anunciada oficialmente ao público no evento WWDC 2014 (Apple Worldwide Developers Conference)*[13] *apresentou uma sintaxe simples e moderna - com suporte à orientação à objetos e a construção semântica funcional; o diferencial foi deixar o desenvolvimento de aplicativos mais produtivo. A principal ferramenta Ágil para o Swift é o Cocoa Touch, trata-se de um framework que gera uma estrutura base no padrão MVC, nele a camada de View faz-uso da ferramenta Interface Builder para desenhar a tela a partir de um editor visual. A camada de controller são classes filhas de UIViewController, geradoras dos ciclos de vidas das telas, de tratar os eventos do usuário, controlar a navegação - rotas - do aplicativo e, por encerra na interação com a camada Model, que contém as classes e os objetos responsáveis pela lógica de negócios da aplicação.* (Lecheta, 2014)

[9] watchOS: Mais informações disponíveis no endereço oficial `https://developer.apple.com/watchos/`. Último acesso em 22 de setembro de 2017.

[10] tvOS: Mais informações disponíveis no endereço oficial `https://developer.apple.com/tvos/`. Último acesso em 22 de setembro de 2017.

[11] Disponível em `https://gcc.gnu.org/viewcvs/gcc/`. Documentação online em urlhttps://gcc.gnu.org/onlinedocs/gcc/. Último acesso em 20 de setembro de 2017.

[12] Todas as documentações para os desenvolvedores de softwares e aplicativos Apple encontram-se em `https://developer.apple.com/documentation`. As informações citadas encontram-se em *Guides and Sample Code* disponível em `https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/ObjectiveC.html`. Último acessos em 20 de setembro de 2017.

[13] WWDC 2014: Disponível em `https://developer.apple.com/videos/play/wwdc2014`. Último acesso em 02 de setembro de 2017.

Este estudo está disponibilizado ao público no ambiente Github iDBLite_Cin_UFPE[14], podendo ser importado via *Hyper Text Transfer Protocol Secure* (HTTPS), *Secure Shell* (SSH)[15] ou fazer *download* do arquivo compactado para um ambiente de estudo. Utilizou-se para verificações de recursos de interfaces a ferramenta XCODE[16] versão 8 da Apple, gratuita, porém específica para desenvolvimento restrito ao sistema operacional MacOSX[17] da empresa Apple. Utilizou-se um MacBook Air (Serial C02SQGAVH3QD) para correlacionar os recursos de interfaceamento[18]. A base de dados para teste foi posta no SGBDR SQLite a qual já executa nativo no próprio Sistema Operacional. Dispõe-se a sincronização Git[19] que ocorre a partir da aplicação GitHub ou recurso disponível no próprio XCODE, com acesso à conta do Github[20]. Orienta-se que se deve dispor ao desenvolvedor a base de dados de estudo (Chees, 2013) replicada em nuvem em ambiente *Software as a Service* (SaaS), como exemplo o Openshift *(Ambiente Red Hat Openshift, 2017)*, possibilitará as interações síncronas com outras plataformas em nuvem - tecnologia *Internet of Things (IoT)*, como exemplo da API ZettaJS (Sobre a API ZettaJS, 2017) sugerida para a sincronizações e produção em nuvem.

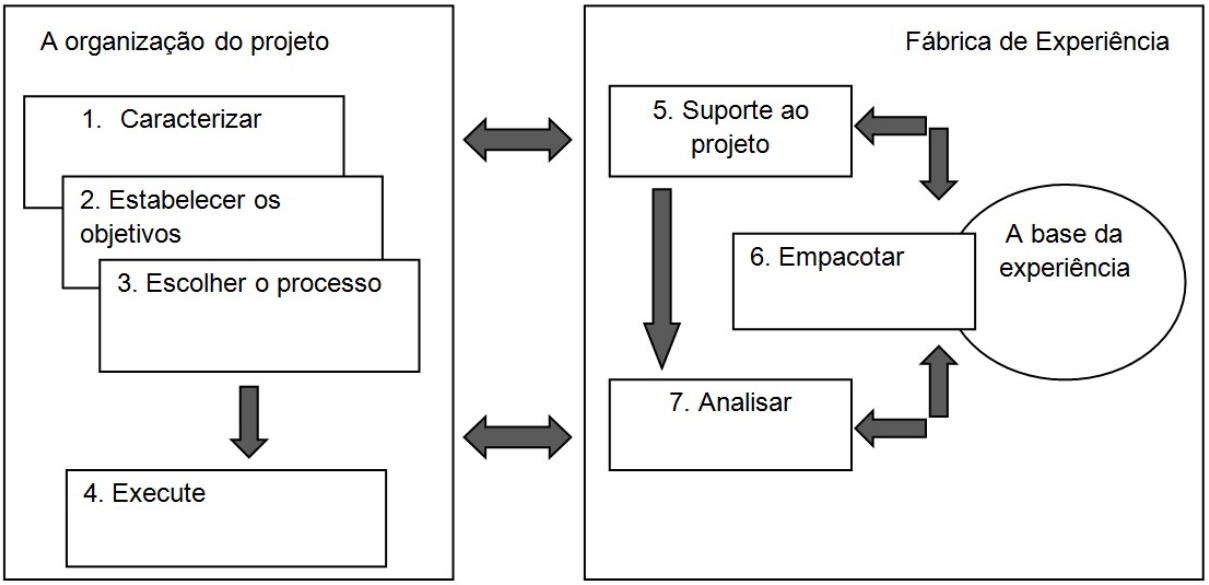
## 3.2 O Uso da Metodologia de Engenharia Experimental

A produção do interfaceamento utilizou-se da metodologia dos Processos de Engenharia Experimental como norteadora das etapas de *design* de interfaces, da Engenharia de Usabilidade e dos Ciclos de Vida do Design de Interação (Ciclos de *Designs* Interativos) - abordados na Seção 2.8.1, nos protótipos das modelagens e suas correlações funcionais com as interfaces. O escopo da prototipação nasce a partir da noção de ambiente e suas características no processo de negócio as quais, para esse caso, são os fatores que motivaram a sua consistência com os objetivos. A determinação do problema alvo - um estudo em interfaceamenteo para gerência SQLite - é delimitado nos fatores quantitativos de recursos (as funcionalidades) - existentes atualmente - a serem atendidos. O Github obedece a seção do critério de empacotamento do Paradigma da Melhoria da Qualidade - *Quality Improvement Paradigm* (QIP) para a disponibilização de códigos fontes, sistemas executáveis e documentações dos Processos de Desenvolvimento, neste caso em estudo, da Engenharia Experimental. Estas serão avaliadas e detalhadas em suas práticas, a determinação dos problemas funcionais, a proposição de melhoria futuras (avaliação empírica pelo público alvo), onde todas informações estão empacotadas para uso futuro. O QIP

[14]Disponível em `https://github.com/gilchristiano/iDBLite_Cin_UFPE`. Acesso permanente.
[15]Mais informações sobre SSH - Secure SHell em `https://www.ssh.com/ssh/`.
[16]Disponível em `https://developer.apple.com/xcode/`. Atualmente encontra-se na versão 9. Último acesso em 27 de agosto de 2017.
[17]Mais informações da versão atual do Sistema Operacional MacOSX Sierra disponível em `https://www.apple.com/lae/macos/sierra/`. Último acesso em 26 de agosto de 2017.
[18]Disponível em `https://gilchristiano.github.io/idblite/`. Acesso permanente.
[19]Git: Sistema livre e de código aberto para Controle de Versões. Mais informações disponíveis em `https://git-scm.com/`. Último acesso em 26 de agosto de 2017.
[20]Acesso Git em git@github.com:gilchristiano/iDBLite$_{Cin_UFPE.git}$

é ligado ao conceito da Fábrica de Experiência (Basili, 1994), o qual é definido como *conjunto das ferramentas para armazenamento, modificações e retiradas de informações do projeto empacotado* (Travassos, 2002). O fluxo das atividades da Fábrica de Experiência é apresentado na Figura 3.1 separando a organização do projeto em escopo abrangente e a fábrica como a fase da construção efetiva a partir da base da experiência.

**Figura 3.1:** Fábrica de Experiência.




Fonte: Wohlin (2012).

### 3.2.1 Aplicando o Processo de Experimentação

Os requisitos funcionais obtidos no resultado do experimento *Data Handling*, disposto na Tabela 2.1, de Cabral (2017) são tratados como parâmetros para a reprodução no experimento, refatorado nas modelagens dos protótipos de interfaces (etapa de prototipações descritas na Seção 3.2.3) e nas avaliações funcionais das interações com previsões dos resultados (óbvio uma vez que não se deve fugir dos requisitos definidos). Para se alcançar estas caracterizações dispôs-se de protótipos com *layouts* aproximados - refatorados - e definitivos, de acordo a versão atual das interfaces e suas funcionalidades, elaboradas na ferramenta Pencil[21]. Como resultados dos sucessivos ajustes nos *layouts* e os atendimentos esperados. São discriminados quatro métodos para a produção de experimentos na área de Engenharia de Software, o científico, de engenharia, o experimental e o analítico (Wohlin, 2012).

1. O Método Científico - Faz-se observações dos fenômenos, delimitando as hipóteses e seus modelos para determinar os motivos e variáveis. O método infere o

[21] Pencil é um software de código aberto voltado à prototipação de interface gráfica - *Graphical User Interface* (GUI), o código fonte do projeto encontra-se disponível em `https://github.com/evolus/pencil`. O website oficial para download da ferramenta está disponível em `https://pencil.evolus.vn/`, na versão 3.0.4, publicada em 27 de junho de 2017. Último acesso em 26 de agosto de 2017.

processo, o ambiente e o produto de software, afere os pontos representativos e gera a construção de modelos;

2. O Método de Engenharia - Faz-se observações sobre as soluções existentes, sugere as mais adequadas, implementa, mensura e analisa, repetindo até alcançar o objetivo ao qual é proposto;
3. O Método Experimental - Sugere o modelo, desenvolve o método qualitativo e/ou quantitativo, aplica a um experimento, medindo, analisando e avaliando o modelo, repetindo sucessivamente o processo; trata-se de uma abordagem orientada à melhoria revolucionária e
4. O Método Analítico - Desenvolve e/ou aplica a teoria formal, onde se deriva os resultados e, se possível, faz-se comparações com observações empíricas.

Adotou-se a Modelagem UML[22] e o Método Experimental, item 3, nos processos de experimentações, pois são elaborados protótipos dos modelos, usando o fator qualitativo quanto às facilidades de acessos, à usabilidade e às melhores práticas ergonômicas de interfaceamento. Abaixo tem-se a elicitação dos Requisitos Funcionais (RF), ao conjunto das funcionalidades básicas do SGBDR SQLite para a prototipação de interfaceamento:

- RF1 - Acessar o ambiente de usuário (autenticação);
- RF2 - Acessar a base de dados SQLite - no próprio dispositivo ou na nuvem - com restrições de acessos para manipulações, permitindo só o acesso à seção de consultas relacionais (usuário autenticado com limitações);
- RF3 - Todos os usuários terão acessos aos históricos das suas consultas relacionais e podem exportar os resultados (query) em .sqlite (.sql), Comma Separated Values (CSV) ou Portable Document Format (PDF);
- RF4 - Acessar a base de dados SQLite para a gerência total (usuário administrador), incluindo criar a base, criar tabelas e seus campos, alterar o nome e as estruturas das tabelas (DDL), incluindo as funcionalidades de consultas definidas acima em RF2;
- RF5 - Possibilidade de exportaçõa das mudanças da base de dados no *smartphone* para repositório na nuvem; e
- RF6 - Gerenciar credenciais de autenticações de usuários, com níveis de permissões, sendo pré-definidos para usuário comum e administrador.

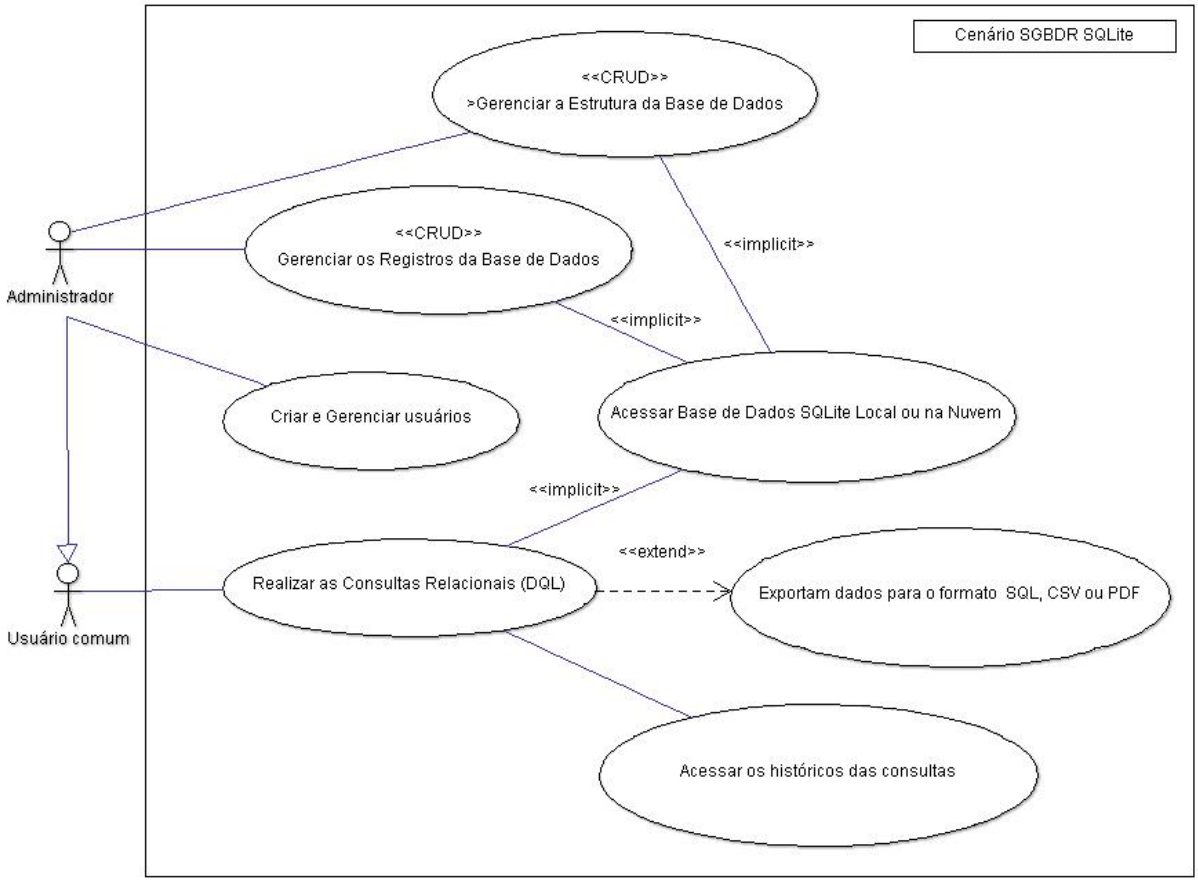
### 3.2.2 Modelagem UML para o Interfaceamento

A UML apresenta uma abordagem conceitual de sistema com notação própria. Ela define a criação de modelos abstratos de processos derivados das especificações dos requisitos

[22]UML: Endereços oficiais em `http://www.uml.org/`, `https://www.uml-diagrams.org/`, Ivar Jacobson `http://dblp.org/pers/hd/j/Jacobson:Ivar` e homologado em 1997 pela *Object Managment Group* (OMG) `http://www.omg.org/`. Últimos acessos em 12 de novembro de 2017.

funcionais - pós-análises - para um ambiente visual (Bezerra, 2014). A depender da ferramenta UML, é possível migrar diretamente os "moldes"para um modelo orientado a objeto por meio de um *framework* (SDK). Os diagramas propostos se dividem em representações que seguem, em sua estrutura, os diagramas estáticos: Diagrama de Casos de Uso e o Diagrama de Classes e o diagrama dinâmico de Estados. Para realizar os esboços dos diagramas UML fez-se uso da ferramenta ArgoUML[23]. O **Diagrama de Casos de Uso** da Figura 3.2 documenta o sistema no nível de usuário. Este é um artefato que surge diretamente das especificações dos requisitos e que define as ações do usuário, pois só pode ocorrer uma ação por vez no *smartphone* (limitação para acessos concorrentes). Em sua elaboração, *deve-se evitar que o diagrama tenha um conjunto*

**Figura 3.2:** Diagrama de Casos de Uso simplificado.




Fonte: Elaborada pelo Autor (2017).

*muito grande de elipses, pois nesse caso, fica inviável compreendê-lo*. Os estereótipos devem possuir processos completos, sobrecarregados em ações; como exemplo, *as operações CRUD não devem ser tratadas como um caso de uso individual, elas devem ser agrupadas em casos de uso do tipo "manter" ou "gerenciar"*. A ação de habilitar o acesso não deve constar no diagrama, *visto que ir ao sistema fazer login e em seguida desligar o computador não pode ser visto como*

[23] ArgoUML (versão 0.34) é uma ferramenta de código aberto sob licença (EPL - *Eclipse Public License* versão 1.0 - `http://www.eclipse.org/legal/epl-v10.html`) com suporte a modelagem para todos os padrões UML na versão 1.4 para diagramas. Encontra-se disponível em `http://argouml.tigris.org/`. Último acesso em 14 de novembro de 2017.

*um processo completo que produz resultado consistente* (Wazlawick, 2011). Observa-se no Diagrama de Casos de Uso a distinção entre usuários comuns e dos administradores do banco de dados. As ações pertinentes a cada usuário são:

1. Usuário comum

    - Após a seção de (*login / senha*) permissão de acesso (a autenticação é intrínseca ao diagrama), o usuário localiza uma base de dados pré-existente (uma por vez), em um diretório no dispositivo ou a acessa remotamente em um serviço SaaS, em nuvem - síncrono;
    - Realizar consultas relacionais (DQL), a partir das visualizações da estrutura (*schema*) da base, nas tabelas e seus campos - e lembrar-se que há restrições de seção para inserir, alterar e excluir as informações; e
    - Exportar os dados das consultas no formato SQLite (.sql ou .sqlite), ou *Comma Separated Values* (CSV), ou *Portable Document Format* (PDF).

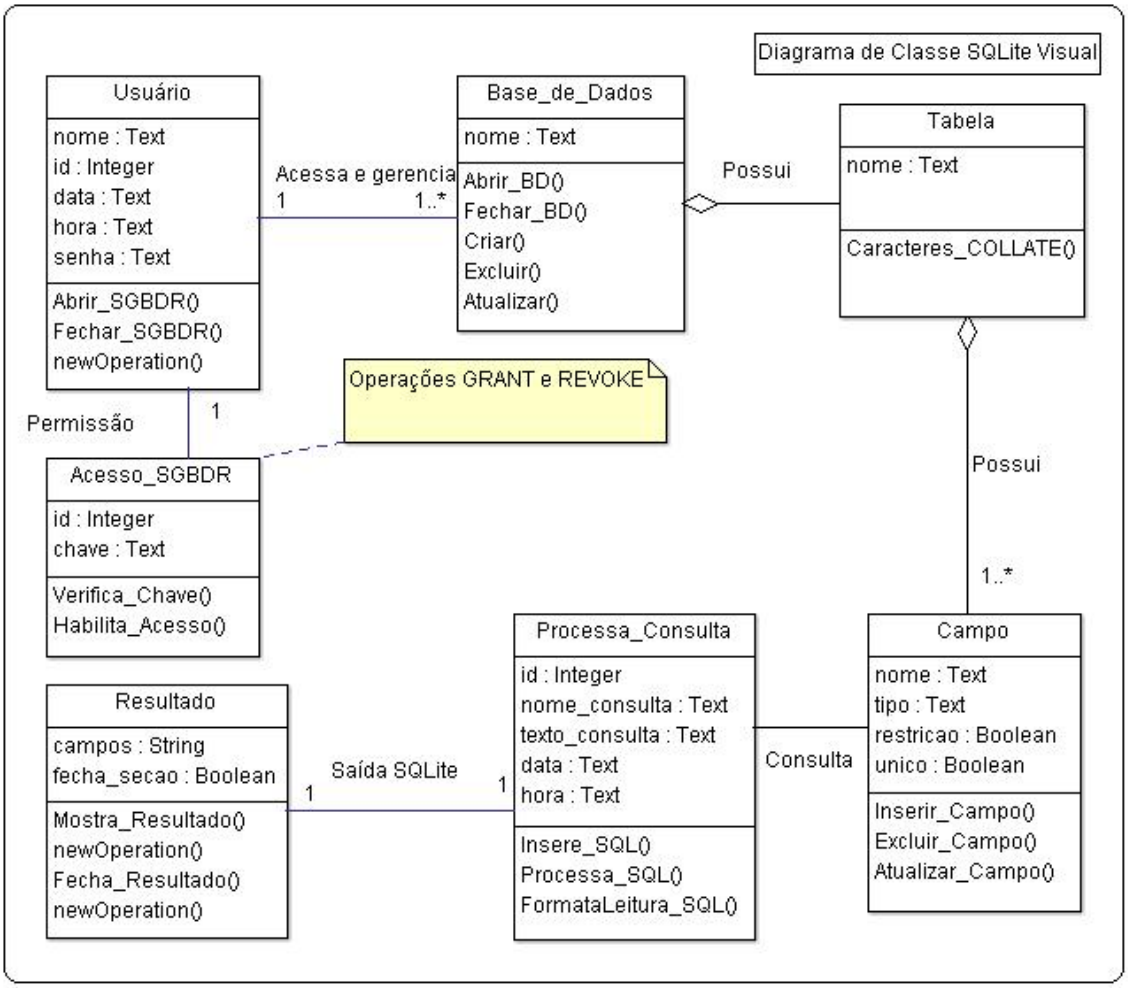
2. Usuário Administrador (ator principal)

    - Possui os mesmos privilégios do usuário comum para visualizações e exportação de dados;
    - Gerencia usuários, criando, atualizando ou excluindo o nome e a senha, tem-se dois tipos pré-definidos: usuário comum ou administrador (*GRANT e REVOKE*);
    - Gerenciar autenticação para acesso de bases de dados;
    - Liberdade para criar base de dados, atribuindo um nome; e
    - A partir de uma base de dados gerada ou pré-existente, pode-se criar, alterar ou excluir campos ou alterar seus atributos (tipo de variável, restrições de condição - ser ou não nula, e sua integridade referencial - índice).

No **Diagrama de Classes** tem-se definidos as variáveis privadas e os métodos para as principais funcionalidades. Definiu-se o uso do tipo *Text*, apesar do tipo String ter suporte no SQLite (fundamentado no Quadro 2.4). Esse tipo de dado tem seu tamanho variável e se ajusta à sequência de caracteres, solucionando o problema para aumento dos espaços de memória pré-reservados e não utilizados. Os tipos data e hora em SQLite são definidos como tipo *Text*[24]. O ambiente funcional de interações usuário x gerenciador para nortear o desenvolvimento é definido em seções de acessos, gerências de usuários (privilégios) e configurações e liberdades para manipulações DDL e DML (Figura 3.2). Os diagramas são recursos da Engenharia de

[24]Utilizou-se o tipo genérico *Text* para orientar a prototipação - pode-se usar também os tipos REAL e INTEGER. O padrão convencionado para data e hora (calendário) em ISO:8601 escrito no formato *Strings* para data e horário - "YYYY-MM-DD HH:MM:SS.SSS"/ Ano-Mês-Dia Hora:Minuto:Segundo.Milissegundos. Referências em `https://sqlite.org/datatype3.html`. Último acesso em 19 de novembro de 2017.

**Figura 3.3:** Diagrama de Classe SGBDR SQLite.




Fonte: Elaborado pelo Autor (2017).

Software para descreverem, organizarem a estrutura, definindo as classes e seus métodos (ações) para posterior interações e documentações. A partir do Diagrama de Casos de Uso (Figura 3.2) tem-se o Diagrama de Classe (Figura 3.3), sendo este o parâmetro para os demais diagramas. Para dimensionar cada funcionalidade, para cada instante de manipulação do usuário, fez-se o Diagrama de Estados protótipo de interfaceamento, no âmbito dos requisitos expostos no diagrama da Figura 3.2, de forma clara. O Diagrama de Classes apresenta o acesso de um usuário por vez. As operações de criação da base de dados, tabelas e campos são respectivamente definidas nas classes Banco_de_Dados, Tabelas e Campos, essa ordem tem inter-dependências obrigatórias. Por obviedade, a classe Processa_Consulta só realizará operações sobre a base de dados após certificar (recomenda-se verificar em segundo plano) a existência de sua estrutura e dos dados populados.

O **Diagrama de Estados** (Figura 3.4) apresenta cada função intrínseca na estrutura: DDL (descreve a estrutura), DML (manipula os dados) e *Data Control Language* (DCL) (gerencia os

usuários) e no caminhamento das ações de usuário entre os estados (ações candidatas a serem métodos *Create, Read Update e Delete* (CRUD)). A partir do estado inicial (usuário) têm-se os acessos com níveis de permissões, limitadas por *default* em usuário comum (destacado na cor verde) ou usuário administrador (destacado na cor vermelho claro). O administrador pode criar ou editar os novos usuários. O Diagrama constitui-se em 9 (nove) estados logicamente associados:

- Modo inicial (*initial mode*) - Local de início de acesso ao sistema;
- Verifica acesso - Realiza a abertura de seção de usuário, medida de segurança (DCL / Cria e edita privilágios);
- Base de Dados - Define a base de dados, o *script* do *schema* DDL;
- Base na Nuvem - Base de Dados disponível ou replicada na Nuvem (ambiente PaaS com garantia de alta disponibilidade). Possibilidade de *upload* do *schema* e dos dados;
- Tabela - Estado de ações sobre uma Tabela: criar e editar (ações CRUD);
- Campo - Estado subsequente a criação ou leitura de uma Tabela. Define-se cada campo e suas tuplas;
- Diretório local - Espaço físico de armazenamento persistente para o SGBDR SQLite e as bases de dados;
- Base de Dados (visualização) - Obtém-se as visualizações a partir das consultas SQL; e
- Relatório - Estado de apresentação dos dados consultados na interface ou em formato de documento portável (PDF).

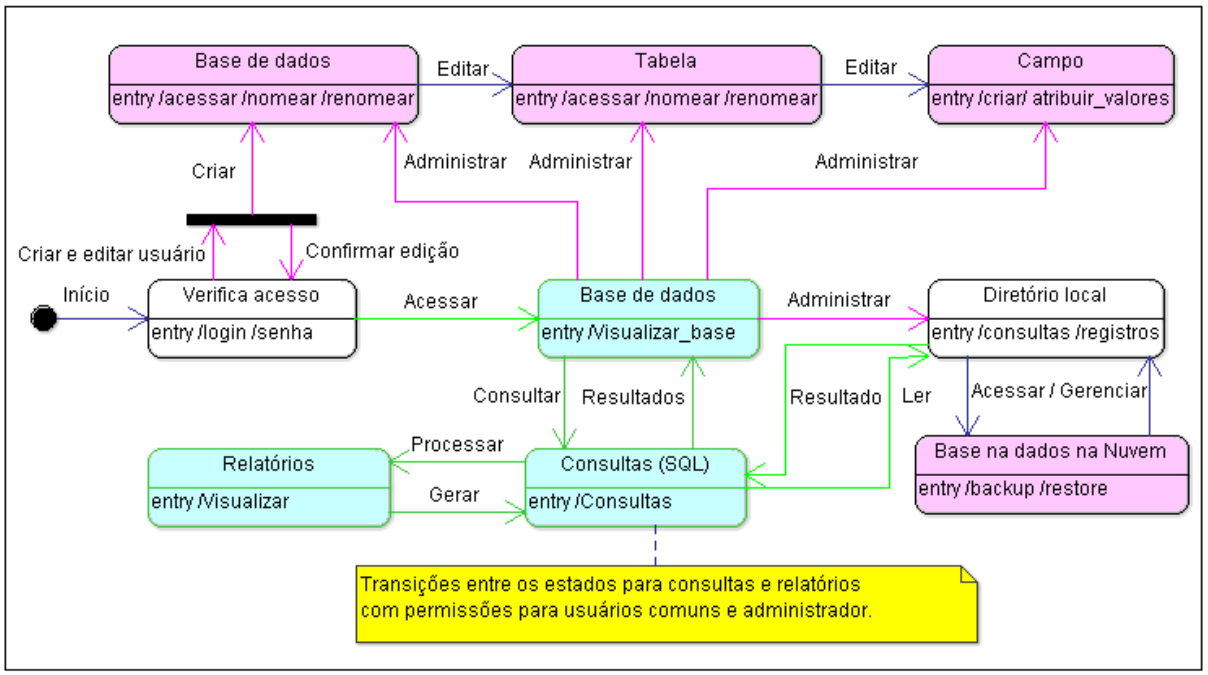
A proposta para interfaceamento SGBDR SQLite tem acesso não concorrente para ações de usuário / dispositivo, motivo este que dispensa representar por Diagrama de Atividades.

### 3.2.3 A Prototipação do Interfaceamento

As bases das experimentações da Fábrica de Experiência simplificam gradualmente as prototipações a uma sequência mínima de interfaces que, no decorrer das reproduções nas usabilidades, alcancem os resultados. Os objetivos para estruturar e manipular as inúmeras bases de dados foram didaticamente organizados a partir das operações fundamentais de definições e manipulações dentro dos limites - subconjunto - propostos na prototipação SQLite. As recomendações para interfaces da empresa *Apple* em *Developer - UI Design Do's and Don'ts*[25]

[25]Guia para Interface de Interação Humana em `https://developer.apple.com/design/tips/`. Último acesso em 16 de novembro de 2017.

**Figura 3.4:** Diagrama de Estado SGBDR SQLite.




Fonte: Elaborado pelo Autor (2017).

para os *layouts* foram aplicadas nas telas das prototipações (botões, textos, espaçamentos, entre outros), elaboradas por meio da ferramenta Pencil[26]. Os protótipos das interfaces são apresentados a seguir, de acordo com a sequência citada para as definições (DCL), o controle de acessos (DDL) e as manipulações (DML).

## *Prototipação DDL: Linguagem de Definição de Dados*

A definição da estrutura da base de dados em SQLite ocorre a partir das operações de criação do banco (nomear), inserções de campos, alterações e exclusões, renomeações da base e dos campos. O protótipo do interfaceamento apresenta as modelagens de interações equivalentes aos comandos DDL. Este apresenta uma interface simplificada de inicialização (Figura 3.5), contendo as operações básicas para criações e de acessos às bases de dados. O desenvolvedor de uma estrutura de base de dados iniciará selecionando o botão *Criar Base de Dados* e então prossegue atribuindo um nome coerente ao destino de sua estrutura (Figura 3.6). Para mostrar os exemplos das prototipações, fez-se simples seções de uma agenda de atendimento comercial (funcionários - atendimentos - clientes), onde se inicia uma base de dados nomeada Agenda. Caso o nome não exista conclui-se efetivamente a criação, caso contrário, há um aviso de "esta base de dados já existe". Os informes de confirmações, alertas e decisões de acessos são postos em janelas *pop-up*. As sintaxes das operações DDL para definições de dados são: criação

[26]Ferramenta de prototipação de interface citada na nota de rodapé número 18. Mais informações em `https://pencil.evolus.vn/`. Último acesso em 09 de setembro de 2017.

**Figura 3.5:** Interface Inicial com Acesso Simplificado para Criar a Base de Dados.


Fonte: Elaborada pelo Autor (2017).

(*CREATE*[27]), definição e exclusão de sinônimos (*ATTACH e DETACH*)[28] - opções disponíveis na aba configurações, alteração (ALTER[29]), exclusões de registros (*TRUNCATE / DELETE*[30]),

[27]A sintaxe suportada pelo SQLite para criação de tabela encontra-se em `https://sqlite.org/lang_createtable.html`. Último acesso em 12 de setembro de 2017.

[28]O comando *ATTACH* cria um sinônimo para o nome da base de dados (*ATTACH DATABASE* 'nome_da_base.db' as 'sinônimo';) e, *DETACH* exclui o identificador sinônimo (*DETACH DATABASE* 'sinônimo';). Informações disponíveis em `https://www.tutorialspoint.com/sqlite/sqlite_attach_database.htm`. Último acesso em 14 de setembro de 2017.

[29]O fluxo da sintaxe suportada pelo SQLite, para alteração de tabela para inclusão de uma nova coluna, encontra-se em `https://sqlite.org/lang_altertable.html`. Último acesso em 12 de setembro de 2017.

[30]A função de exclusão de um campo se faz pelo comando *DELETE*. Detalhes disponíveis em `https://www.sqlite.org/lang_delete.html`. Último acesso em 12 de setembro de 2017.

adicionar comentários (*COMMENT*[31]) e renomeação (*RENAME*[32]). A partir do segundo botão

**Figura 3.6:** Interface de Confirmação ou Não para Criação de Base de Dados.


Fonte: Elaborado pelo Autor (2017).

de acesso, em "Localizar Base de Dados", no protótipo da *interface* (Figura 3.5), após o acesso ao ambiente das bases de dados locais (arquivos .sql ou .sqlite), tem-se a seleção da base Agenda e conseguinte as ações possíveis para gerir: tabelas ou *views*. No protótipo de interação, faz-se o acesso às listagens das tabelas da base de dados Agenda (atendimento, clientes e funcionários). Segue as interações nesta *interface*: voltar, configurações, editar (contendo um menu *pop-up* ou lista suspensa para inserir ou excluir tabela), e propõe-se um editor flutuante para editar em

[31] O fluxo da sintaxe para inserir comentário, seja em uma linha via comando – ou, /* comentários em múltiplas linhas */, no estilo da linguagem C, conforme segue em 1.1.3 e apresentado em `https://www-s.acm.illinois.edu/webmonkeys/book/c_guide/1.1.html`, encontra-se em `https://sqlite.org/lang_comment.html`. Último acesso em 12 de setembro de 2017.

[32] A sintaxe SQLite para renomear encontra-se também na nota número 21, em *ALTER*, no endereço `https://sqlite.org/lang_altertable.html`. Último acesso em 12 de setembro de 2017

comandos SQL. No protótipo, para abrir uma tabela - cliente por exemplo - deve-se clicar no nome diretamente. Para excluir uma tabela será necessário selecioná-la e em seguida clicar em excluir a partir do menu Editar. Só há possibilidade de uma exclusão por vez, a qual estará vinculada a um *pop-up* de confirmação da ação (Figura 3.7). No decorrer das funcionalidades dos protótipos - interações - desta subseção DDL e os passos ordenados em suas interfaces, clicando diretamente em Clientes, obtém-se a tela com os campos (Figura 3.9) abaixo, uma estrutura já pré-definida. A partir desta interface o usuário desenvolvedor poderá: voltar à interface (tela)

**Figura 3.7:** Interfaces com a Dinâmica de Interação para a Exclusão de Tabela.


Fonte: Elaborada pelo Autor (2017).

anterior, inserir um novo campo, editar a tabela cliente para ajustar-se a um novo tipo de dado, restrições e valor *default*, ou excluir um campo - esta última ação necessitará pré-selecionar um campo desejado e depois clicar em excluir. Para inserir um novo campo deve-se clicar na aba "Novo campo"e inserir o nome do identificador do novo campo. Para o exemplo de interface a Figura 3.8 foi intencionalmente apresentada com a criação do campo 'codigo' (sem

**Figura 3.8:** Interface para Definição de um Novo Campo, seu Nome e Propriedade.


Fonte: Elaborada pelo Autor (2017).

acento). O protótipo desta interface "informa"que, ao selecionar *PRIMARY KEY*, as opções *UNIQUE* e *FOREIGN KEY* serão desabilitadas, uma vez que toda chave primária é única e, torná-la simultânea à chave estrangeira pode levar a inconsistências em relações algébricas (exceto auto-relacionamento), por isto, optou-se por recomendar neste protótipo.

As demais funcionalidades DDL serão: *Constraint name*, AUTO_INCREMENT e *COLLATE*, respectivamente usam-se para restrições da chave estrangeira, habilitar o auto incremento do código (índice) e definição do tipo de caracteres aceitos - SQLite possui as instruções sqlite3_create_collation para os suportes SQLITE_UTF8, SQLITE_UTF16, SQLITE_UTF16LE, SQLITE_UTF16BE e SQLITE_UTF16_ALIGNED[33]. Para o protótipo iDBLite, indica-se a

[33]SQLite C Interface para definição de *COLLATE* encontra-se em `https://sqlite.org/c3ref/create_collation.html`. Último acesso em 12 de setembro de 2017.

**Figura 3.9:** Interface para Tabela Cliente com Exemplo da Listagem dos Campos.


Fonte: Elaborada pelo Autor (2017).

instrução UTF8_GENERAL_CI como *COLLATE* padrão. No âmbito das funcionalidades DDL, foram apresentados protótipos de *layouts* com as condições para visualização, inserção e exclusão para tabela e campo. Conclui-se nesta subseção do interfaceamento DDL a apresentação das funcionalidades para editar os campos de uma tabela (Figura 3.10), especificamente a tabela Cliente. Em cada linha tem-se o nome (o nome é um *link* para um menu suspenso com: nome editável, se é obrigado a popular - nulo ou não - caracteres aceitos em COLLATE), ao lado segue o tipo de dado, definição para valor *default*, chave primária ou chave estrangeira, se auto incrementável (AI). Não é recomendado que se modifique arbitrariamente os nomes dos identificadores dos campos, da condição de nulidade e dos tipos de caracteres aceitos. Estas ações causarão erros em sistemas que estejam em produção, mais especificamente ao nome quando as operações DML

**Figura 3.10:** Interface com as Funcionalidades para Edição dos Campos de uma Tabela.


Fonte: Elaborada pelo Autor (2017).

ocorrerem com as chamadas explícitas por nomes e não por posicionamento de linha na tabela. A interface de edição (Figura 3.10) possui um formulário com todos os campos e as opções de escolhas para os atributos: (NN) Não Nulo, (AI) Auto Incremento, (UNQ) Único e *Default* para padrão (não obrigatório). A ordem disposta na interface de edição (Figura 3.10) em cada tabela será fidedigna à ordem dos campos durante a criação. Em termos gerais, as interações nas interfaces para as operações DDL devem cumprir os comandos para: criação (*CREATE*) e a definição e limpeza do alias (*ATTACH / DETACH*)[34], a atualização (*UPDATE*) e a exclusão (*DROP*) em toda estrutura ou em parte. As ações da prototipação DDL foram elencadas no Quadro 2.2 da Seção 2.5, e cada definição de sua operação sobre a estrutura da uma base de dados é disposta ao lado da referência da(s) respectiva(s) proposta(s) de interfaceamento no

[34]Os comandos DDLs *ATTACH* e *DETACH* foram citados na tabela 2.4 e na nota de rodapé número 21.

Quadro 3.1:

**Quadro 3.1:** Funcionalidades DDL e Interfaces Correspondentes.

| Funcionalidade DDL | Interfaceamento proposto |
|---|---|
| CRIAR BASE DE DADOS<br>($sqlite3 base_de_dados.db)<br>cria ou informa base já existente. | Referências na Figura 3.5 e Figura 3.6. |
| ATTACH (sinônimo) e<br>DETACH (retira o sinônimo) | Referência na Figura 3.7, na seção configurações. |
| REMOVE BASE DE DADOS | Referência na Figura 3.7. |
| INSERIR E EXCLUIR TABELA | Referências na Figura 3.8. |
| LISTAGEM DA ESTRUTURA DE CAMPOS | Referência na Figura 3.9. |
| ALTERAR NOME E TIPOS DE DADOS | Referência na Figura 3.10. |

Fonte: Elaborado pelo Autor (2017).

### *Prototipação DML: Linguagem de Manipulação de Dados*

Os protótipos de interfaceamento para as manipulações das bases de dados, em atendimento às relações algébricas DML (unárias e binárias), apresentadas na Seção 2.6.1 e nas fundamentações da Seção 2.6, seguiram as referências das melhores práticas para elaborações de *layouts* da Apple[35] atendidas pelo construtor de interface XCODE[36] - ícones, textos, botões e formulários espaçados, em conjunto com os requisitos funcionais do escopo das próprias relações - para esta versão atual - item tratado pela aplicação *Data Handling* vide Tabela 2.1 - avaliação final obtida pelo *Data Handling*, enumerados de 1 a 6, respectivamente: 1 (tema inicial), 2 (tela com a listagem), 3 (tela de manipulação da base de dados), 4 (tela ou caixa em destaque), 5 (tela para comandos de inserções ou consultas de dados) e 6 (tela dos resultados das consultas). Observa-se que há inúmeras possibilidades de saídas relacionais como expresso nas projeções algébricas - representado pela letra grega pí ($\Pi$) citado na Seção 2.6.1.1, para consultas unárias e arranjos algébricos binários. Em uma lógica de primeira ordem as prototipações das *interfaces* de saídas *frontends* foram construídas com *layouts* dinâmicos (adaptativos), em telas únicas, com paginações ou em *grides* (organizados em grade). A Figura 3.11 e a Figura 3.14 mostram os resultados dos protótipos de interfaces para uma consulta em que há apenas saídas de tuplas para um só resultado, obviamente só ocorreu esse registro na data informada. Para acessar o ambiente de Consulta SQL seguem os procedimentos: clica-se em SQL, como mostrado na Figura 3.11, no último botão à direita, na barra superior, o qual dará acesso ao terminal para executar as

[35] *iOS Human Interface Guidelines*, vide referência número 14, são as propostas de *designs* de interfaces, disponível no endereço `https://developer.apple.com/ios/human-interface-guidelines/overview/design-principles/`. Último acesso em 16 de novembro de 2017.

[36] Disponível em `https://developer.apple.com/xcode/interface-builder/`. Último acesso em 19 de novembro de 2017.

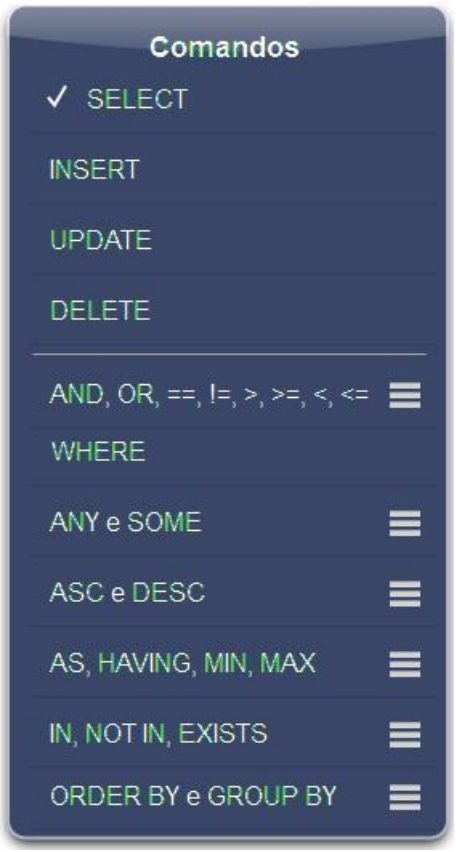
**Figura 3.11:** Interface de Acesso SQL e Inserção do **Select**.


Fonte: Elaborada pelo Autor (2017).

consultas; seleciona no terminal, na próxima interface, qual a operação no *list view* - encontra-se destacado o **SELECT** no protótipo. Para o protótipo contendo a listagem dos acessos rápidos (Figura 3.12) com os comandos na forma de menu flutuante - protótipo - são disponibilizadas as opções: *SELECT*, *INSERT*, *UPDATE* e *DELETE*, logo após, tem-se um separador de lista com as opções dos operadores condicionais *WHERE, ANY e SOME, ASC e DESC, AS, HAVING, MIN, MAX*, entre outros, todos organizados em opções de sub-listas.

Os operadores são inseridos dinamicamente no texto no local onde estiver o cursor de digitação. Por exemplo, em operações unárias (Seção 2.6.1.1), os operadores de comparações - podem ser inseridos na seleção como condicional do predicado $\sigma$. Para proceder com consultas aninhadas - subconsultas - clica-se no ícone ajuda, um tutorial orientará o desenvolvedor para inserir as opções em um menu editor de linha de comandos. Não se esgotaram as opções neste protótipo:

**Figura 3.12:** Comandos Essenciais DML para Acessos Rápidos.




Fonte: Elaborada pelo Autor (2017).

os comandos não disponíveis podem ser inseridos a partir do teclado flutuante. Para o protótipo atender à seleção binária, com múltiplas saídas no *ranger* ou faixa dos registros das consultas, por motivo das pequenas dimensões das telas dos principais modelos de iPhones, sendo de 4.7 e 5.5 polegadas[37], torna-se necessário apresentar as saídas em lista ou em acgrid. Diante desta necessidade recomenda-se criar protótipos para *layouts* dinâmicos. O usuário pode escolher entre visualizar em lista ou em *grid*. No protótipo, o usuário *DB Design* pode retornar a qualquer tempo à interface inicial a partir do acesso no rodapé em "Acessar outra base de dados?". No início, (Figura 3.5) tem-se o acesso - o usuário administrador - uma estrutura de base de dados SQL na nuvem, a partir do botão "Importar e Exportar Base de Dados". Recomenda-se utilizar os protocolos de acessos HTTPS ou *Secure File Transfer Protocol* (SFTP) na etapa de codificação. A ação indicada terá uma estrada para copiar a base de dados para o dispositivo via endereço *Internet Protocol* (IP) ou atualizar. O usuário administrador pode exportar fazendo o *upload* da estrutura SQL e/ou da base de dados populada, no mesmo local do repositório na nuvem. Como exemplo para registro de base de dados SQLite pode-se disponibilizar no serviço *Openshift*[38],

[37]Referências sobre as dimensões das telas dos iPhones - modelos atuais - encontram-se em `https://www.apple.com/br/iphone-6s/specs/` e `https://www.apple.com/br/iphone-7/specs/`. Último acesso em 22 de setembro de 2017.

[38]Referências disponíveis em `https://www.openshift.org/`. Último acesso em 21 de novembro de 2017.

**Figura 3.13:** Interface para Alteração dos Dados Específicos por Cliente.

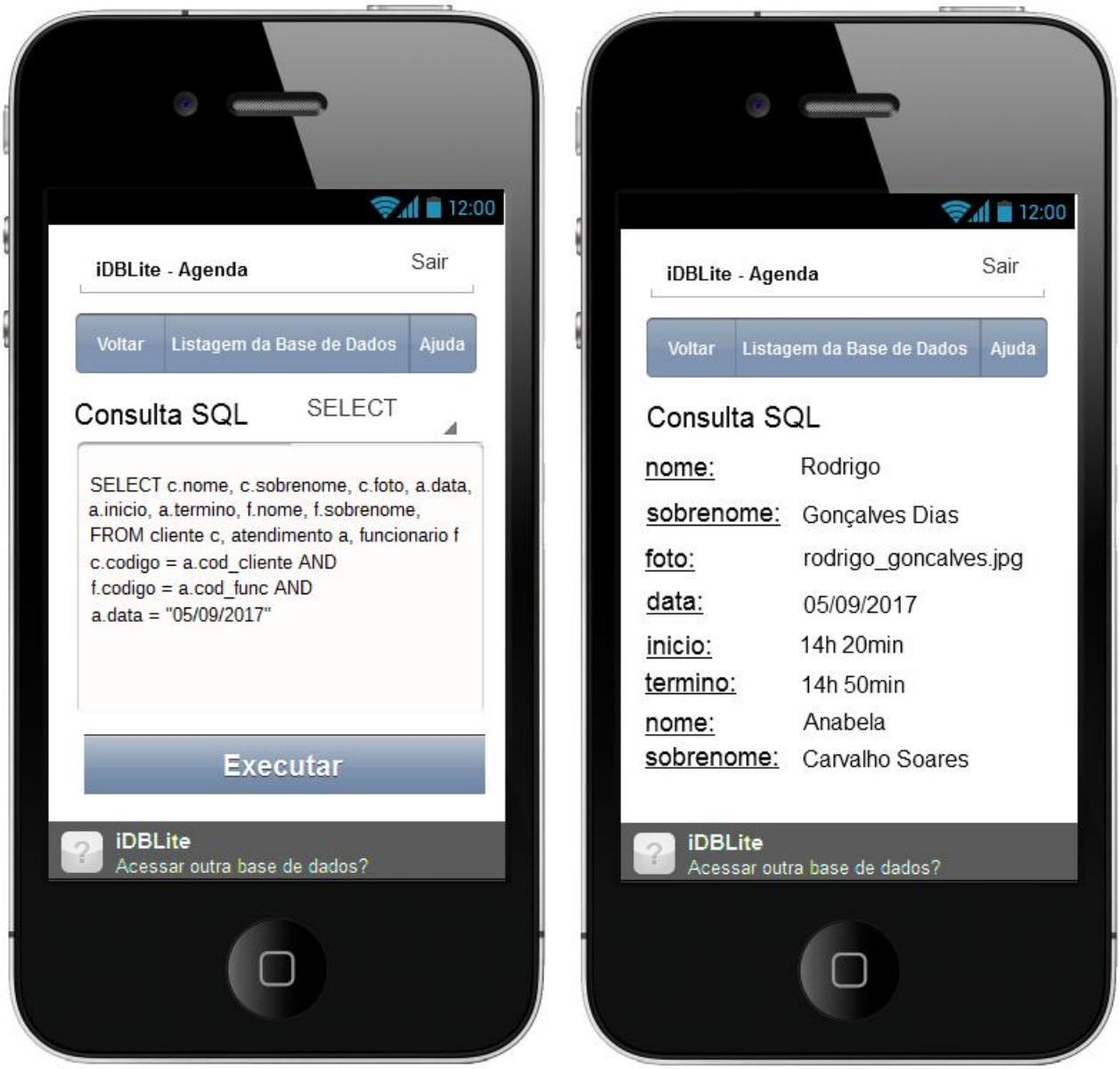

Fonte: Elaborada pelo Autor (2017).

a versão com recursos limitados e suficientes para atender a testes de acessos, ou um outro ambiente SaaS com suporte à hospedagem SQLite.

## 3.3 Orientações para o Uso do *XCODE/SWIFT*

Os recursos da ferramenta XCODE (disponíveis na versão 8.2 ou superior), para modelar interfaces de usuário, está dividido em *Views* e *Controllers*. Para cada *View* deve-se atribuir um nome, e sua referência deve ser auto explicativa. As *Views* têm *menu* superior com inspetores para fontes, objetos, formulário, imagens, cores, bordas, entre outros e tamanho - *size* (dimensões, localização dos elementos na interface, auto ajustes de posicionamentos de *layouts* ao girar o

**Figura 3.14:** Interface de Execução e Visualização de Saída.




Fonte: Elaborada pelo Autor (2017).

iPhone - função acelerômetro). As *views* são visíveis para edição direta por inserção de elementos de interface e formulários. Os *Controllers* - objeto *ViewController* são criados automaticamente com código padrão de inicialização - *init()* - e deve importar a biblioteca *UIKit*[39] no cabeçalho. Todos os recursos que estão em *UIKit* devem ser chamados por *UIViewController* na definição da classe principal, isto garante que a classe UIKit seja herdada pelo *Controller*. Os recursos (objetos) de interfaceamento estão no Editor Assistente (*Assistent editor*). Lista-se neste editor um menu com acessos rápidos para os componentes *View*, mais precisamente no sub-menu biblioteca de objetos (*Object Library*). Nesta listagem dos componentes o desenvolvedor terá a

[39] Apple Documentation Framework UIKitReferência - Disponível em `https://developer.apple.com/documentation/uikit`. Último acesso em 21 de novembro de 2017

**Figura 3.15:** Interface de Consulta SQL com Saída em Lista.


Fonte: Elaborado pelo Autor (2017).

facilidade dos acessos e inserir itens de interfaces no *View* que estiver aberto, por meio do *mouse* via ação de clicar, segurar e arrastar (*drag-and-drop*). As novas *Views* podem ser inseridas a partir da mesma listagem na biblioteca de objetos, junto aos demais itens (Knott, 2016). Encontram-se nesta listagem, além do *View Controller*, os principais objetos:

- *Storyboard* - Define parte ou um todo *ViewController*;
- *Navigation Controller* - Controle de navegação entre interfaces;
- *Table View Controller* - *View Controller* com tabelas *default*;
- *Collection View Controller* - Um *ViewController* para coleção de *Views*;

**Figura 3.16:** Interface de Consulta SQL com Saída em *Grid*.


Fonte:Elaborada pelo Autor (2017).

- *Tab Bar Controller* - Controle de Barra de Tabela;
- *Split View Controller* - Divisor de *View Controller*;
- *Page View Controller* - Cria uma página;
- *Label* - Texto fixo;
- *Button* - Botão;
- *Segmented Control* - Quebra em múltiplas *Views*;
- *Text Field* - Campo de texto;
- *Slider* - Seção de tela;
- *Activity Indicator* - Indicador de ação, semelhante a uma (splash);

- *View* - Armazena as propriedades de um *View Controller*;
- *Progress View* - Barra de progresso;
- *Map Kit View* - Objeto para a inserção de mapa;
- *Web View* - Seção de *interface* para acesso *web* na aplicação;
- *ToolBar* - Barra de tarefas;
- *Bar Button Item* - Item Botão na Barra; e
- *Container View* - Visualizador de *Container*.

Após inseridos na *View*, estes objetos podem ser selecionados (segurar botão *Control* no teclado + clicar no objeto e arrastar) e arrastados até o *Controller* associado à *View*. Um menu flutuante "perguntará"sobre o nome e funcionalidades, incluindo se será estático ou se responderá a uma ação. Um código será inserido automaticamente no *Controller* associado à View, desta forma, o desenvolvedor implementará suas lógicas de programação sobre as interfaces correlatas e as interações entre interfaces. Para inserir a biblioteca SQLite, o XCODE possui o recurso de chamada direta na *interface*, abrindo a opção "Escolha o *framework* e bibliotecas para adicionar"(*Choose frameworks and libraries to add*) e selecionando o pacote *libsqlite3.tbd*. Orienta-se o estudo das soluções de terceiros *SwiftyDB* (SwiftyDB, 2017) e *FMDB* (FMDB, 2017) (citados em notas de rodapé, na Seção 1.2). Estes são recursos propostos para "facilitar", por meio de classes Swift, os acessos e manipulações das bases de dados SQLite. Uma vez que o pacote de comunicação com o SQLite foi desenvolvido em *Objective-C* (*Cocoa Framework* - citado na Seção 3.1) será necessário criar uma ponte entre o código nativo e a linguagem Swift. Deve-se criar *Objective-C Bridging Header* na tela inicial do projeto *TARGETS* em *Swift Compiler - General*. No arquivo de biblioteca (extensão .h) a ser criado deve chamar por linha de comando - importar - a biblioteca *import <sqlite3.h>*. A partir destas configurações pode-se ter acesso as classes para manipulação SQLite (Languedoc, 2016).

## 3.4 Considerações Finais do Capítulo

O estudo abordou os conceitos da arquitetura e da estrutura do SGBDR SQLite, seção relevante para a compreensão do objeto de pesquisa. Assim, foram consideradas operações para definição e manipulação das bases de dados e as restrições nativas. O estudo aponta a solução SQLite como suporte a marcação e a sincronização de dados por XML, JSON e REST. Estes artefatos da comunicação - síncrona ou assíncrona - com um serviço em nuvem (Base de Dados na Nuvem), a depender do escopo do projeto, são os alicerces da programação para a computação móvel, permitindo a interoperabilidade dos dados. A solução visual para gerência de dados embarcados (em *smartphones*) se mostrou importante, principalmente quando há

a necessidade de gerirem as próprias bases de dados. As facilidades incluem acesso a um serviço local ou remoto, compartilhamento com um sistema *online*, visualizar, corrigir e obter relatórios, em tempo real ("a quente"ou em modo de produção). Diante da necessidade de gerir as bases em SQLite em um ambiente visual (interfacear), fez-se o estudo algébrico relacional para dimensionar as capacidades e as restrições SQL, os requisitos funcionais, a modelagem UML, as interfaces nos ciclos de vida de interfaces da Engenharia de Software Experimental e por fim a prototipação, correlacionando com as possibilidades de manipulações, escritas e visualizações.

São listados os recursos de desenvolvimento Swift/XCODE sugeridos (Seção 3.1) quanto aos suportes em interface e suas funcionalidades. As interfaces disponíveis para a etapa de desenvolvimento[40] devem ser avaliadas - definitivos - ressaltando que nunca se esgotam os incrementos e ajustes pois as metodologias e as tecnologias de desenvolvimento evoluem ou mudam.

No próximo Capítulo será discutida a Avaliação Qualitativa do Interfaceamento por especialistas na área de Tecnologia da Informação.

[40]Projeto disponível em `https://github.com/gilchristiano/idblite`. Último acesso em 15 de setembro de 2017.

# 4

# AVALIAÇÃO QUALITATIVA DO INTERFACEAMENTO

Este capítulo apresenta a avaliação empírica realizada por especialistas em Tecnologia da Informação para o interfaceamento obtido no método dos Ciclos de *Designs* Interativos da Engenharia de Software Experimental. O objetivo será apresentar a confiabilidade deste método registrando o quanto os *layouts* atendem aos seus anseios, mais precisamente como um gerenciador visual de banco de dados para a plataforma SQLite.

## 4.1 Introdução

O interfaceamento do SGBDR SQLite desenvolvido no capítulo anterior atendeu cada requisito funcional elicitado. A modelagem conceitual foi didaticamente organizada de acordo com as funcionalidades DDL, DML, DQL e DCL. Para alcançar o modelo de protótipo em conformidade com os *layouts* esperados, fez-se as depurações sucessivas das telas por meio de Ciclos de *Design* Interativos, até finalizar no protótipo mais enxuto ou menos adequado. Os requisitos funcionais constam nas interfaces do resultado final, mas por si mesmo não garantem a acertabilidade quanto aos *layouts*. Uma vez definidos os *layouts* - versão final iDBLite - torna-se necessário passar para a etapa de avaliação empírica do público alvo, especialista na área de Tecnologia da Informação. Esta etapa tem por objetivo verificar o índice de aceitação, identificar problemas de interfaces, avaliar se a proposta de interfaceamento SQLite tem aceitação e se está em conformidade com o padrão obtido ou objetivo proposto.

## 4.2 Planejamento da Avaliação

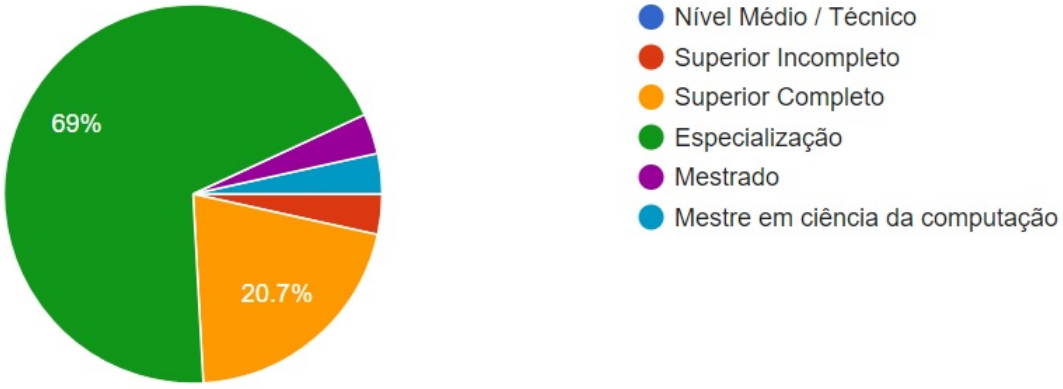
Para realizar a avaliação fez-se um questionário *online* na ferramenta *Google Forms*[1]. As questões foram organizadas para obterem informações profissionais e pareceres quanto aos interfaceamentos. A avaliação de base empírica considera as experiências dos participantes e a quantidade de opiniões. O questionário foi postado para profissionais de Tecnologia da Informação e para Docentes que, em sua maioria, são graduados em Análise de Sistemas, Sistemas de Informação, Ciência da Computação, Engenharia da Computação, Tecnólogo em Informática ou áreas correlatas.

## 4.3 Aplicação do Questionário

A avaliação foi disponibilizada *online*[2] entre os dias 26 e 28 de novembro de 2017 e obteve 29 respostas. As perguntas e as respostas obtidas estão elencadas abaixo na mesma ordem dispostas no questionário.

- Qual o seu grau de formação?

**Figura 4.1:** Especialistas por Grau de Formação




Fonte: *Elaborada pelo Autor* (2017).

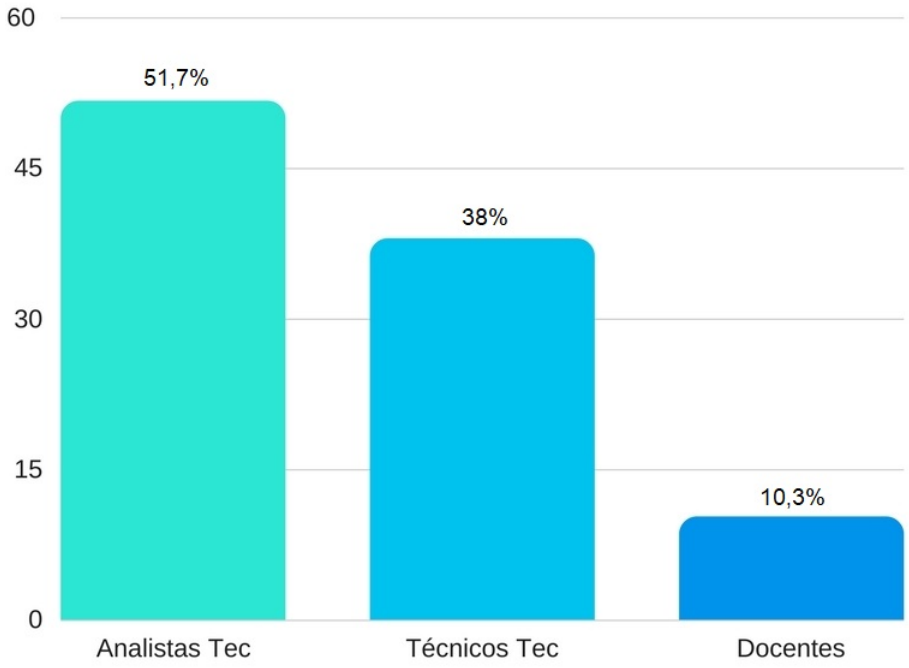
Respectivamente a Figura 4.2 apresenta: 69% de Especialistas, 6,8% com Mestrado, 3,4% com Superior Incompleto e 20,7% com Superior Completo.

- Qual a sua atividade profissional (cargo)?

[1]Aplicativo *online* para editar e publicar formulários de acessos *online*. Ferramenta de propriedade da empresa Google, possui licença de uso gratuito e está disponível no endereço eletrônico `https://docs.google.com/forms/?authuser=0`. Último acesso em 28 de novembro de 2017.

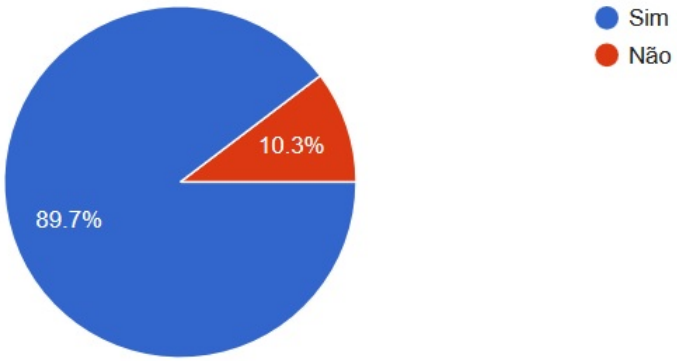
[2]Disponível em `https://goo.gl/forms/Yb2pefH21XZHJJHl1` e a versão em PDF em `https://github.com/gilchristiano/idblite/blob/master/download/questionario_mprof_dissertacao_2017.pdf`. Último acesso em 28 de novembro de2017.

**Figura 4.2:** Atividade Profissional (Porcentagem)




Fonte: *Elaborada pelo Autor* (2017).

**Figura 4.3:** Integrantes do Quadro de Tecnologia da Informação nos Institutos Federais ou Universidades Federais




Fonte: *Elaborada pelo Autor* (2017).

Conforme a Figura 4.2 as porcentagens foram: 51,7 % são Analistas de Tecnologia da Informação, Analista Desenvolvedor ou Engenheiro de Softwares, 38% são Técnicos de Tecnologia da Informação / Técnicos Programadores e 10,3% são Docentes na Área de Informática.

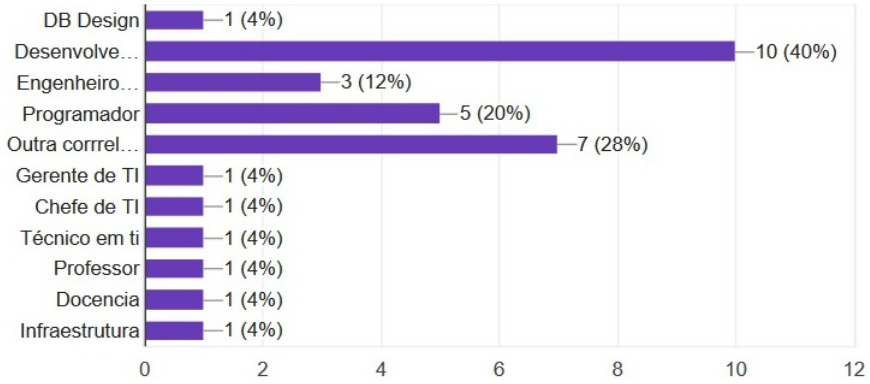
- Você é profissional integrante do quadro dos servidores dos Institutos Federais de Educação ou de Universidades Públicas ou Privadas?

  Conforme a Figura 4.3, a pesquisa aferiu 89,7% de Servidores do Quadro de Tecnologia da Informação dos Institutos Federais de Educação ou Universidades Públicas

ou Privadas e 10,3% de não integrantes.

- Qual a sua atribuição principal?

**Figura 4.4:** Distribuição de Especialistas por Atribuições

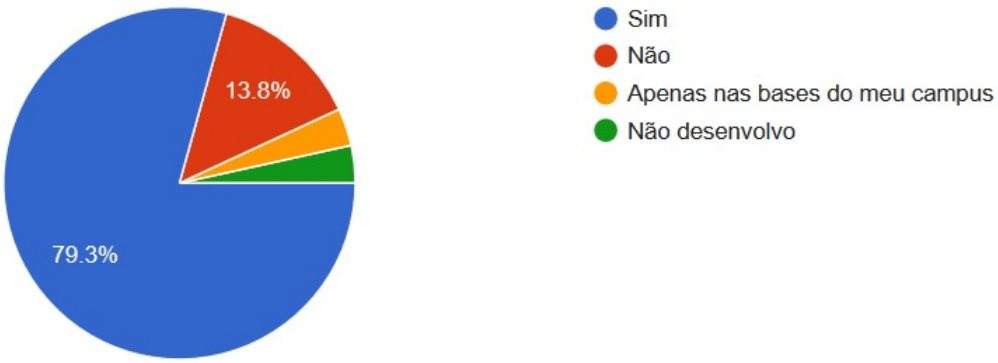



Fonte: *Elaborada pelo Autor* (2017).

As atribuições principais apresentadas (Figura 4.4) foram: DB Design com 3,4 %, Desenvolvedor com 41,4%, Engenheiro de Software com 10,3%, Programador com 24,1%, Gerente de TI com 3,4%, Chefe de TI com 3,4%, Técnico em TI com 3,4%, Docente 6,8%, Infraestrutura com 3,4 %, Suporte Técnico com 3,4 % e outra correlata com 27,6%.

- Para realizar seus trabalhos de desenvolvimento, você tem permissão de acesso às bases de dados para testes ou bases de sistemas em produção nos servidores de Banco de Dados?

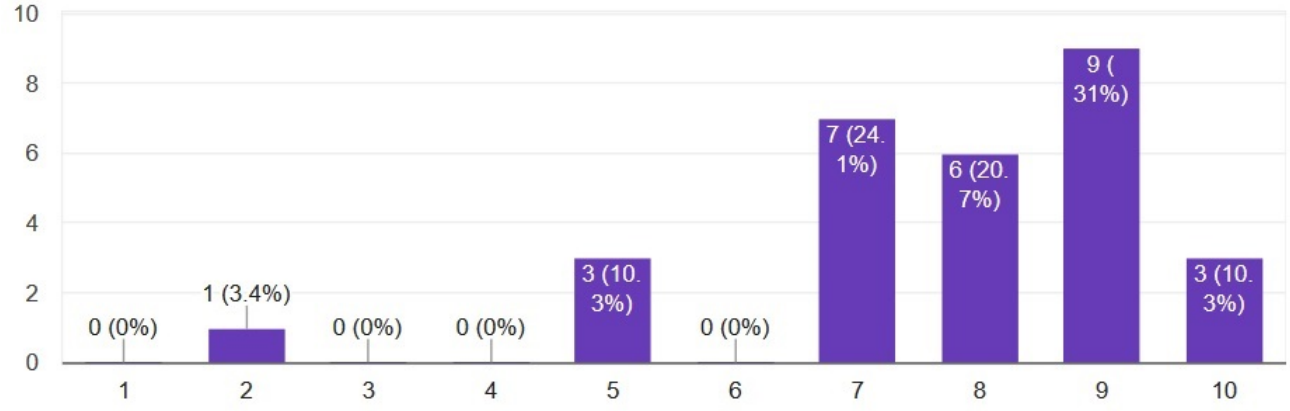
**Figura 4.5:** Permissão de Acesso às Bases de Dados




Fonte: *Elaborada pelo Autor* (2017).

- Considerando um aplicativo para Gestão de Banco de Dados SQLite em seu dispositivo (*smartphone / iPhone*) ou para acesso à base de dados remota/em nuvem. Qual o índice de importância dessa solução em seu trabalho?

**Figura 4.6:** Índice de Importância da Solução no Trabalho



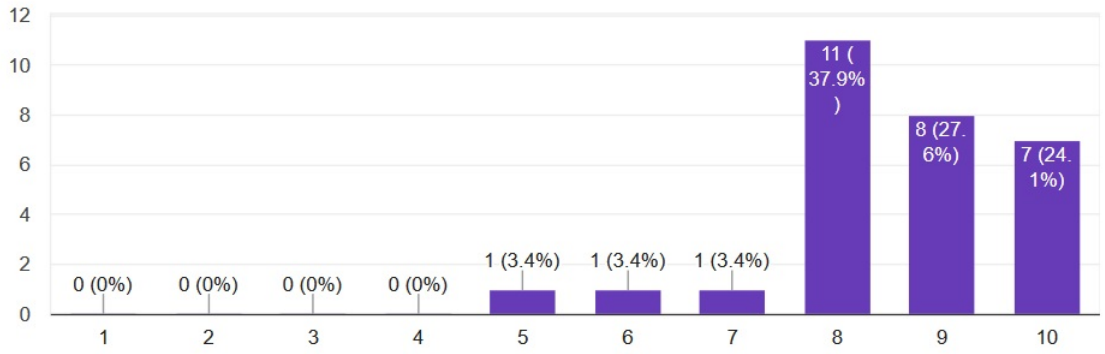

Fonte: *Elaborada pelo Autor* (2017).

Conforme a Figura 4.6 constando a porcentagem de especialistas quanto as notas que atribuíram (importância da aplicação para seus trabalhos) foram: 10,3% deram nota 10, 31% deram nota 9, 20,7% deram nota 8, 24,1% deram nota 7, 10,3% deram nota 5 e 3,4% deram nota 2.

- Com base em seus conhecimentos de usabilidade em Gerenciadores de Banco de Dados, atribuir notas qualitativas ao interfaceamento:
  - Interfaceamento DDL (Linguagem de Definição de Dados)
    1. Criar base de dados;
    2. Interface de acesso à base de dados selecionada; e
    3. Opções básicas de criação e exclusão de Tabela.

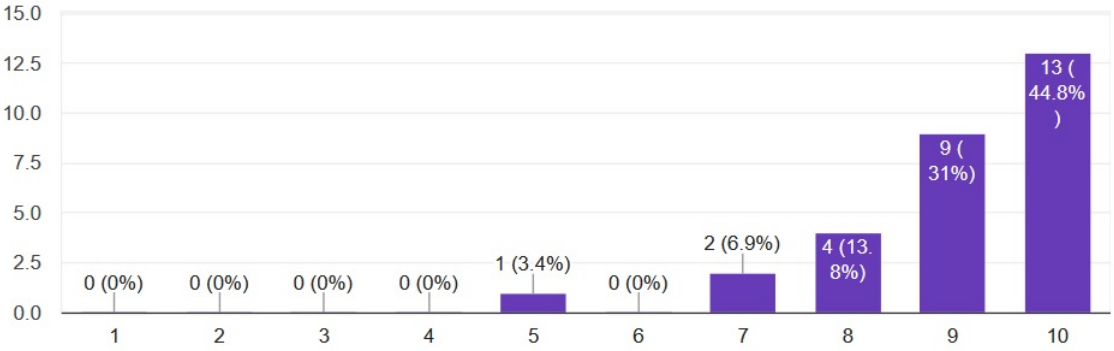
    A partir das avaliações das interfaces DDL (Figura 4.7) aferiu-se as notas: 24.1% (7 especialistas) deram nota 10, 27.6% (8 especialistas) deram nota 9, 37.9% (11 especialistas) deram nota 8, 3.4% deram nota 7, 3.4% deram nota 6 e 3.4% deram nota 5.
  - Interfaceamento DML (Linguagem de Manipulação de Dados)
    1. Inserção do campo 'codigo' na Tabela Cliente; e
    2. Listagem e dos campos para edição.

**Figura 4.7:** Notas Atribuídas ao Interfaceamento DDL




Fonte: *Elaborada pelo Autor* (2017).

**Figura 4.8:** Notas Atribuídas ao Interfaceamento DML




Fonte: *Elaborada pelo Autor* (2017).

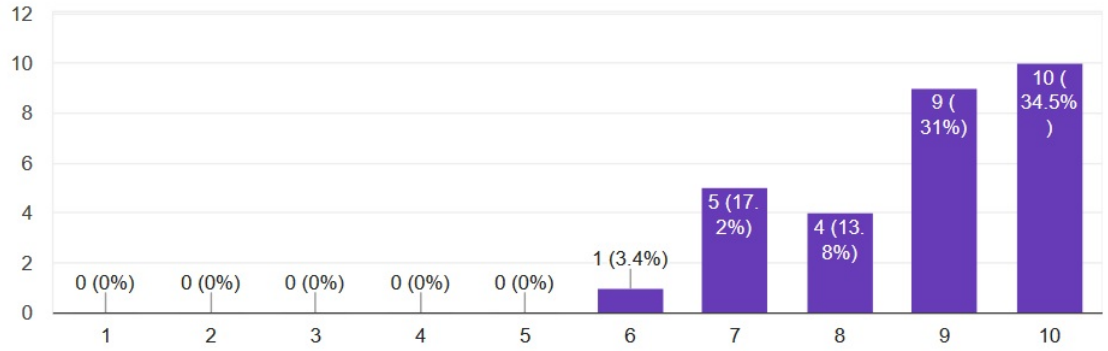
A partir das avaliações das interfaces DML (Figura 4.10) obteve-se as notas: 44.8% deram nota 10, 31% deram nota 9, 13.8% deram nota 8, 6.9% deram nota 7, e 3.4% deram nota 5.

- Interfaceamento DQL (Linguagem de Consulta de Dados)
    1. Interface de acesso SQL e inserção de Consulta;
    2. Listagem com o resultado da Consulta; e
    3. Seleciona o Usuário, clica em 'Alterar dados' e abrirá a interface para editar diretamente os dados pessoais.

    A partir das observações das interfaces de consultas obteve-se: 34.5% deram nota 10, 31% deram nota 9, 13.8% deram nota 8, 17.2% deram nota 7, e 3.4% deram nota 6.

- Linguagem de Controle de Dados (DCL - Linguagem de Controle de Dados)
    1. Acessar as Configurações para Gerência de Usuários;
    2. Acessar o Ambiente de Exclusão de Usuário; e

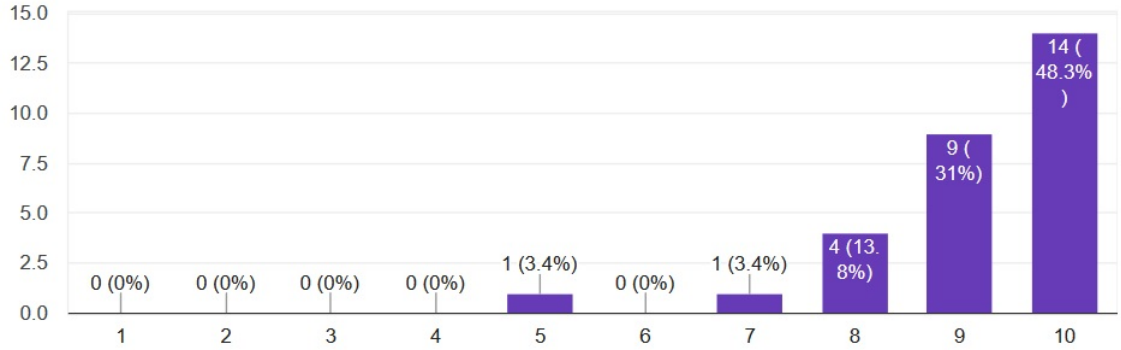
**Figura 4.9:** Notas Atribuídas ao Interfaceamento DQL




Fonte: *Elaborada pelo Autor* (2017).

3. Ambiente de Gerência de Privilégios de Usuário (SQL *GRANT/ REVOKE*).

**Figura 4.10:** Notas Atribuídas ao Interfaceamento DCL




Fonte: *Elaborada pelo Autor* (2017).

As notas atribuídas para o interfaceamento da Linguagem de Controle de Dados foram: 48,3% deram nota 10, 31% deram nota 9, 13.8% deram nota 8, 3,4% deram nota 7, e 3.4% deram nota 5.

- Com base nas propostas de interfaceamento acima elencadas, quais requisitos você poderia acrescentar para melhorias na proposta de desenvolvimento?

    1. Somente no melhoramento das *Integrated Development Environment* (IDE)s - "Ambientes Integrados de Desenvolvimento- Versão Android;
    2. Muito boa a identidade visual que foi criada. Contribuiria com a criação de ícones ou imagens para identificar melhor cada tela/ambiente, por exemplo, a tela de inclusão de banco de dados, tabelas e outros;

3. Muito boa a proposta;
4. Possibilidade de exportar dados. Muito bom o trabalho;
5. Não é necessário me atende perfeitamente;
6. A digitação em dispositivos móveis, principalmente em *smartphones*, é dificultosa pois os teclados *touch* não tem a mesma praticidade que um teclado físico. Diante disso, a interface no dispositivo móvel deve oferecer consultas pré formatados ou modelos de consultas para que o usuário tenha que digitar o menos possível;
7. A lista de resposta de seleção poderia ser em menos linhas para visualizar mais resultados na mesma tela; e
8. Possibilidade de ver todas as tabelas / *views* / funções que fazem uso de determinada tabela. Para poder ver o impacto que uma alteração pode causar.

A avaliação alcançou o público alvo de forma que todos que assinalaram informam ter vínculo com a Tecnologia da Informação ou com a Docência nesta área. Os ambientes foram dispostos conforme as sequências de usos, em grupos de *layouts* para um mesmo conjunto funcional. Estes ambientes têm sua origem nos estudos da estrutura base do SQLite, nos estudos das fundamentações algébricas e posteriormente inseridos nos Ciclos de *Design* Interativos. As avaliação dos *layouts* das linguagens DDL, DML, DQL e DCL tiveram pontuações altas e muitas aprovações, infere-se que existem familiaridades dos especialistas quanto as interfaces para um SGBDR. Caso estivessem em desacordos com os objetos propostos nos *layouts* habituais ou em disposições equivocadas, os especialistas teriam sinalizados na questão aberta. Foram apresentadas 8 (oito) sugestões, onde ressalta-se a "Possibilidade de exportar dados", a qual foi prevista nos *frameworks* atuais, no exemplo na Seção 2.5 e no segundo Requisito Funcional na Seção 3.2.1. Os demais detalhes dos resultados são apresentados na próxima seção.

## 4.4 Análise dos Resultados

As avaliações foram realizadas por 29 profissionais diretamente ligados à área de Tecnologia da Informação, sendo aproximadamente 90% servidores do Governo Federal ou Universidades Públicas ou Privadas. A maioria dos avaliados é desenvolvedor, gestor ou docente, os quais fazem uso ou têm conhecimentos em gerenciadores de banco de dados. O questionário obteve respostas específicas para cada grupo de interfaces para acessos e operações em base de dados avulsa. A elaboração das interfaces pelo método dos Ciclos de *Designs* Interativos se mostrou coerente, observa-se que 79,3% dos avaliados utilizam dos recursos de acessos aos banco de dados de testes ou em produção, 3,4% bases de dados no *campus*, e aplicaram a nota média ponderada de 7,7 em uma escala de 1 a 10, considerando a proposta do protótipo uma solução

aplicável em seus serviços laborais. Os *layouts* tiveram uma nota média ponderada de aceitação de 8,5 para DDL, nota média ponderada de 9 para DML, nota média ponderada de 8,7 para DQL e nota média ponderada de 9,1 para DCL. Estes resultados são observações qualitativas e com base em experiências habituais de interações com interfaces de sistemas, tratando-se de valores pessoais - não-determinístico - semânticos e intuitivos. As respostas das perguntas abertas foram sugestões, observações relevantes e elogios. Destacam-se as sugestões e observações:

- A sugestão de criar o mesmo protótipo de interfaceamento, para atender também ao Sistema Operacional Android (desenvolvimento híbrido);
- Disponibilizar ícones ou botões de acessos para digitações em consultas. Interface flutuante com palavras reservadas para consultas SQL;
- Sugestão para aumentar a quantidade de linhas para ocupar mais informações na saída; e
- Sugestão para a listagem de todas as tabelas e *views*, com possibilidade de ver as relações, ou destacar as relações.

As sugestões e elogios são construtivas e devem ser re-inseridas no ciclo de *Design* para refatorar o projeto na próxima versão.

## 4.5 Considerações Finais do Capítulo

Este capítulo apresentou o método de avaliação empírica, o planejamento e a aplicação do questionário a um grupo de especialistas em Tecnologia da Informação. Justificou-se a necessidade da avaliação para subsidiar o método de interfaceamento pelos Ciclos de *Design* e obter o seu índice de acertabilidade e confiabilidade. Analisou-se as respostas considerando-se as médias das pontuações, as quais no âmbito geral oscilaram entre 8 e 9. Os *layouts* tiveram boas aceitação, confirmando que o método de prototipação de interfaces se mostrou viável para os ciclos de desenvolvimento de software. Conclui-se que os sistemas informatizados em geral estão em constantes ciclos de versões, e as interfaces são mais próximas dos anseios dos usuários quanto mais ocorrerem refatoramentos e as participações destes nas elaborações e avaliações.

# 5
# CONCLUSÕES

Este capítulo tem por objetivo apresentar as considerações finais, os tópicos relevantes da dissertação, as principais contribuições, as sugestões de trabalhos futuros e as limitações do trabalho.

## 5.1 Considerações Finais

Este estudo abordou por meio da pesquisa da Plataforma SQLite, os conceitos e os fundamentos abrangentes da sua implementação, considerando a estrutura, a arquitetura e os conceitos algébricos relacionais. Fez-se referências e observações em trabalhos relacionados. Um breve estudo das métricas e métodos de interfaceamento com enfoque no Ciclo de Vida do *Design* de Interfaces que foi elaborado. Aplicou-se o método dos Ciclos de *Designs* Interativos da Engenharia de Software Experimental na prototipação do interfaceamento iDBLite, seguindo os requisitos funcionais para atender a um SGBDR SQLite para dispositivos móveis. Foram utilizadas as soluções visuais da ferramenta XCODE/Swift da Apple e procedimento de conexão com o SQLite. Para avaliar os resultados dos interfaceamentos, aplicou-se o método empírico por meio de questionário aos especialistas em Tecnologia da Informação, colhendo seus pareceres sobre o interfaceamento, seus índices de aceitações ou conformidades com o objetivo proposto.

## 5.2 Principais Contribuições

As principais contribuições deste trabalho são detalhadas a seguir de acordo com os itens elencados como objetivos específicos na Seção 1.3.2.

- Foi apresentada uma pesquisa dos fundamentos e dos conceitos sobre a arquitetura e a estrutura do SGBDR SQLite;
- Discorreu-se sobre as fundamentações algébricas para consultas relacionais no SQLite;

- Aplicou-se as recomendações da Apple para *design* de interfaces, utilizando-se dos conceitos de métricas de *design* e de usabilidade;
- Fez-se um estudo descritivo e aplicou-se no interfaceamento com enfoque ao *layout* do iPhone. Foi analisada a ferramenta XCODE/Swift bem como *frameworks* para desenvolvimento *mobile*, conclui-se que este possui recursos para desenvolvimento da solução iDBLite;
- Foi desenvolvido o protótipo dos *layouts* - iDBLite[1] - com uma estrutura para acesso público, aberta para contribuições e desenvolvimento; e
- Avaliou-se os *layouts* do interfaceamento por meio de questionário de pesquisa de opinião do público alvo. Concluiu-se que o método de prototipação pelos Ciclos de *Design* Interativos, com as refatorações das interfaces, aproxima o protótipo da versão para por em produção, com interface amigável, auto-explicativa e intuitiva.

## 5.3 Limitações do Trabalho

Este trabalho apresentou algumas limitações:

- As constantes *releases*[2] do SQLite podem alterar ou inserir novas funcionalidades que tornam necessário reavaliar o protótipo para novos interfaceamentos;
- As atualizações do XCODE/Swift podem comprometer os projetos em fase de desenvolvimento, pois a prioridade da empresa é atender aos seus recursos; e
- Não há solução nativa do SQLite para sincronia com outras bases de dados de outros gerenciadores relacionais, devendo ser implementado por um módulo externo.

## 5.4 Trabalhos Futuros

Recomenda-se desenvolver as seguintes pesquisas diretas ou correlacionadas:

- Refazer os testes de interfaceamento aplicando outras técnicas, a exemplo da Avaliação Analítica[3] ou Heurística[4];

[1]Apresentação do iDBLite em `https://gilchristiano.github.io/idblite/` e documentação disponível em `https://github.com/gilchristiano/iDBLite_Cin_UFPE`. Último acesso em 28 de novembro de 2017.

[2]SQLite mudanças (*releases* disponível em `https://sqlite.org/changes.html`. Último acesso em 28 de novembro de 2017.

[3]Referência em *Usability Inspection Methods*, por Robert L. Mack e Jakob Nielsen (1995). Disponível em `https://doi.org/10.1016/B978-0-08-051574-8.50020-0`. Último acesso em 28 de novembro de 2017

[4]Referência encontra se em *Heuristic Evaluation of User Interfaces*, por Jakob Nielsen e Rolf Molich. Disponível

- Implementar uma solução visual seguindo os modelos propostos neste estudo e aplicá-los em projetos pilotos;
- Estender as funcionalidades por meio de API para sincronizações com outras bases em SQLite, incluindo os gerenciadores de banco de dados em nuvem;
- Estudar outras técnicas para sincronizações de bases de dados e "conversas"com outros SGBDR, a exemplo do Postgres[5], MySQL[6] ou MariaDB[7];
- Estudar outras ferramentas de desenvolvimento *Mobile* híbridas para interfaceamento e implementação do SQLite; e
- Propor artifícios algébricos relacionais para resolver funcionalidades não atendidas nativamente pelo SQLite e disponibilizá-los no protótipo.

em `https://dl.acm.org/citation.cfm?id=97281&CFID=1011438119&CFTOKEN=50453572` e, referência de trabalho em *How to Conduct a Heuristic Evaluation*, por Jakob Nielsen. Disponível em `https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/`. Últimos acessos em 28 de novembro de 2017.

[5]Disponível em `https://www.postgresql.org/`. Último acesso em 21 de novembro de 2017.

[6]Disponível em `https://www.oracle.com/mysql/index.html`. Último acesso em 21 de novembro de 2017.

[7]Disponível informações em `https://mariadb.org/`. Último acesso em 21 de novembro de 2017.

# Referências

Ambiente Red Hat Openshift. Solução SaaS para o desenvolvimento de software e a sua execução em nuvem. Encontra-se disponível no endereço eletrônico `https://www.openshift.com/`. Último acesso em 26 de agosto de 2017.

ANSI. *American National Standards Institute*: Disponível em https://www.ansi.org/, equivalente a ABNT (Associação Brasileira de Normas Técnicas). Último acesso em 20 de fevereiro de 2017.

Banco de dados não-relacional (NoSQL) CouchDB. (*Cluster of Unreliable Commodity Hardware*) - Disponível no link original `http://www.unqlspec.org/display/UnQL/Home`, o qual redireciona para `https://developer.couchbase.com/open-source-projects` e para o projeto original NoSQL em `http://couchdb.apache.org/`. Banco de dados não-relacional orientado a documentos, o qual utiliza a API JSON RESTful para acesso. Sob licença de uso Apache e encontra-se disponível para contribuições. Últimos acessos em 14 de fevereiro de 2017.

Basili, V. C. G. R. Experience Factory, Encyclopedia of Software Engineering. **ResearchGate**, [S.l.], n.Software Engineering, Experimental Process Software, p.469–476, 1994.

Bezerra, E. **Princípios de Análise e Projeto de Sistema com UML**. [S.l.]: Elsevier Editora, 2014.

Bhosale, S. SQLite: light database system. , [S.l.], p.882–885, 2015.

Cabral, D. R. **Visualização e Manipulação de Dados em Dispositivos Móveis.** Monografia (Bacharelado em Informática), UFPE (Universidade Federal de Pernambuco), Recife, Brazil.

Chees, B. J. S. **Computação em Nuvem - Tecnologias e Estratégias**. [S.l.]: M.Books do Brasil, São Paulo, 2013.

Codd, E. F. Introdução à Engenharia de Software Experimental. , [S.l.], p.377–387, 1970.

Couchbase. **Couchbase**:. Mais informações em `https://www.couchbase.com/nosql-databases/couchbase-mobile`. Último acesso em 20 de março de 2017.

Capacidade de Gerência Automática de Memória do Swift. Sobre a capacidade de gerência automática de memória (RAM), disponível em `https://goo.gl/fd2uwt`. Último acesso em 22 de setembro de 2017.

Carissimi , A. S. T. S. O. S. **Sistemas Operacionais. 4ª ed. Porto Alegre**: bookman, 2010. (livros didáticos informática ufrgs, v. 11). [S.l.: s.n.], 2010.

Conexão Genérica Java ME. (*Java Micro Edition*) - GCF É uma solução do pacote *Mobile Information Device Profile* (MIDP) 2.0; *este possui 20 métodos para trabalhar especificamente com Hypertext Transfer Protocol (HTTP). O MIDP 2.0, é o único protocolo que com certeza está implementado é HTTP. Por meio da classe HttpConnection, você pode se comunicar com*

*um servidor Web ou com qualquer dispositivo remoto que suporte HTTP*. Disponível em http://www.oracle.com/technetwork/systems/gcf-156792.html. Último acesso em 17 de abril de 2017.

CMMI - Sobre Maturidade de Software. Refere-se a maturidade no ciclo de desenvolvimento do SQLite, definido com o alto grau de qualidade e com mínimas ocorrências de erros. Mais informações sobre maturidade de software - *Capability Maturity Model Integration*(CMMI) nos endereços eletrônicos https://www.sei.cmu.edu/cmmi/ e http://cmmiinstitute.com/. Últimos acessos em 01 de outubro de 2017.

CVS. **Solução CVS**. *Concurrent Versions System* - O Sistema de Controle de Versões permite o versionamento de arquivos em diretórios guardando seus contextos de logs de tempo para usuários e suas manipulações. Mais informações disponíveis em http://www.nongnu.org/cvs/. Último acesso em 15 de novembro de 2017.

Detalhes sintáticos para os usos dos tipos LIKE e GLOB. Mais informações sobre a construção sintática (expressões regulares) para os tipos LIKE e GLOB disponível no endereço eletrônico https://www.sqlite.org/lang_expr.html#glob. Último acesso em 10 de novembro de 2017.

Detalhes sobre a implementação do tipo BLOB. Referências sobre a interface de implementação em linguagem C disponíveis em https://www.sqlite.org/c3ref/blob.html. Último acesso em 10 de novembro de 2017.

Estatística de Vulnerabilidades do Apple Iphone OS. Informações disponíveis em https://www.cvedetails.com/product/15556/Apple-Iphone-Os.html?vendorid=49. Último acesso em 22 de setembro de 2017.

FileMaker em Referências. FileMaker é um *SGBDR* multiplataforma com ambiente *GUI* (*Graphical User Interface*) no *frontend* (interface de usuário DB *Design*). Atualmente é uma subsidiária da Apple. Mais informações disponíveis em http://www.filemaker.com/br/. Último acesso em 02 de janeiro de 2017.

FMDB. **Solução FMDB**. *A Cocoa / Objective-C wrapper around SQLite* - FMDB Versão 2.7 está disponível no endereço eletrônico https://github.com/ccgus/fmdb. Último acesso em 02 de novembro de 2017.

Godfrey, N. **Agile Swift: swift programming using agile tools and techniques**. [S.l.]: Apress, 2017. 173p.

HEROKU. **Heroku**:. Refere-se a uma Plataforma como Serviço (*PaaS* – Platform as a Service). Encontra-se disponível no endereço eletrônico https://www.heroku.com/. Último acesso em 20 de março de 2017.

HyperSQL em Referências. Version 2.3 - Documentação disponível em http://hsqldb.org/web/hsqlDocsFrame.html e http://hsqldb.org/doc/guide/running-chapt.html. Últimos acessos em 02 de fevereiro de 2017.

HSQLDB HyperSQL em Referências. Encontra-se disponível no endereço eletrônico http://hsqldb.org/. Último acesso em 02 de fevereiro de 2017.

Informações sobre o SO Android. O endereço para suporte aos desenvolvedores encontra-se em https://developer.android.com/index.html. As classes para gerenciamento a partir da API Android, para desenvolvimento de aplicativos com Banco de Dados Relacional SQLite (sqlite3) encontra-se em https://developer.android.com/reference/android/database/sqlite/package-summary.html. Últimos acessos em 24 de outubro de 2017.

JEON, S. A Recovery method of deleted record for SQLite database. **Springer-Verlag**, [S.l.], v.16, n.16, p.707–715, 2011.

Knott, M. **Beginning Xcode - Swift 3 Edition**. [S.l.: s.n.], 2016. www.apress.comp.

Languedoc, K. **Build iOS Database Apps with Swift and SQLite**. [S.l.]: Bookman, UFRGS, 2016.

Lecheta, R. R. **Desenvolvendo para iPhone e iPad: aprenda a desenlvolver aplicações utilizando o ios sdk**. [S.l.: s.n.], 2014.

Lewis, H. P. C. **Elementos de Teoria da Computação**. [S.l.]: Apress, 2000.

Linguagem C. Linguagem de programação imperativa, procedural, de propósito geral, compilada (executa diretamente no processador), padronizada pela ISO (*International Organization for Standardization*). O compilador C foi implementado em 1972 por Dennis Ritchie na empresa AT&T Bell Labs. Mais informações no endereço eletrônico https://www.bell-labs.com/usr/dmr/www/ elaborado em homenagem póstuma ao Dennis Ritchie. Último acesso em 20 de fevereiro de 2017.

iOS. Disponível no endereço eletrônico https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/PersistentStoreFeatures.html. Último acesso em 01 de outubro de 2017.

Muchow, J. W. **Core J2ME - Technology  MIDP**. [S.l.: s.n.], 2004. 417–420p.

Navathe, E. **Sistemas de Banco de Dados**. [S.l.]: Pearson education, Inc, 2011.

Newman, C. **SQLite: a pratical guide to using administering, and programming the database bundled with php 5**. [S.l.: s.n.], 2004. 1–78p.

Núcleo de Dados para Programação. SQLite *iPhone OS Data Management*, disponível em *Core Data framework*. Acesso em https://developer.apple.com/library/content/releasenotes/General/WhatsNewIniOS/Articles/iOS10.html. Último acesso em 25 de março de 2017.

Openshift. **OpenShift**:. Solução disponível em https://www.openshift.com/. Plataforma *online* para gestão de infraestrutura de dados. Último acesso em 20 de março de 2017.

Owens, M. **The Definitive Guide to SQLite**. [S.l.]: Apress, 2006.

PaaS *Platform-as-a-Service*. Estrutura física dedicada para hospedagem onde se tem em produção um conjunto de hardware - com limite de recursos pré-definido - configurável pelo cliente onde este sobe um sistema operacional (virtualizado) para atender a um serviço

apropriado. Como exemplo de soluções têm-se o ESX VMWare e OpenShift respectivamente disponíveis em
https://www.vmware.com/radius/saas-paas-and-iaas-the-basics/ e
https://www.openshift.com/. Últimos acessos em 21 de fevereiro de 2017.

Pereira, M. T. Forensic analysis of the Firefox 3 Internet history and recovery of deleted SQLite records. **Elsevier**, [S.l.], v.16, n.5, p.93–103, 2009.

Price, A. M. A. **Implementação de Linguagens de Programação**: compiladores. [S.l.]: Bookman, UFRGS, 2006.

Persistência de dados em JavaME. (*Java Micro Edition*) - *Record Management Store* (RMS) Encontra-se em *Package* javax.microedition.rms: Trata-se de um mecanismo de persistência para armazenamento e recuperação de dados. Documentação disponível em
http://docs.oracle.com/javame/8.0/api/meep/api/javax/microedition/rms/package-summary.html. Última *release* do JavaME em abril de 2014. Último acesso em 11 de abril de 2017.

Principais empresas que usam SQLite. Informações sobre as principais empresas que fazem uso do SQLite estão disponíveis em http://sqlite.org/mostdeployed.html e
https://www.sqlite.org/famous.html. Últimos acessos em 01 de outubro de 2017.

REST. **API REST**. Referências em http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm e API Rest Node.JS
https://nodejs.org/en/. Último acesso em 22 de novembro de 2017.

Referências sobre DB Browser para SQLite. *é uma aplicação visual gratuita, bilicenciada sob a Mozilla Public Licence Versão 2 e a GNU General Public Licence Versão 3 ou posterior para criação, desenho e edição de Banco de Dados SQLite*. Referência citada na aplicação na aba Sobre. A aplicação e demais informações encontram-se disponíveis em
http://sqlitebrowser.org/, e o projeto está disponível no repositório GitHub em
https://github.com/sqlitebrowser/sqlitebrowser. Últimos acessos em 10 de outubro de 2017.

Rogers, S. apud. **Engenharia de Software**. [S.l.]: Pearson Education, 2006.

Rosa, L. C. B. **Técnicas de armazenamento de dados locais em ambiente Android**. Instituto Universitário de Lisboa – Portugal, 2015.

Rede Social Facebook. Maior rede social do mundo (nesta data). Ambiente online para publicação de textos e mídias e comunicação síncrona por (*chat* ou conversas por textos. Acesso no endereço https://www.facebook.com/. Último acesso em 22 de novembro de 2017.

Rede Social Twitter. Rede social de compartilhamento de texto e mídias. Ambiente disponível em https://twitter.com. Último acesso em 22 de novembro de 2017.

Referências sobre a Linguagem C++. Disponíveis nos endereços
http://en.cppreference.com/w/ e http://www.cplusplus.com/. Últimos acessos em 20 de março de 2017.

Silberschatz, A. **Sistema de Bancos de Dados**. [S.l.]: Elsevier, 2012.

Silveira, G. J. **Swift**: programe para iphone e ipad. [S.l.]: Casa do Código: Série Caelum, 2016.

Sobre a Apple Security Framework. As informações sobre as medidas de segurança nos *frameworks* da Apple estão disponíveis em `https://developer.apple.com/documentation/security`. Último acesso em 22 de setembro de 2017.

SwiftyDB. **Solução SwiftyDB**. Encontra-se disponível em `http://oyvindkg.github.io/swiftydb/`. Último acesso em 02 de novembro de 2017.

Serviço de Mensagens Curtas. *Short Message Service*: Serviço de Mensagens Curtas é um meio de troca de comunicações entre usuários por meio de aparelhos celulares – dispositivos antecessores aos *smartphones* – e que ainda existem como um dos atuais recursos. Informações disponíveis em projeto Kannel: *Open Source WAP and SMS gateway* disponível em `http://www.kannel.org/news.shtml#1.4.1`. Último acesso em 08 de março de 2017.

Sobre a API ZettaJS. ZettaJS: Anunciada como a primeira API para Internet das Coisas; mais informações disponíveis em `http://zettajs.org`. Último acesso em 26 de agosto de 2017.

Sobre o Node JS. Mais informações específicas em Node.js nos endereço eletrônico oficial `https://nodejs.org/en/`. *Plugins* disponíveis em `https://github.com/JayrAlencar/sqlite-sync.js` e `https://github.com/JayrAlencar/sqlite-sync.js/wiki`. Solução proprietária SQLite para sincronização no endereço `http://sqlite-sync.com/` e `https://github.com/alixaxel/ArrestDB`. Últimos acessos em 22 de novembro de 2017.

Sobre o Pilot-DB em Referências. Mantido por Marc Chalain (marc-chalain@voila.fr) e aplicativo disponível em `http://pilot-db.sourceforge.net/`. Último acesso em 23 de fevereiro de 2017.

Sobre o Windows *Mobile*. Mais informações disponíveis no endereço eletrônico `https://www.microsoft.com/en/mobile/`. Último acesso em 01 de outubro de 2017.

Solid-State Drive. Unidade de armazenamento nãovolátil que substitui os discos rígidos dos computadores. Mais informações disponíveis em `http://www.semiconductorstore.com/blog/2014/The-Development-and-History-of-Solid-State-Drives-SSDs/854`. Último acesso em 25 de março de 2017.

Suporte no Google Chrome para Web SQL Database. Google Chrome: Suporte para Web SQL Database na versão 4, baseado em SQLite. A versão do SQLite pesquisada foi a *Release* 3.6.19 de 14 de outubro de 2009. Informações disponíveis em `https://dev.w3.org/html5/webdatabase/#databases` e `http://www.sqlite.org/releaselog/3_6_19.html`. Últimos acessos em 13 de maio de 2017.

SQLite para Linux. Orientações disponíveis nos endereços eletrônicos `http://www.devfuria.com.br/linux/instalando-sqlite/` e pela

Comunidade Linux Mint em
`https://community.linuxmint.com/software/view/sqlite3`. Últimos acessos em 01 de outubro de 2017.

SQLite para unix. *Orientações para iniciar o uso do SQLite em Sistemas Operacionais Unix elaborado e publicado pela Universidade de Stanford. Disponível no endereço eletrônico `https://cs.stanford.edu/people/widom/cs145/sqlite/SQLiteIntro.html`. Último acesso em 01 de outubro de 2017.*

SQLite no Android. Orientações para o uso do SQLite em aplcativos no Sistema Operacional Android estão disponíveis em
`https://sqlite.org/android/doc/trunk/www/index.wiki` e documentação em `https://sqlite.org/android/doc/trunk/www/install.wiki`. Últimos acessos em 01 de outubro de 2017.

SQLite no Mac OSX. (versão 10.9) - Informações disponíveis nos endereços eletrônicos `https://developer.apple.com/legacy/library/documentation/Darwin/Reference/ManPages/man1/sqlite3.1.html` e sobre ambientes de gerenciamentos em
`https://www.slant.co/topics/6655/~mac-os-x-guis-for-sqlite`. Últimos acessos em 01 de outubro de 2017.

SQLite no Windows. SQLite via linha de comando do Shell do Windows para SQLite com orientações para iniciar, executar, acesso aos comandos (conhecido por ponto-comandos), suas regras e funcionalidades estão disponível no endereço eletrônico https://sqlite.org/cli.html. Último acesso em 01 de outubro de 2017.

SQLite para WinCE. Uma enginede de banco de dados embarcada. Disponível em
`http://sqlite-wince.sourceforge.net/index.html`. Último acesso em 01 de outubro de 2017.

SQLite para WinRT. Informações disponíveis em `https://sqlwinrt.codeplex.com`. Último acesso em 01 de outubro de 2017.

Travassos, G. H. Introdução à Engenharia de Software Experimental. , [S.l.], p.882–885, 2002.

TweetDeck - Localização dos dados em SQLite. Para confirmação do repositório de dados armazenados em SQLite no *TweetDeck* (após instalado) segue-se o caminho "Library/Preferences/TweetDeckFast.[palavra formada por letras e números randômicos]/Local Store", no arquivo td_??_user.db. Último acesso em 11 de novembro de 2017.

Wazlawick, R. **Análise e Projeto de Sistemas de Informação Orientados a Objetos. Série SBC, Sociedade Brasileira de Computação.** [S.l.: s.n.], 2011. 35–42p.

Wohlin, C. O. Experimentation in Software Engineering: an introduction. **Kluwer Academic Publishers, USA - Relatório Técnico UFRJ:RT-ES-590-02**, [S.l.], 2012.

XML. **XML**:. Desenvolvido e mantido pela W3C *World Wide Web Consortium*. Mais informações disponíveis em https://www.w3.org/XML/. Último acesso em 15 de novembro de 2017.

Yvonne, R. **Design de interação**: além da interação humano-computador. [S.l.]: Bookman, 2013.

# Apêndice A - Modelagem Conceitual para uma Seção de Ambiente Acadêmico

Este apêndice apresenta uma modelagem conceitual de um mini-mundo, conforme Figura A.1, reproduzindo a dinâmica de uma seção de Ambiente Acadêmico [1] via roteiro abaixo. A modelagem é aplicada nas seções das consultas DQL das correlações entre as formalizações da álgebra relacional e as consultas em SGBDR. O roteiro descritivo lista atributos e relações discriminados em: instituição, cursos, turmas, disciplinas, docentes e alunos:

- Uma Instituição possui os atributos: identificador (id), nome, sigla, seu endereço e telefones para contatos, portaria do Diário Oficial da União (D.O.U) e informações adicionais que o usuário queira registrar; a instituição possui cursos e cada curso está vinculado unicamente a um instituto. A relação de pertinência de curso à instituição possui atributos para data de criação e de atividade (função booleana para em atividade ou não).

- Um Curso é uma entidade que possui os atributos identificador (id), nome, sigla (acrônimo), nível (médio, superior ou outro), serie, carga horária, portaria do Diário Oficial da União (D.O.U); um curso possui várias entidades turmas e cada turma está vinculada a uma só entidade curso. A relação de pertinência - possui - contem atributos para a data de criação e de atividade (tipo booleano).

- Uma Turma é uma entidade que contem os atributos: identificador (id) e a sigla da turma (um acrônimo); cada turma possui várias disciplinas e cada disciplina pode ser ofertada em várias turmas, nesta relação de pertinência - possui - tem-se o atributo local (sala local) onde ocorrerá as aulas de cada disciplina.

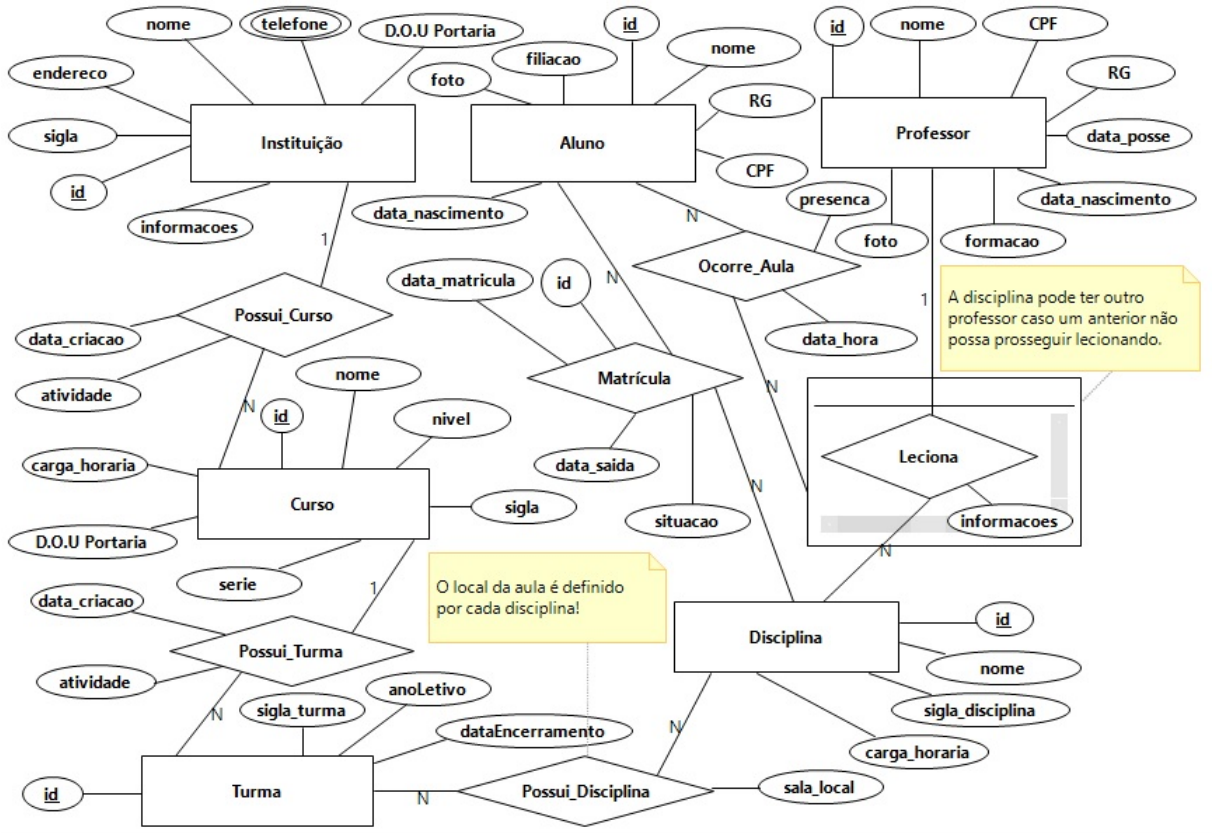
- Uma Disciplina é uma entidade que contem os atributos: identificador (id), nome, sigla da disciplina e a carga horária da disciplina; a disciplina é lecionada por um professor e um professor pode lecionar (Leciona) várias disciplinas. Uma disciplina possui vários alunos matriculados e um aluno pode estar matriculado em várias disciplinas, nesta relação de pertinência Matriculado tem-se os atributos código de matrícula, data da matrícula, data de saída e situação (pode ser registrado por valores pré-definidos para: em curso, transferido, concluído, entre outras situações).

[1] Disponível em `https://github.com/gilchristiano/idblite/tree/master/eercase`. Último acesso em 04 de novembro de 2017.

- Uma entidade Aluno contem os atributos: identificador (id), nome, data de nascimento (data_nascimento), Registro Geral (RG), Certidão de Pessoa Física (CPF), filiação (nome de um dos pais ou responsável) e foto. Cada aluno deve estar em aula para um evento leciona e cada evento leciona ocorre aula (relação ocorre_aula) para vários alunos; cada relação de ocorrência de aula (ocorre_aula) tem-se os registros nos atributos presença e data/hora.

- Uma entidade Professor contem os atributos: identificador (id), nome, CPF, RG, data de nascimento (data_nascimento), data de posse (data_posse), formação e foto. Cada professor pode lecionar várias disciplinas e cada disciplina é lecionada por um único professor. A relação entre a entidade Professor e a Disciplina é do tipo associativa, será necessário ocorrer Leciona antes de se iniciar os registros das aulas.

**Figura A.1:** Modelagem Conceitual em EERCASE




Fonte: Elaborada pelo Autor (2017).

A estrutura relacional DDL foi elaborada a partir da extensão SQLite Manager[2] (versão 0.8.3.1) do Firefox e foi posta no mesmo link da nota de rodapé supracitada.

[2]O SQLite Manager encontra-se disponível no repositório GitHub no endereço `https://github.com/lazierthanthou/sqlite-manager`. Último acesso em 06 de novembro de 2017.