Instituto de Matemática e Estatística
Universidade de São Paulo

Bacharelado em Ciências da Computação

Library Mapper

Thiago Gomes Toledo
Supervisor: Prof. Dr. Marcelo Finger

São Paulo - SP

Segundo semestre de 2011

Agradeço a minha mãe Mara Lucia Gomes Toledo e a meu pai João Batista Toledo por terem me dado a oportunidade de realizar meu sonho; A minha esposa Lucianna Thomaz Almeida Toledo por me dar conforto, ajuda e sábias dicas; A meu filho que me traz alegria só por imaginar seu rosto e a Deus por ter me dado todos e tudo que sempre precisei.
[1]

[1]Para mais informações sobre essa monografia, apresentação, código, etc.. acesse o site: www.linux.ime.usp.br/ renoir/-mac499/proposta.html

Sumário

1 Introdução

Como frequentador da biblioteca do Instituto de Matemática e Estatística da USP desde 2008 e trabalhando na biblioteca do Instituto de Geociencias da USP em 2010-2012, pude notar que as reclamações numa biblioteca sobre a dificuldade em se achar um livro lideram o ranking de problemas apontados por visitantes que usualmente têm que apelar para a boa vontade dos funcionários, pois acham muito complicado ou até sem nenhum sentido o modo como as fileiras de livros são identificadas.

Em ambos os casos os alunos ainda eram capazes de encontrar seus livros ou teses dado que a maior delas possuia não mais que 20.000 exemplares.Contudo, numa biblioteca como a da Faculdade de Filosofia, Letras e Ciências Humanas que hoje possui um inventório de mais de 500 mil livros e ainda 600 mil teses[1], a busca se torna completamente inviável sem nenhum auxílio real.

2 Parte Objetiva

Nessa parte do texto, serão apresentados os conceitos teóricos do trabalho, além da parte prática de desenvolvimento.

2.1 Objetivo

Com uma ferramenta que mostre como ir do ponto onde está o visitante até o livro através de um mapa, esse dificilmente se sentiria perdido ou precisaria contar com o auxílio de terceiros.Portanto, o objetivo desse projeto Library Mapper é que o vistante busque, escolha o livro e receba no seu aparelho mobile um mapa com o caminho correto até esse livro.

2.2 Problema

Como dispositivos GPS não mapeam dentro de residências, é inviável utilizar a Geolocalização do visitante para qualquer fim e portanto era necessário criar algum método para identificar onde ele estava localizado, como era esse lugar e como gerar uma busca entre onde ele estava e onde devia chegar.

2.3 Solução

Inicialmente desenvolvida para dispositivos IOS, essa ferramenta utiliza tags com códigos QR para auxiliar o reconhecimento da localização atual do visitante. O visitante ao entrar numa biblioteca procura essa tag e com seu dispositivo IOS, reconhece o código nela contido.Esse código abre um Web Browser no dispositivo e uma página da biblioteca na internet é mostrada na tela.O visitante digita o livro num campo de busca e uma vez que esse existe nessa biblioteca, a tela do web browser é substituída por uma tela com um mapa que mostra o caminho da tag selecionada até a fileira onde esta o livro.

Para ser possível a localização do livro, será utilizada a prévia catalogação feita pela biblioteca da posição do mesmo e uma réplica proporcional do mapa real da biblioteca .Essa réplica é desenvolvida através de uma plataforma web que gera mapas 2D também desenvolvida nesse projeto.As informações obtidas nesse mapa, localização de estantes e suas correspondentes identificações, Qr-Codes, obstáculos e limites da biblioteca, serão enviados e armazenados num banco de dados da aplicação.

O mapa será gerado num grafo tipo grid e cada nó está dividido em 4 tipos principais que são QR-Code, Forbidden, Free e Bookshelf numa classe Node e será baseado nesses tipos que a busca pelo livro dado será feita.

Como todas as arestas destre grafo tem peso único um, o algoritmo de busca usado é uma Bi-direcional BFS (Breadth-first search) com uma busca saindo da origem(QR-Code selecionado pelo visitante) e outra do

[1] Dados levantados em janeiro 2012 com a diretora da biblioteca, Maria Aparecida Laet

destino (estante onde está o livro).

Essa busca é feita de uma maneira paralela onde cada thread é um dos pólos do caminho Bi-direcional.

2.4 Conceitos e tecnologias estudadas

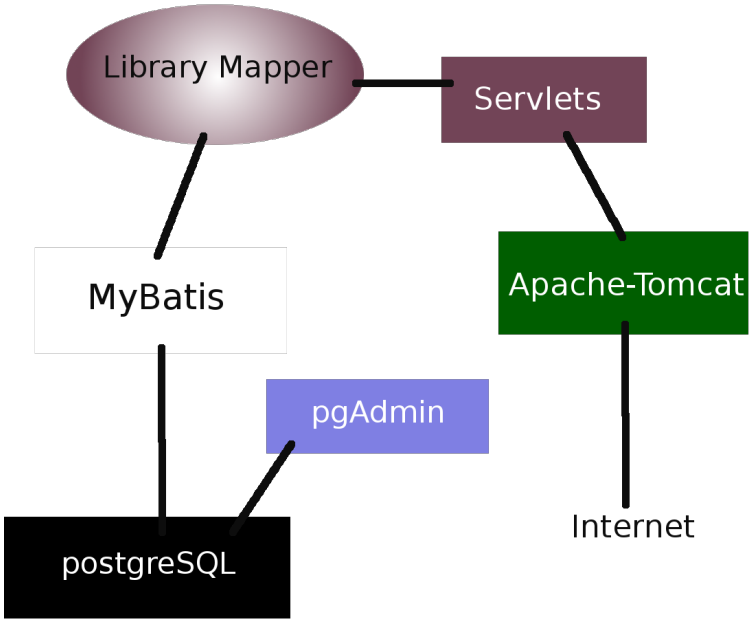


Figura 1: Técnologias usadas

2.4.1 O que são Qr-Codes?

Quick Response Code foi inventado por DENSO WAVE, uma empresa Japonesa de captura automática de dados, FA e campos de solução de negócios, em 1994 e foi um dos primeiros códigos de barra 2D a ser usado em aplicações onde era necessário uma combinação de camera de aparelhos celulares e códigos de barra 2D. [3]Esses códigos trocam barras e espaços por pontos e espaços arranjados num vetor ou uma matrix, aumentando consideravelmente a quantidade de dados armazenados num espaço igualmente usado para um código de barra 1D. Apenas em 2002 cameras de celulares foram comercialmente vendidas com dispositivos reconhecedores de Qr-codes e desde então uma vasta variedade de aplicativos começou a surgir no Japão e vem tomando o mundo.

Em 2006, como parte do Windows Live service, a Microsoft lançou o Windows Live Barcode , onde Qr-codes eram usados como ponte para troca de informações entre aparelhos mobile e outras medias como PCs, Tv de plasma e revistas.

As Qr-Codes geradas para esse projeto passarão para o usuário o endereço de uma página web de busca dos livros da bilioteca em que ele se encontra.Junto nessa URL tem um id do Qr-Code, que mostra para o programa onde no grafo do mapa da biblioteca se encontra o visitante, para dessa maneira poder criar o caminho que o mesmo deve traçar.

2.4.2 Grid Graphs

Um grafo G é uma par *(V,E)*, onde V é um conjunto finito e E é uma relação binária de V. O conjunto V é chamado de o *conjunto de vértices* de G e seus elementos são chamados de *vértices*.O conjunto E é chamado de *conjunto de arestas* de G e seus elementos são chamados de *arestas*.Existem dois tipos de grafos - grafo direcionados e grafos não-direcionados. [2]

Uma aresta num *grafo não-direcionado* é um conjunto *{u,v}* onde $u,v \in V$ e $u \neq v$ que por convenção é definida como (u, v) ou (v, u) e é possível ir de u para v e de v para u.

Já num *grafo direcionado* (ou *dígrafo*) uma aresta é identificada por setas e $(u, v) \neq (v, u)$.

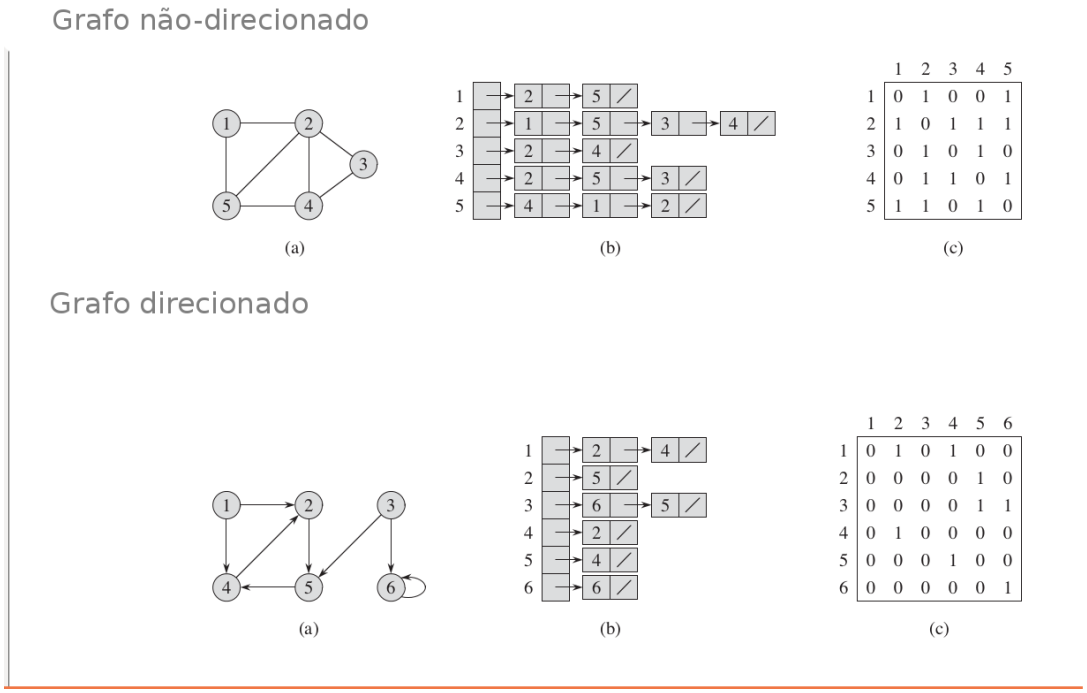


Figura 2: Grafo retirado do livro [2]

Em ambos os casos uma aresta *(u,v)* pode possuir um *peso* que indica o custo por passar do vértice u para o v.

Um Grid Graph (grafo do tipo Grid) é um grafo não-direcionado; uma grade de nós (vértices) n x n sendo que cada nó tem exatamente 4 arestas com exceção dos nós das extremidades.

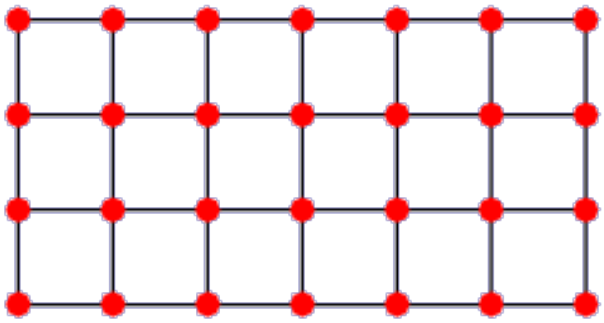


Figura 3: Representação de um Grid

Para o LibraryMapper será usado um grafo não-direcinado do tipo Grid alocado numa matriz, sendo que cada vértice do grafo será um nó da classe Node.Essa Possui os seguintes atributos:

```
1  package library.domain;
2
3  public class Node {
4
5      private Integer idNode;
6      private Integer positionX;
7      private Integer positionY;
8      private Integer contentId;
9      private Integer idLibrary;
10     private String codeIdInitialShelf;
11     private String codeIdFinalShelf;
12     private String contentType;//Free,QrCode,Forbidden,Shelf
13     private Node parentFromBeginNode;
14     private Node parentFromEndNode;
15     public int isInUse;
16     private Semaphore semaphore;
17     private String color;
18     private String whoMarkedThisNode;
19 /*... continua ...*/
```

2.4.3 Breadth-First Search (BFS)

Dado um grafo *G=(V,E)* com um vértice *s* distinto, Breadth-first search explora sistematicamente as arestas de *G* para descobrir todo vértice que pode ser alcançado por *s*.Ele computa a distância (menor número de arestas) do vértice *s* até todos os vértices que ele possa alcançar.Para manter o controle do progresso da busca, BFS procura cores em cada vértice - branco, cinza e preto. Todos os vértices começam com cor branca, eventualmente passam para cinza e por fim ficam pretos. [2]

Uma vez que um vértice é descoberto ele se torna não-branco.Um vértice só se tornará preto quando todos os vértices que são adjacentes á ele tenham sido descobertos, enquanto isso ele será cinza.Isso representa a fronteira entre vértices descobertos e não-descobertos como mostrado na Figura 4

Como no pior caso, a busca tem que ser feita em todos os (m) nós de um grafo e passar por todas suas (a) arestas.Como a BFS não volta nos caminhos que já traçou, a complexidade desse algoritmo é O(a).

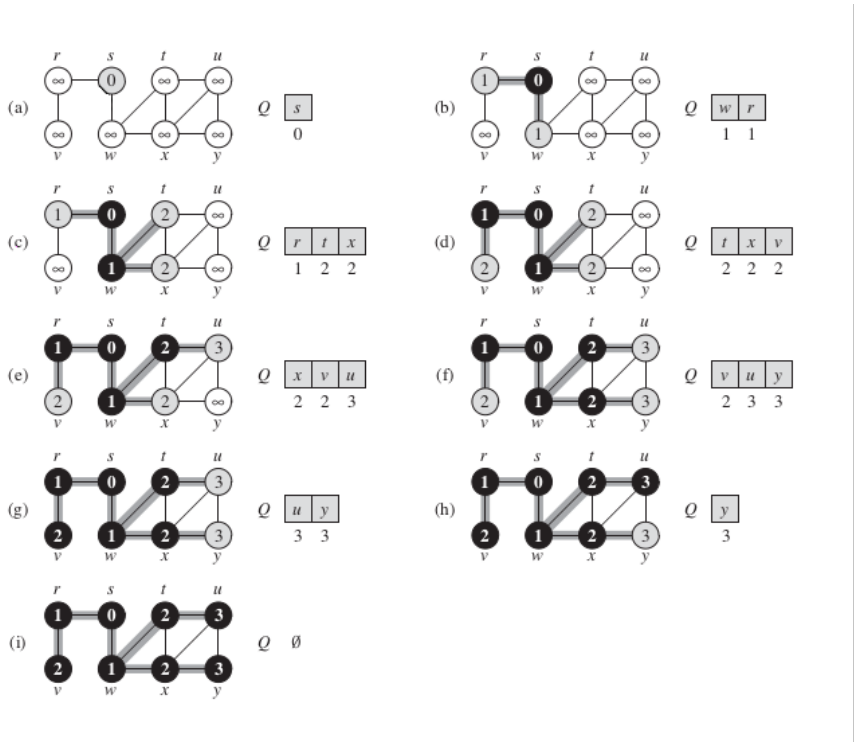


Figura 4: BFS retirada do livro [2]

Além das cores, nesse projeto essa BFS distingue entre nós (vértices) do tipo Free, Bookshelf, Qr-Code e Forbidden e será aplicada de uma maneira Bi-direcional.

2.4.4 Bi-directional BFS

A idéia por trás de uma busca Bi-direcional é fazer duas buscas rodarem simultaneamente - Uma "para frente", saindo do estado inicial (a origem) da busca inteira e outra "para trás", saindo do estado final (a chegada) da busca inteira esperando que ambas se encotrem no meio. [5].

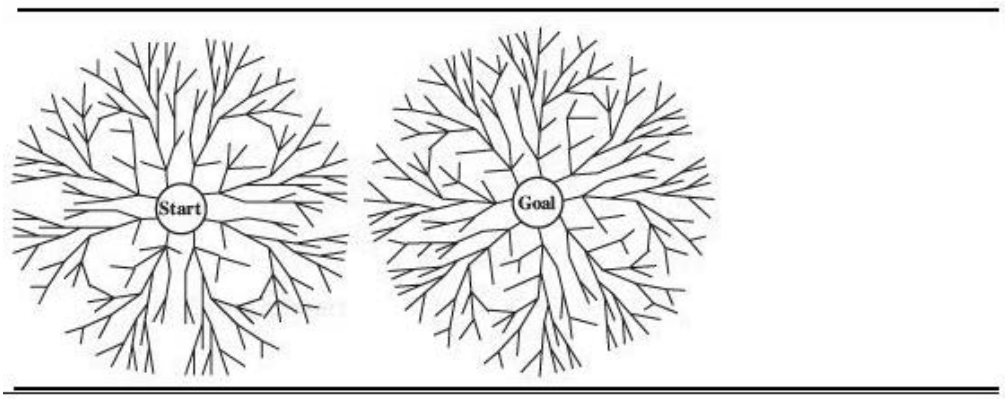


Figura 5: BFS bi-direcional retirada do livro [5]

Nesse projeto a busca Bi-direcional será realizada através de métodos concorrentes de programação.Cada pólo iniciará sua busca paralelamente liberando assim duas threads.Caso uma thread chegue em um nó que já foi visitado ela define aquele nó como o "meio do caminho"e a partir dele gera a rota inteira do mapa.Mas caso o nó que chegue esteja sendo visitado, ela para e abandona o que estiver fazendo para depois ser reciclada num outro novo processo. Nas próximas seções serão demonstradas com maiores detalhes, como essa busca foi implementada no Library Mapper.

Pela profundidade da busca ter caído pela metade, ou seja, no máximo uma busca tem que ir até a metade do grafo, a complexidade do algoritmo é O(a/2).

2.4.5 pgAdmin - Interface gráfica para gerenciamento do banco de dados PostgreSQL

pgAdmin é o recurso de administração Open-Source para o banco de dados PostgreSQL. O aplicativo pode ser usado no Linux, FreeBSD, Solaris, Mac OSX e Windows para gerenciar o PostgreSQL 7.3 e acima executado em qualquer plataforma, bem como versões comerciais e derivados do PostgreSQL como Postgres Plus avançado Server e banco de dados Greenplum. [4]

pgAdmin é projetado para atender as necessidades de todos os usuários, desde escrever consultas SQL simples para o desenvolvimento de bancos de dados complexos. A interface gráfica suporta todos os recursos do PostgreSQL e facilita a administração. O aplicativo também inclui um destaque de sintaxe SQL editor, um editor de código do lado do servidor, um agente de agendamento de tarefas SQL/batch/shell,etc. Conexão com o servidor pode ser feita usando TCP / IP ou soquetes de domínio Unix (nas plataformas * nix), e pode ser criptografado SSL para a segurança. Drivers adicionais não são necessários para se comunicar com o servidor de banco de dados.

Com o pgAdmin não é possível iniciar um servidor de Banco de dados no linux. É necessário um terminal para criar e configurar um servidor de banco de dados e referenciá-lo no pgAdmin.Contudo uma vez isso feito, fica fácil criar banco de dados e também editar, deletar tabelas e colunas desse banco.Como exemplo, a figura 6 mostra como é a tela de administração dos bancos de dados em conjunto de uma das tabelas do banco do Library Mapper.

No Library Mapper foi criado um banco de dados com o nome libraryMapperBD que utiliza a porta padrão do servidor de dados para se comunicar com o programa. Os detalhes desse banco serão vistos logo em seguida.

2.4.6 PostgreSQL

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e MS Windows. É totalmente compatível com ACID, tem suporte completo a chaves estrangeiras, junções (JOINs), visões, gatilhos e procedimentos armazenados (em múltiplas linguagens). Inclui a maior parte dos tipos de dados do ISO SQL:1999, incluindo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, e TIMESTAMP. Suporta também o armazenamento de objetos binários, incluindo figuras, sons ou vídeos. Possui interfaces nativas de programação para C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, entre outros, e uma excepcional documentação.

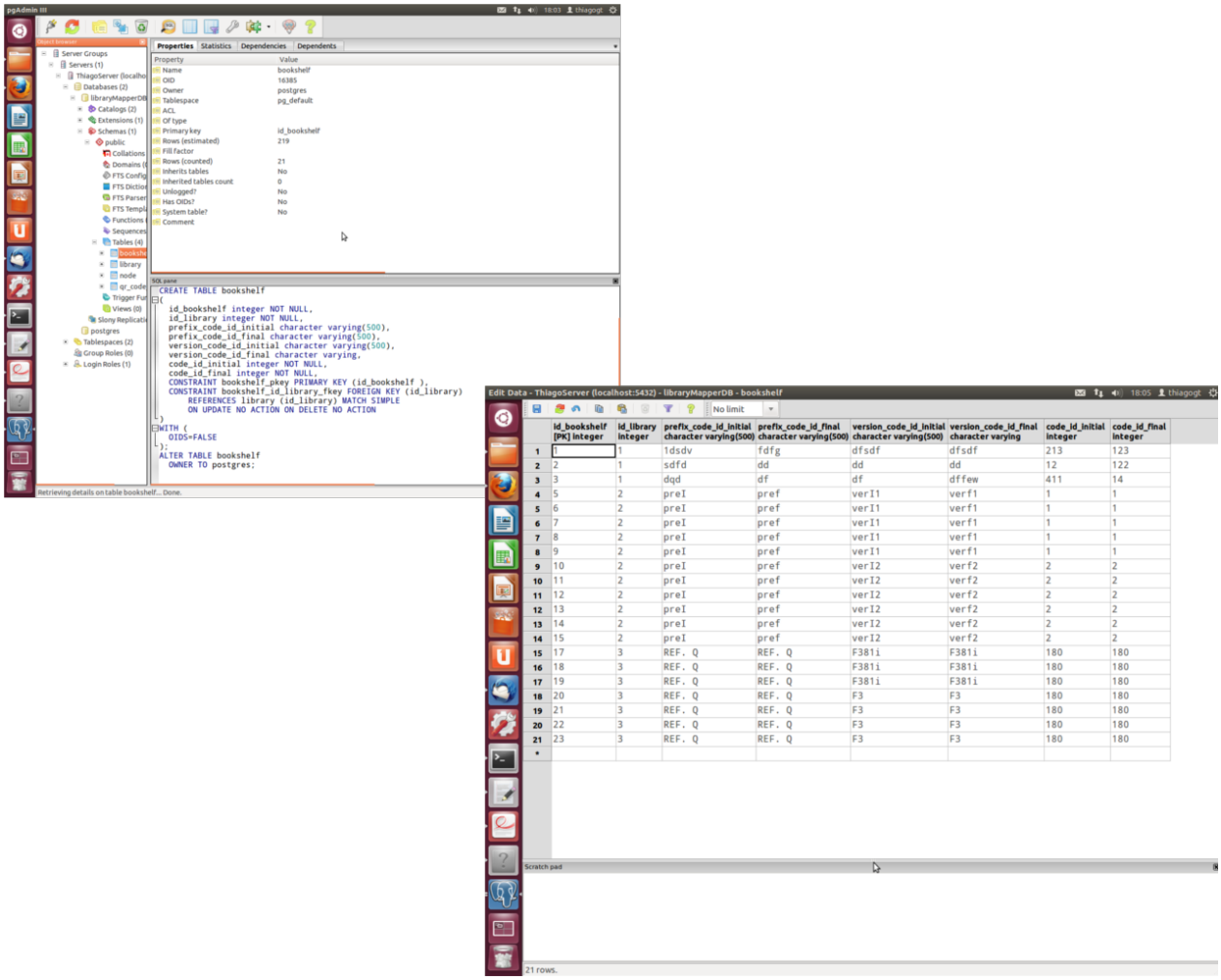


Figura 6: tela de administração pgAdmin e tabela de dados

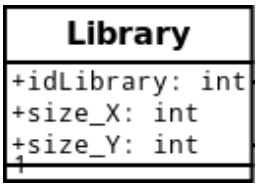
Como um banco de dados de nível corporativo, o PostgreSQL possui funcionalidades sofisticadas como o controle de concorrência multiversionado (MVCC, em inglês), recuperação em um ponto no tempo (PITR em inglês), tablespaces, replicação assíncrona, transações agrupadas (savepoints), cópias de segurança a quente (online/hot backup), um sofisticado planejador de consultas (otimizador) e registrador de transações sequencial (WAL) para tolerância a falhas. Suporta conjuntos de caracteres internacionais, codificação de caracteres multibyte, Unicode e sua ordenação por localização, sensibilidade a caixa (maiúsculas e minúsculas) e formatação. É altamente escalável, tanto na quantidade enorme de dados que pode gerenciar, quanto no número de usuários concorrentes que pode acomodar. Existem sistemas ativos com o PostgreSQL em ambiente de produção que gerenciam mais de 4TB de dados. Alguns limites do PostgreSQL estão incluídos na tabela abaixo. [1]

Tamanho Máximo do Banco de Dados	ilimitado
Tamanho máximo de uma Tabela	32 TB
Tamanho Máximo de uma Linha	1.6 TB
Tamanho Máximo de um Campo	1 GB
Máximo de Linhas por Tabela	Ilimitado
Máximo de Colunas por Tabela	250–1600 dependendo do tipo de coluna
Máximo de Índices por Tabela	Ilimitado

Tabela 1: Dados retirados do site PostgreSQL

O banco de dados usado pelo Library Mapper é criado para armazenar a matriz que se refere ao mapa da biblioteca, representada aqui por Mapa(i,j).
O banco contém 4 tabelas:

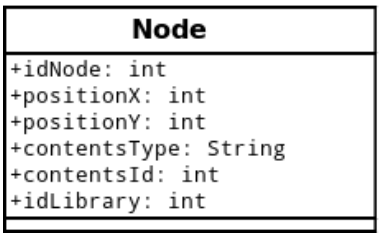
- library



Essa tabela foi criada para o cliente ter liberdade de criar inúmeras versões de bibliotecas e também carregá-las quando queira.

idLibrary: o id de cada nova biblioteca criada.
size_X: número de colunas na matriz Mapa(i,j).
size_Y: número de linhas na matriz Mapa(i,j).

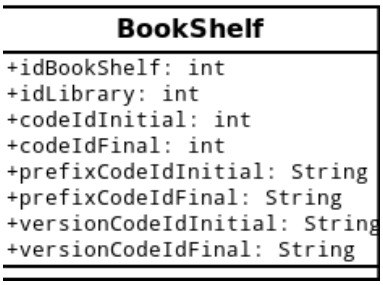
- node



Cada Node equivale a uma posição na matriz Mapa(i,j)

idNode: o id de cada novo Node.
positionX: coluna j na matriz Mapa(i,j).
positionY: linha i na matriz Mapa(i,j).
contentsType: Tipo do Node, podendo variar entre Free, Forbidden, QrCode ou Shelf.
contentsId: Uma vez definido qual o tipo do Node é necessário saber qual o seu id na tabela relacionada.Caso não possua uam tabela, como é o caso de Free e Forbidden, o ContentsId é um valor desnecessário
idLibrary: o id da biblioteca a qual pertence.

- bookshelf



Quando uma bookshelf é criada ela recebe dois identificadores, sendo um referente ao primeiro livro contido nela e o outro o último.Dessa maneira, é possível identificar o range de livros contido nela.Mas para ser possível fazer uma busca no banco de dados nesse range, é necessário retirar dois valores inteiros, sendo um o início e o outro o fim do range.Por isso a tabela bookshelf divide os identificadores em 3 partes: prefixo ,codeId e versão.Dessa maneira, com a identificação do prefixo mais a verificação no range da estante, é possível retornar uma consulta precisa da estante que se encontra o livro.A figura 7 mostra a maneira como uma das duas indentificações é dividida para ser guardada no banco de dados.

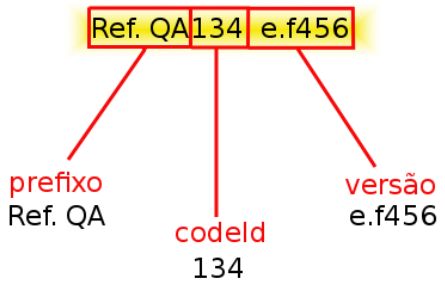
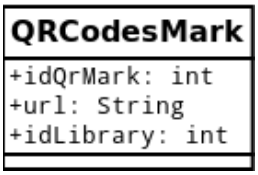


Figura 7: Identificação da estante parseada.

idBookShelf: o id de cada nova estante
idLibrary: o id da biblioteca a qual pertence.
prefixCodeIdInitial: prefixo incial da estante
codeIdInitial:range inicial da estante
versionCodeIdInitial:versão inicial da estante
prefixCodeIdFinal:prefixo final da estante
codeIdFinal:range final da estante
versionCodeIdFinal:versão final da estante

- qr_cod_mark



idQrMark: id de cada novo QrCode
url: a url contida no QrCode.
idLibrary:o id da biblioteca a qual esse QrCode pertence

A imagem abaixo ilustra o banco de dados completo.

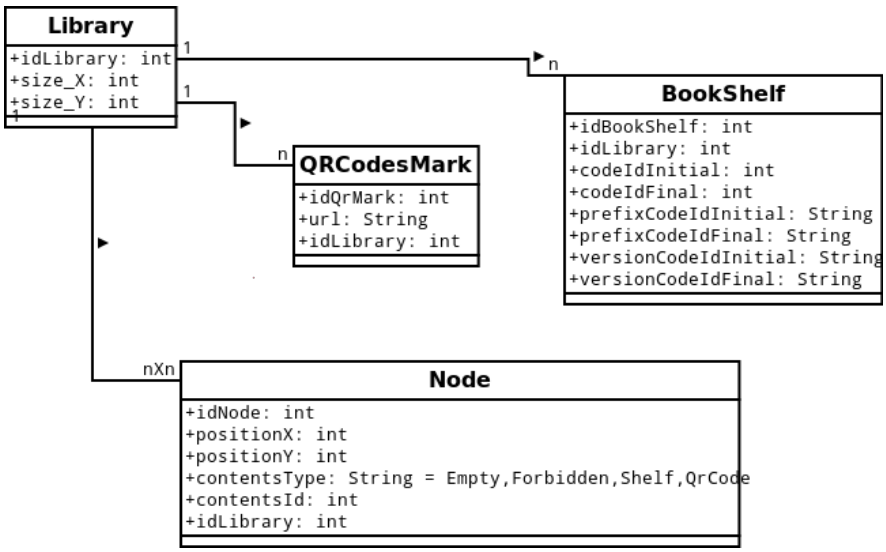


Figura 8: Identificação da estante parseada.

2.4.7 POO - Programação Orientada a Objetos

2.4.8 Mybatis - Integração MER e POO

2.4.9 Servlet - WebService JAVA

2.4.10 Apache Tomcat - Servidor WEB

2.4.11 jQuery e JavaScript

2.4.12 HTML5 - Canvas

2.4.13 Desenvolvimento mobile

2.4.14 Métodos Ágeis - Kanban e Integração Contínua

2.5 Library Mapper, o cliente

Essa seção é reservada para explicar toda a parte relacionada ao cliente da aplicação e, nesse caso, o cliente é o responsável pela biblioteca.O único trabalho desse cliente é montar a biblioteca pela interface Web, identificando cada estante corretamente e onde ficarão os QR-Codes. Nessa primeira versão do Library Mapper, os qr-codes criados no programa são armazenados no banco de dados da aplicação e não há como identificá-los sem ter que olhar no BD pela ordem que foram criados.

Referências

[1] Postgresql Brasil community page. `http://www.postgresql.org.br/sobre`.

[2] H. Thomas Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, 1 edition, 2009.

[3] Hiroko Kato, Keng T. Tan, and Douglas Chai. *Barcodes for Mobile Devices*. Cambridge, 1 edition, 2010.

[4] pgAdmin page. `http://www.pgadmin.org/index.php`.

[5] Stuart Russel and Peter Norvig. *Artificial Inteligence, A Modern Approach*. Prentice Hall, 3 edition, 2009.