Universidade de São Paulo

ICMC

Bacharelado em Ciências de Computação

Disciplina de Organização de Arquivos

Profa. Dra. Cristina Dutra de Aguiar Ciferri

PAE Lucas de Carvalho Scabora (Turma B)

Segundo Trabalho Prático

Carlos Eduardo Ayoub Fialho #7563703
João Gustavo Cabral de Marins #7563982
Romeu Bertho Junior #7151905

15 de Junho de 2015

Índice

1. Descrição dos campos
2. Como Usar
3. Remoçao Logica e Reaproveitamento de Espaço
4. Arquivos de Índice
5. Telas da Interface
6. Bateria de testes

Como usar

Para compilar o programa: `$make`
Para rodar: `$./piupiu.app`
Usar o arquivo de teste: `$./piupiu.app < in.txt`

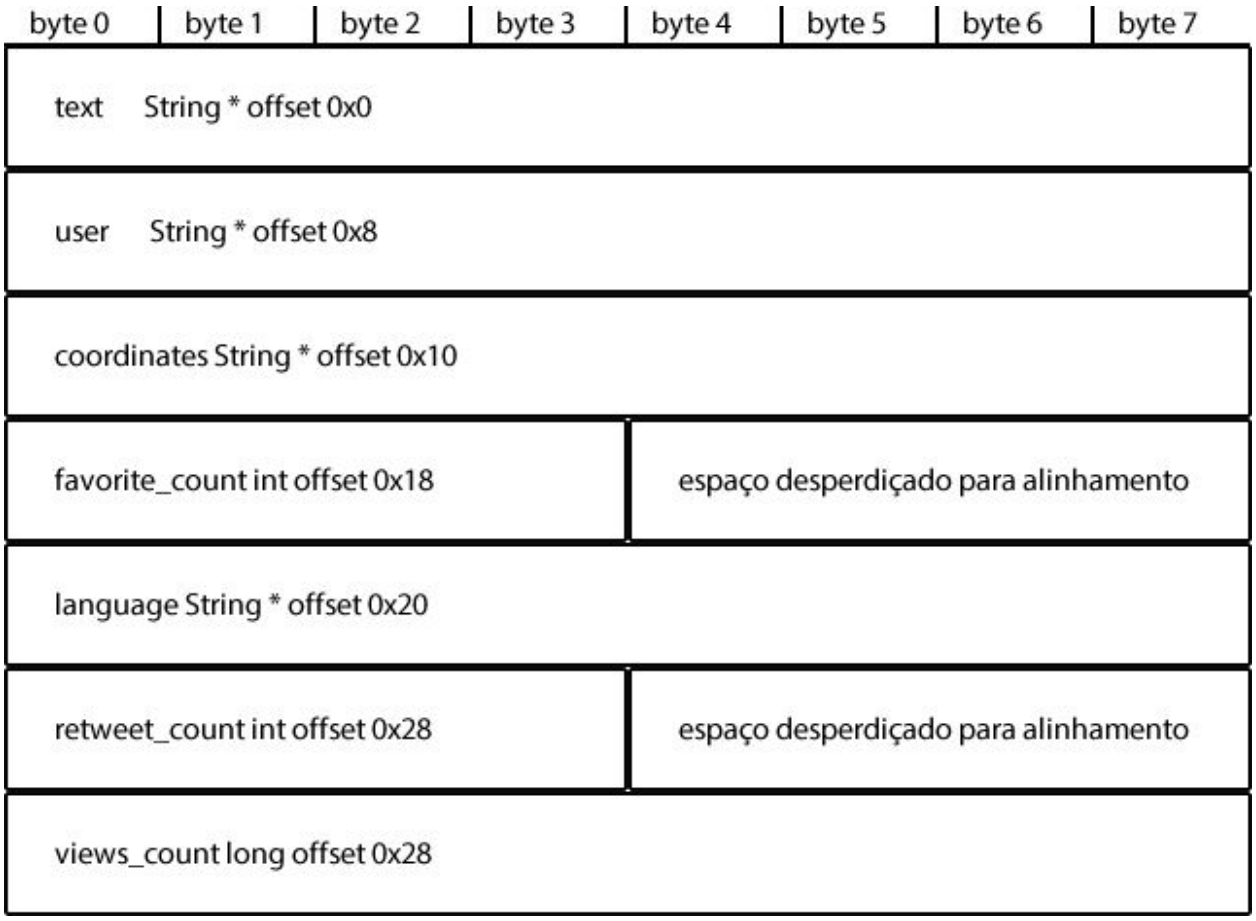
Dentro do programa, para navegar, utilize as opções do menu

Descrição dos campos

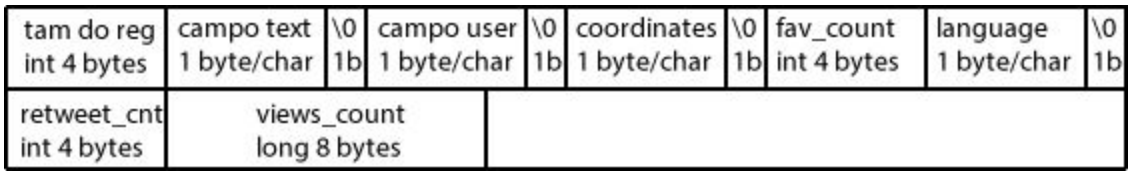
1. `text`: tipo string, tamanho variável, guarda o texto do tweet
2. `user`: tipo string, tamanho variável, guarda o nome do usuário que postou o tweet
3. `coordinates`: tipo string, tamanho variável, guarda as coordenadas que o usuário postou o tweet
4. `favorite_count`: tipo inteiro, tamaho do inteiro (geralmente 4 bytes), guarda a quantidade de marcações como favorito
5. `language`, tipo string, tamanho variável, guarda o idioma do tweet
6. `retweet_count`: tipo inteiro, tamanho do inteiro (geralmente 4 bytes), guarda a quantidade de vezes que o tweet foi retweetado
7. `views_count`: tipo long, tamanho do long (geralmente 8 bytes), guarda a quatidade de vizualizações do tweet

Como delimitadores nos campos de tamanho variável foi utilizado o caractere '\0'.
A principal vantagem deste caractere é que ele não é um caractere imprimível.

Representação da struct `tweet` na memória em uma maquina de 64 bits:



Representação do registro no arquivo:



Importante notar que o campo de tamanho do registro armazena o tamanho sem considerar si mesmo. Por exemplo, se cada campo de string tiver apenas 1 caractere (sem considerar o '\0') o campo de tamanho do registro irá armazenar o valor 24. 2 bytes pra cada string(1 caractere + \0) + 2 * 4 (tamanho dos int) + 1 * 8 (tamanho do long).

O campo de tamanho do registro foi escolhido como int pois dificilmente um registro de tweet passará de `2 ^ 31-1 bytes`, visto que o servico twitter.com limita em 140 caracteres cada tweet.

Outra informação importante é que o offset de um registro é o offset do campo text, o offset do campo de tamanho do registro fica 4 bytes antes.

Remoçao Lógica e Reaproveitamento de Espaço

Reaproveitamento de espaço

O formato de arquivo é preparado para o reaproveitamento de espaço da seguinte maneira. Os primeiros 8 bytes representam o topo da lista de registros removidos (long int) Foi escolhido long int pois é o tipo de dado utilizado pela biblioteca stdio.h para tratar de offsets em arquivos. E o tamanho de um arquivo de tweets pode passar de `2 ^ 31-1 bytes`. Quando a lista está vazia este campo guarda o valor `-1`.

Assim que um registro é removido, este campo guarda o offset deste registro.

Remoção lógica

Para marcar um registro como removido, basta tornar o campo de tamanho do registro seu oposto. Exemplo, se o tamanho era 24, passa a ser -24. Desta maneira não se perde a informação do tamanho do registro e não precisa utilizar um valor predeterminado, impossibilitando o usuário de inseri-lo.

Um registro removido fica da seguinte maneira
campo tamanho do registro registro: `= -tamanho do registro`
em seguida os proximos 8 bytes armazenam o offset do próximo registro na lista de remoções. O resto do registro fica inalterado e é considerado lixo.

Estratégia de reaproveitamento

Foi implementada a estratégia *best-fit* como requisitado pela especificação do projeto. O algorítmo foi implementado da seguinte maneira:

1. Percorre a lista de remoções
2. Se encontra um espaço com tamanho igual ao registro a ser inserido o loop é interrompido e espaço é usado para o novo registro
3. Se não encontra, pega o espaço que sobrará menos espaço

Com uma restrição no passo 3, um espaço só é elegível para reaproveitamento se seu espaço for suficiente para guardar o novo registro em qustão e ainda sobrar espaço para pelo menos um registro de tamanho minimo (todos os campos string com apenas 1 caractere).

Esta restrição foi implementada para evitar espaços no arquivo que não caibam nem um registro mínimo.
O espaço que sobrar é colocado na lista de remoção e está apto a receber um novo registro.

Arquivos de Índice

O projeto conta só com índices secundarios fortemente ligados. A repetição de chaves é tratada com um arquivo de lista invertida.

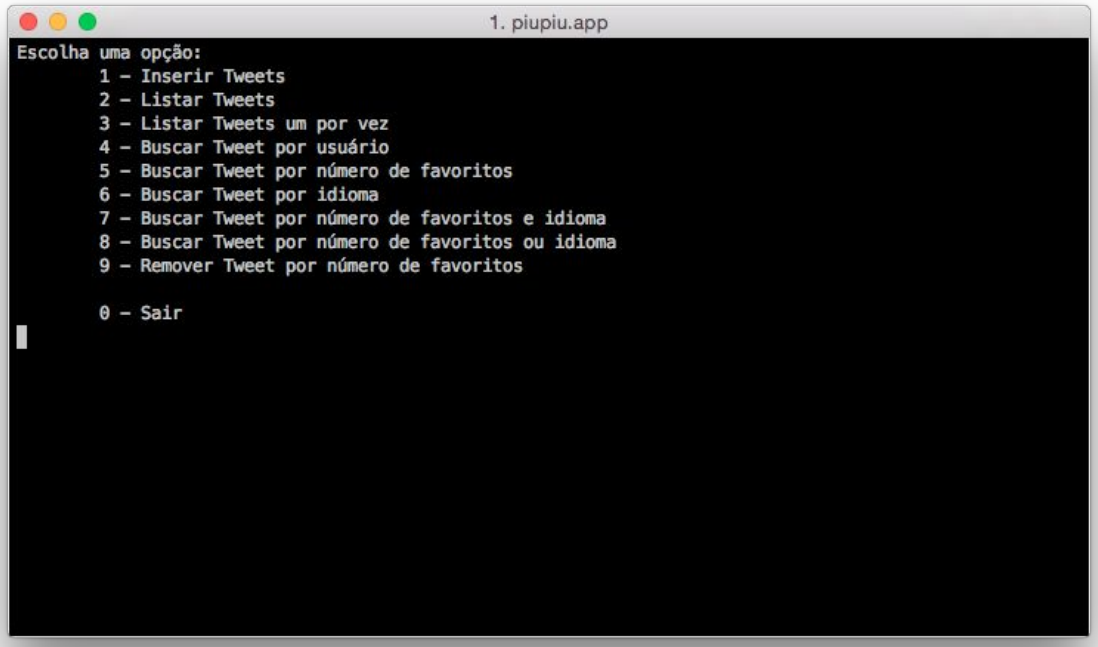
Os arquivos `.idx` guardam as chaves seguidas dos offsets na primeira occorência na lista invertida (arquivo `.lst`).
Os arquivos `.lst` guardam o offset do registro seguido do offset da proxima entrada na lista ou `-1` caso seja a última entrada da lista.

Quando o programa é iniciado, ele verifica se existem pelo menos 10 registros.
Caso exista, verifica se existe um arquivo de indice e lê desse arquivo. Em seguida o apaga, para manter a consistência. Caso o programa seja encerrado corretamente, ele salva o indice, que está em memoria primaria, em memoria secundaria.
Caso o arquivo não exista, ele cria a partir do arquivo de dados.

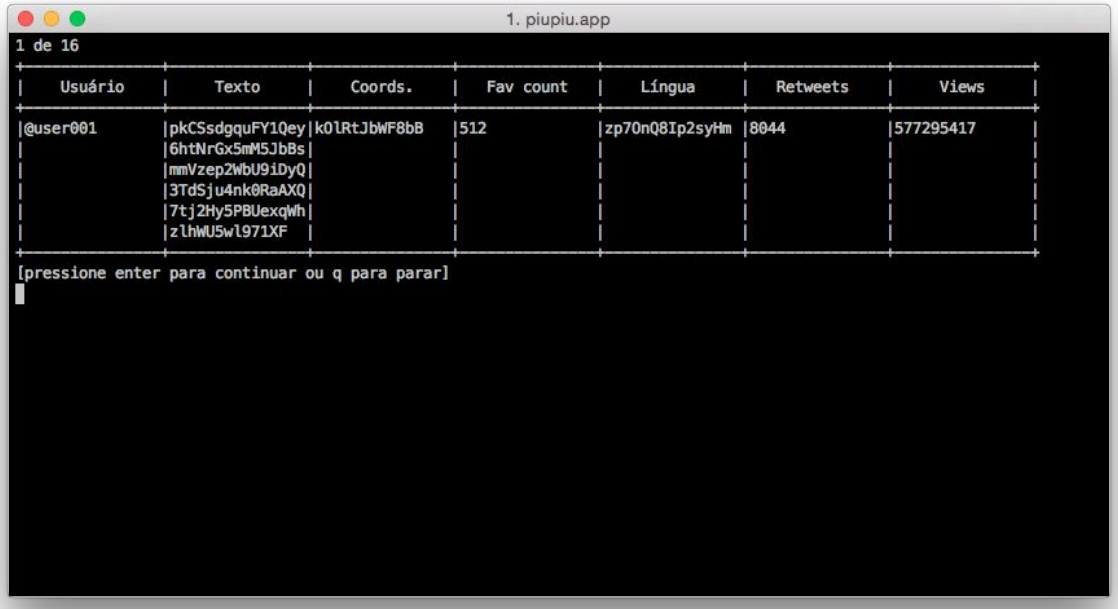
Em memória primaira o indice é armazenado em uma lista ordenada.

Telas da Interface

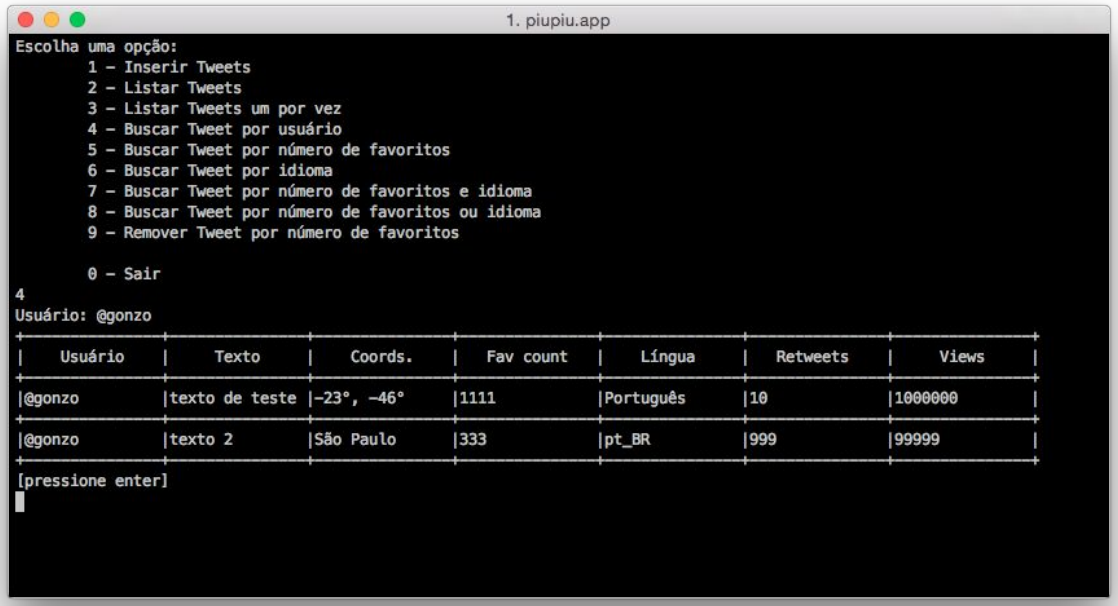
Menu principal:



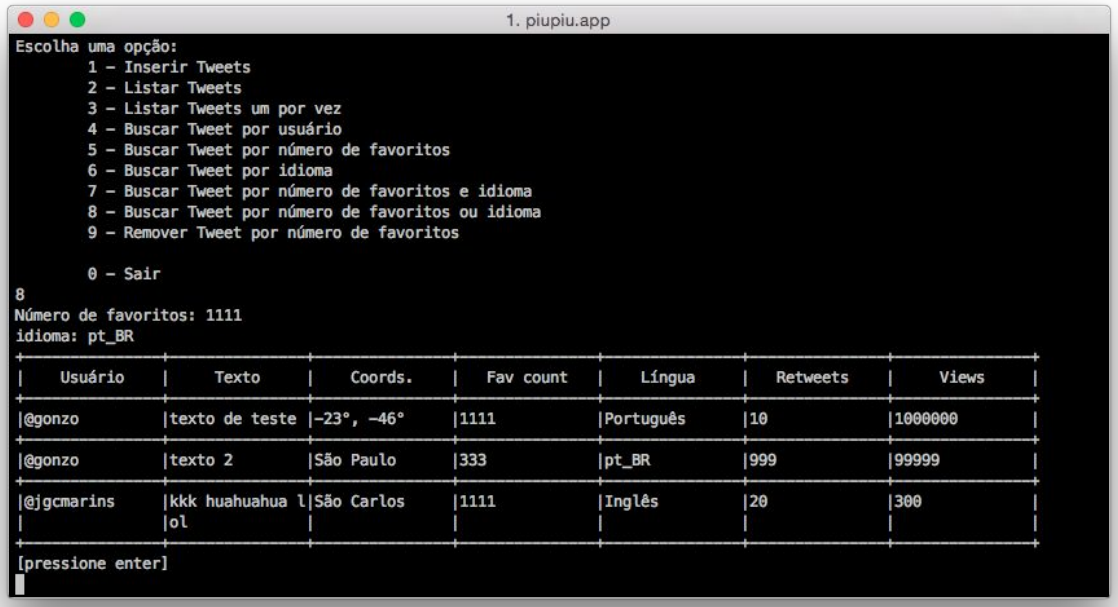
Lista de tweets um por um:



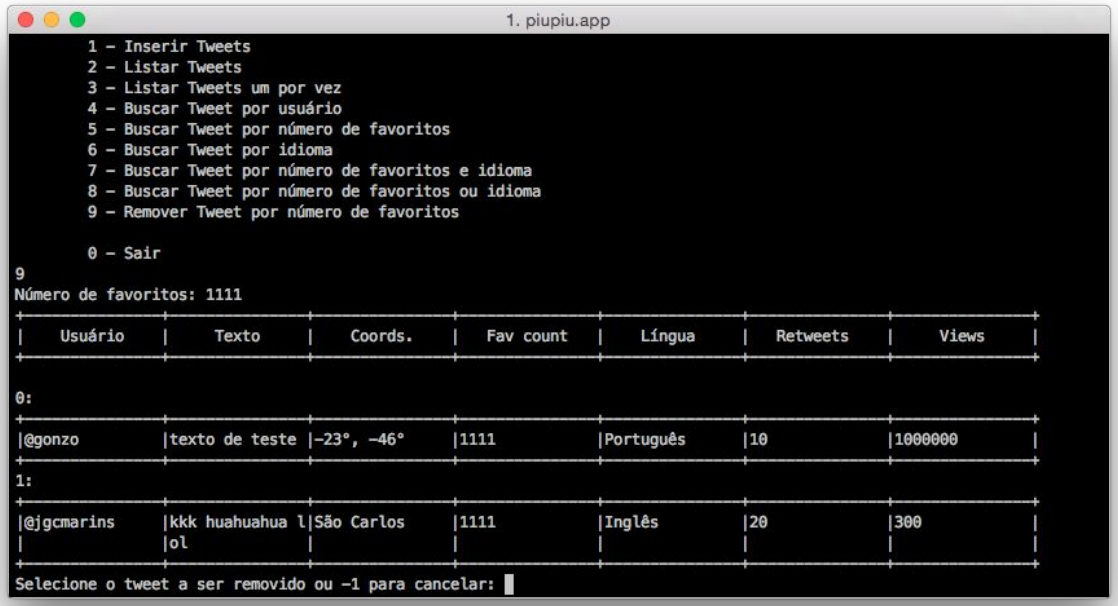
Busca por usuário:



Busca por favoritos e idioma:



Remoção:



Bateria de Testes

Junto com os arquivos encontra-se um arquivo de teste (`in.txt`).
Este arquivo testa todas as funcionalidades do programa.