Aplicação do Algoritmo de Dijkstra para Otimização de Rotas em Percursos Urbanos utilizando Transporte Público

Nícolas Gordiano Barbosa de Sousa, Ana Cláudia Rossi

41442075@mackenzista.com.br, anaclaudia.rossi@mackenzie.br

Faculdade de Computação e Informática – Universidade Presbiteriana Mackenzie
Rua da Consolação, 930 – 01.302-907 – São Paulo – SP – Brasil

Abstract. *Rail transport in the Great São Paulo Area, despite its issues, proves to be faster and more efficient than the individual transport, transporting 7.8 million people every day between the state capital and neighboring cities. However, an information that's not much provided by the rail companies is the travel time between its stations and which routes requires less transfers between lines. This article presents an architecture and a mobile application prototype that calculates the fastest path between the origin and destination stations of the São Paulo rail system.*

Resumo. *O transporte ferroviário da Região Metropolitana de São Paulo, mesmo com suas falhas, mostra-se mais rápido e eficiente que o transporte individual, transportando 7,8 milhões de pessoas por dia entre a capital paulista e cidades vizinhas. Entretanto, uma das informações pouco fornecidas pelas companhias ferroviárias é o tempo de viagem entre suas estações e qual o caminho que requer menos transferências entre linhas. Neste artigo é apresentada uma arquitetura e um protótipo de aplicativo mobile que calcula o caminho mais rápido entre as estações de origem e destino do sistema ferroviário de São Paulo.*

1. Introdução

A computação, cada vez mais presente na vida das pessoas, vem, aos poucos, solucionando um problema comum no cotidiano: a mobilidade urbana. Entre as contribuições da computação para resolver, ainda que aos poucos, problemas com trânsito cada vez mais congestionado e transporte público cada vez mais lotado, já encontram-se diversos aplicativos voltados para esta área; alguns chamam um motorista para levar outra pessoa para onde ela desejar, outros indicam onde está o ônibus mais próximo e quanto tempo ele levará para chegar até o ponto, outros calculam a melhor rota até o destino.

Existem muitos problemas relacionados à mobilidade urbana. É fato que a falta de conforto e a demora do transporte público faz com que a maioria das pessoas utilize o transporte individual. Porém, quanto mais pessoas optam por tirar seus carros das garagens, o acúmulo dos mesmos nas vias das cidades causa um outro problema: o congestionamento, que, se sua intensidade for muito alta, torna-se uma das principais causas de atraso. Enquanto isso, um dos modais de transporte público, o transporte ferroviário, mostra-se muito mais rápido do que o transporte rodoviário, não apenas pelo congestionamento ser algo quase inexistente neste modal, como também por sua velocidade, superior a dos veículos para asfalto, ajudando seus usuários a chegar mais longe em menos tempo.

No entanto, uma informação pouco fornecida pelas companhias ferroviárias é o tempo de viagem entre suas estações, e qual o caminho que requer menos transferências entre suas linhas, mostrando que viagens feitas por muitos de seus passageiros podem ser facilitadas utilizando outras rotas. Também falta a informação de quais estações são as mais próximas de pontos de interesse como parques, centros de comércio, museus, entre outros.

O objetivo deste trabalho é desenvolver uma arquitetura e um protótipo de aplicativo que buscará o trajeto de menor tempo entre estações de metrô, com origem e destino definidos por um usuário.

Este documento está dividido em cinco seções. A seção um apresenta a introdução do documento. A seção dois trata a metodologia usada neste projeto. A seção três apresenta o referencial teórico usado no projeto. A seção quatro mostra os resultados obtidos até o fechamento deste artigo. A seção final trata das considerações finais sobre este projeto e sobre futuros trabalhos voltados para a mobilidade urbana.

2. Método de Pesquisa

A metodologia se baseia em pesquisa em artigos acadêmicos sobre o desenvolvimento de aplicativos voltados para a mobilidade urbana, bem como sobre as ferramentas definidas para a resolução do problema.

Primeiramente foram estudados artigos com propostas de aplicativos sobre mobilidade e seu desenvolvimento, a fim de definir o tipo do aplicativo a ser desenvolvido no projeto. Confirmado o projeto, foi estudado o modal de transporte a ser tratado pelo projeto proposto, para cálculos usados no próprio. Feito isso, foram estudados recursos de programação para definir as técnicas a serem utilizadas no desenvolvimento do projeto, sendo eles a Teoria dos Grafos e Estrutura de Dados.

Com base nesses estudos, a proposta foi baseada no Algoritmo de Dijkstra, método utilizado pela função principal do algoritmo e em Tabela Hash.

3. Referencial Teórico

3.1 Teoria dos Grafos

Um grafo é um conjunto de objetos, chamados vértices que podem estar relacionados entre si através de arestas, embora também existam grafos sem arestas, chamados grafos triviais. Um grafo pode ou não ser orientado, ou seja, ter suas arestas direcionadas para um determinado sentido para arestas específicas. No protótipo proposto, serão usados grafos bidirecionais, ou seja, cujas arestas podem ser percorridas para qualquer lado.

No artigo (NELSON et al, 2018), seus autores buscavam otimizar o transporte de alunos entre os três *campi* da Universidade de Nebraska, em Omaha, Estados Unidos. Utilizando a Teoria dos grafos, eles criaram uma proposta de trajeto que reduzisse o tempo de viagem. O algoritmo proposto

> […] É baseado na implementação padrão do famoso algoritmo de caminho mais curto de Dijkstra. Este algoritmo encontra o caminho

> mais curto, menor custo e menor esforço através de um conjunto de nós dados em um grafo. (NELSON et al, 2018, p.3)

O algoritmo de Dijkstra, base do trabalho de (NELSON et al, 2018), é explicado mais detalhadamente por (BARROS et al., 2007). Segundo estes, deve-se ter um grafo cujas arestas estão marcadas com os valores que representam o custo entre um vértice e outro. Os vértices devem ser identificados com valores inteiros para posterior uso na matriz de adjacência. Feito isso, são definidos os vértices de origem e destino e são criados dois vetores, um com os vértices marcados ao serem verificados pelo algoritmo e outro com os vértices ainda desmarcados. Depois, é criado um vetor para receber os custos entre os vértices já marcados, e outro para receber os vértices recém-verificados.

O algoritmo de Dijkstra foi escolhido para este projeto por tratar do problema do menor caminho entre dois pontos, intuito principal da proposta apresentada.

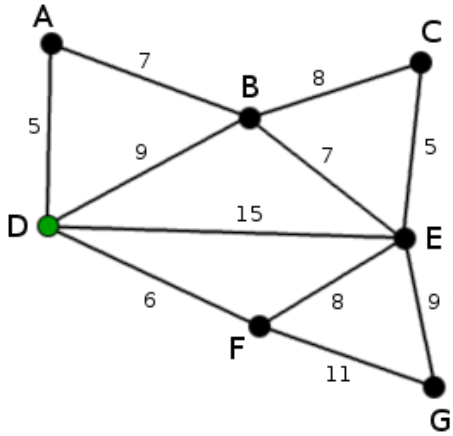

Figura 1. Exemplo de grafo com arestas marcadas com números, que representam o chamado custo de tempo entre seus vértices, elemento central do algoritmo de Dijkstra.

Com tudo preparado, o algoritmo começa a cumprir sua principal função, iniciando uma rotina de verificação vértice a vértice a partir do vértice de origem. Essa rotina

> […] Consiste basicamente dos seguintes passos: a) Verificar as conexões do vértice da vez (ui) que ainda não foram usadas (desmarcadas); b) calcular as distâncias acumuladas das conexões disponíveis; c) identificar entre as disponíveis qual apresenta a menor distância acumulada e marcá-la; d) repetir a rotina até o vértice da vez ser o vértice de destino. (BARROS et al, 2007, p.2)

O algoritmo, segundo (DE CARVALHO, sem data), possui complexidade $O(n^2)$, sendo "n" o número de vértices do grafo.

3.2 Tabela *Hash*

Além disso, o aplicativo contará também com a utilização do modelo chave-valor, ou tabelas *Hash*, para manipulação de seus dados pela função principal. O modelo chave-valor, segundo (BUNGAMA, 2016),

> […] É o mais simples de todos os modelos. Ele manipula os pares chave-valor, ou tabelas *Hash*, cujo acesso é feito exclusivamente pela chave. Trata-se de mecanismo na memória como Redis [11] ou sistemas distribuídos inspirado como o Dynamo [25] e o Riak [33]. Estes sistemas oferecem recursos simplificados, muitas vezes menos riqueza funcional, em termos de consulta por exemplo, todavia excelente desempenho graças ao seu modelo de acesso simples. Um sistema de pares de chave-valor é relativamente fácil de se implementar: o padrão de acesso pela chave não requer nenhum mecanismo de consulta complexa, e este ponto de acesso único permite melhorar o desempenho. (BUNGAMA, 2016, p.33)

Em outras palavras,Tabela *Hash,* é um tipo de estrutura de dados que consiste em um vetor cujos elementos estão armazenados de forma não ordenada, através da função Hash, que espalha os elementos pelo vetor atráves de um cálculo baseado nas chaves a eles relacionadas.

A vantagem da tabela *Hash* é que ela permite um acesso mais rápido a seus elementos, através da busca pela chave em que o valor está armazenado. Entretanto, pode ocorrer colisão, quando a função *hash* duas chaves diferentes são alocadas a um mesmo espaço de memória da tabela.

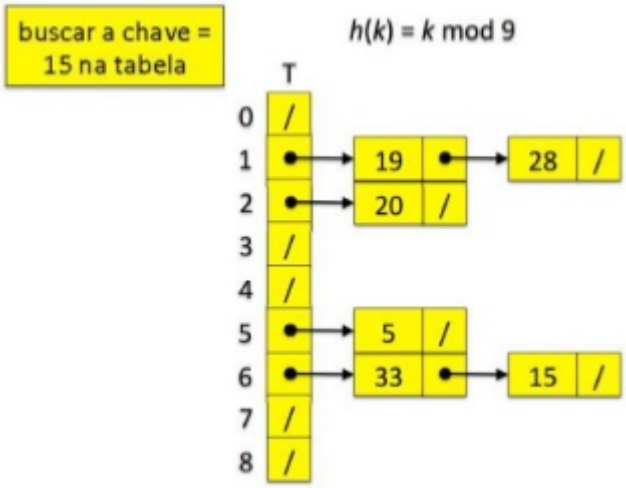

Figura 2. Exemplo de tabela *Hash.*

3.3 Arquitetura de Software de Aplicações Móveis

No artigo (AZLIANOR et al, 2008), seus autores propõem uma arquitetura de software voltado para o sistema ferroviário leve de Kuala Lumpur, capital da Malásia. A arquitetura proposta pelos autores se baseia no modelo cliente-servidor, dividido em três componentes principais: servidor de aplicação *Web*, servidor de banco de dados e cliente *mobile.*

> Este servidor consiste em serviços da web e de aplicativos pelo qual o PHP está configurado para executar processos de negócios e transações, bem como comunicação de dados ao acessar o servidor de banco de dados. Ele responde com as informações solicitadas pelo cliente através de dispositivos móveis. (AZLIANOR et al, 2008)

O servidor de aplicação é responsável pela execução da aplicação baseado nas informações fornecidas pelo usuário, realizando as transações necessárias para cumprir a solicitação do cliente. No caso da proposta de (AZLIANOR et al, 2008), ele possui 4 componentes: o componente de Controle de Processos, que processa e organiza as entradas fornecidas pelo cliente, o componente de Gerenciamento de Dados, que acessa o servidor de banco de dados, o componente de Otimização de Rota, para, através dos dados fornecidos pelo Gerenciamento de Dados, criar a melhor rota entre origem e destino fornecidos pelo cliente, e a Geração de Mapa para retornar ao cliente, de forma visual, a rota calculada entre a origem e destino fornecidos.

O servidor de banco de dados armazena os dados usados na aplicação, no caso de (AZLIANOR et al, 2008), os dados correspondentes às linhas do sistema ferroviário de Kuala Lumpur, e suas respectivas estações. Os dados nele armazenados podem ser acessados pelo servidor de aplicação Web para posterior manipulação.

O cliente, por sua vez, conta com uma interface de usuário e uma aplicação Java para solicitar uma busca por informações armazenadas no banco de dados, que são acessadas e retornadas ao cliente pelo servidor de aplicação.

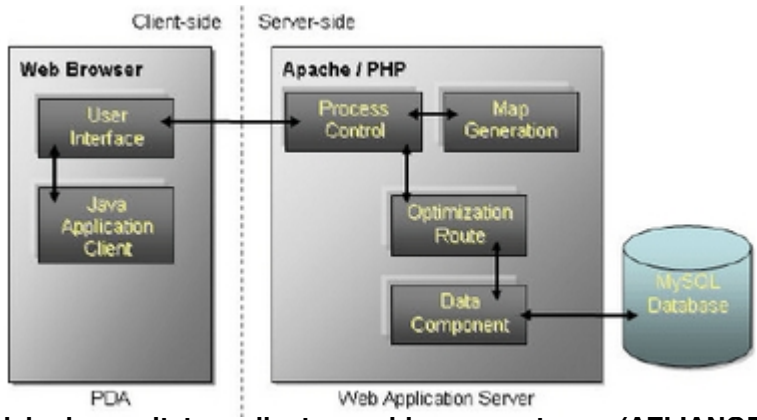

Figura 3. Modelo de arquitetura cliente-servidor proposto por (AZLIANOR et al, 2008).

4. Resultados

4.1. Cenário de Aplicação

Baseado nos estudos supracitados, como parte da elaboração do projeto, foi criado um grafo, que está na figura 2, representando o sistema ferroviário da Grande São Paulo, com todas as linhas de metrô e de trem representadas em subgrafos. As estações estão representadas nos nós do grafo, podendo ser “atendidas” por um ou mais subgrafos. As arestas dos grafos representam os trechos entre as estações e estão marcadas com o custo de tempo, em minutos, da viagem entre as estações, o principal elemento usado pelo algoritmo de Dijkstra, utilizado no projeto. O grafo também serve de base para a Tabela *Hash*, já que o projeto usará também o modelo chave-valor, sendo nela em que estarão armazenadas as estações de cada linha. No grafo, os nomes dos estados, que na verdade são números, são as chaves, armazenadas na tabela *Hash*, e que serão usadas pelo algoritmo principal para calcular as rotas. Os nomes das estações estão representados nos quadros abaixo de cada nó do grafo, e são os valores das chaves correspondentes.

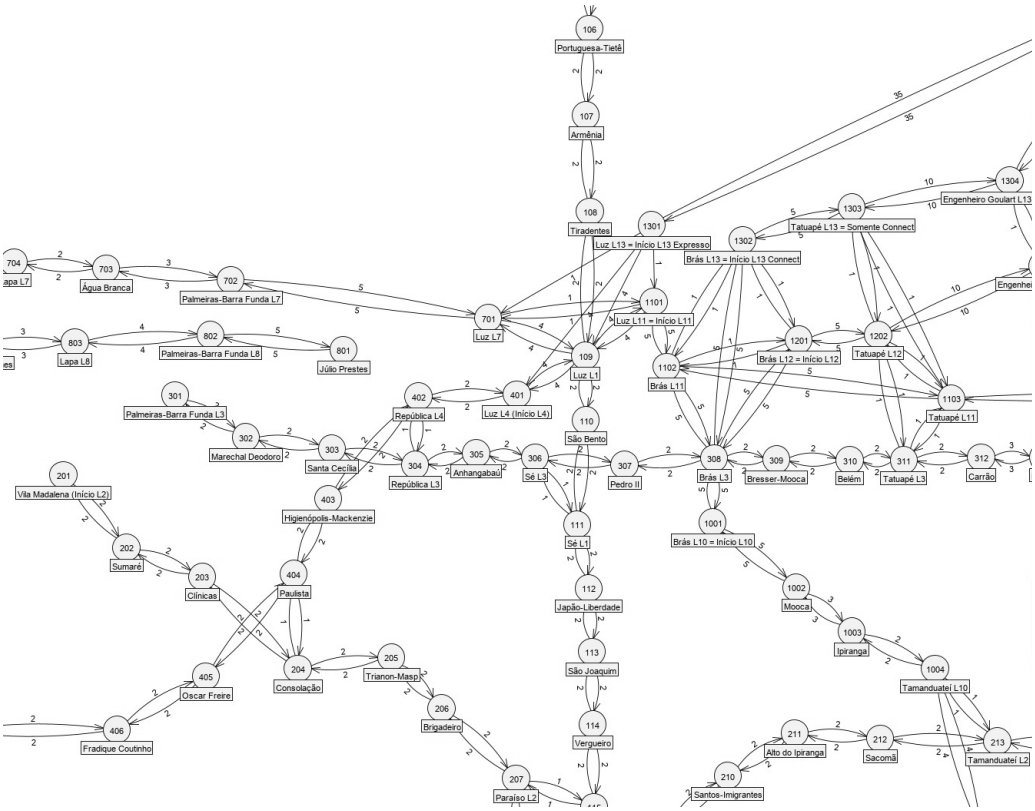

Figura 4. Parte do grafo representando o mapa das linhas de metrô e trem de São Paulo, base para o aplicativo proposto.

Baseado no grafo representado na figura 4, encontra-se em implementação o protótipo do aplicativo, batizado de “Caminhos dos Trilhos SP”. O protótipo está sendo desenvolvido na linguagem Java.

No código do aplicativo, cada linha de metrô e trem, está cadastrada em forma de Tabela Hash, utilizando o modelo chave-valor, sendo chaves do tipo Inteiro e valores do tipo *String*, que são os nomes das estações. Baseando-se no grafo, as chaves seguem uma numeração padrão, sendo o primeiro dígito (ou os dois primeiros, caso tenha quatro dígitos) o número da linha, e os dois últimos, o número sequencial da estação naquela linha (exemplo: a estação Higienópolis-Mackenzie possui a chave 403 no grafo e no código, por ser a terceira estação da linha 4 – Amarela). Feito isso, seguindo a lógica de (BARROS et al., 2007), ainda baseando-se no grafo da rede, é criada uma tabela de adjacência da rede completa, incluindo transferência das linhas no caso de chaves diferentes que possuem o mesmo valor, como a estação Sé, que possui as chaves 111 (décima primeira estação da linha 1 – Azul) e 306 (sexta estação da Linha 3 – Vermelha). As transferências entre linhas também possuem seu próprio custo de tempo representado no grafo, assim como os trechos entre estações de uma mesma linha.

Após o usuário definir a origem e o destino, o aplicativo percorrerá a tabela Hash de cada linha até encontrar a chave com o valor correspondente à origem, mudando, então, para a tabela de adjacência, onde fará esse mesmo processo. Depois, percorrerá as adjacências até encontrar a chave com o valor correspondente ao destino. Com tudo feito, retornará ao usuário as instruções de como ligar origem e destino, mostrando sempre os valores, nunca as chaves.

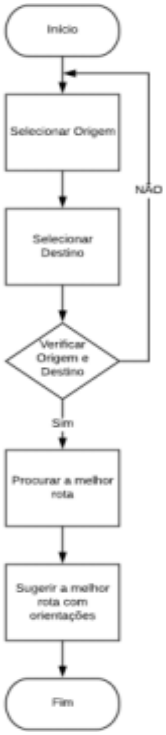

Figura 5. Fluxograma de funcionamento do "Caminhos dos Trilhos SP".

4.2. Arquitetura do Aplicativo

O aplicativo usará a arquitetura de cliente-servidor, similar a proposta por (AZLIANOR et al, 2008).

O lado do Cliente é composto por uma interface de usuário e por uma aplicação Java (Java Application Client), através dos quais o usuário irá selecionar as estações de origem e destino para que a rota seja calculada.

O lado do Servidor é composto pelos seguintes componentes: o Controlador de Processos, responsável por processar e organizar as entradas fornecidas pelo usuário, no caso, as estações de origem e destino; o Gerenciamento de Dados, para acesso aos dados armazenados no servidor, e a Otimização de Rota, que calcula a melhor rota entre as estações escolhidas pelo usuário.

Diferentemente da proposta de (AZLIANOR et al, 2008), que utiliza conexão de internet, visando a não dependência de conexão de *internet* para o funcionamento da função principal, as tabelas *Hash* respresentando as linhas e contendo suas respectivas estações estão armazenadas no servidor de aplicação.

A figura 6 mostra a arquitetura cliente-servidor usada pelo aplicativo e seus respectivos componentes. A figura 7 mostra um diagrama de Casos de Uso do aplicativo proposto.

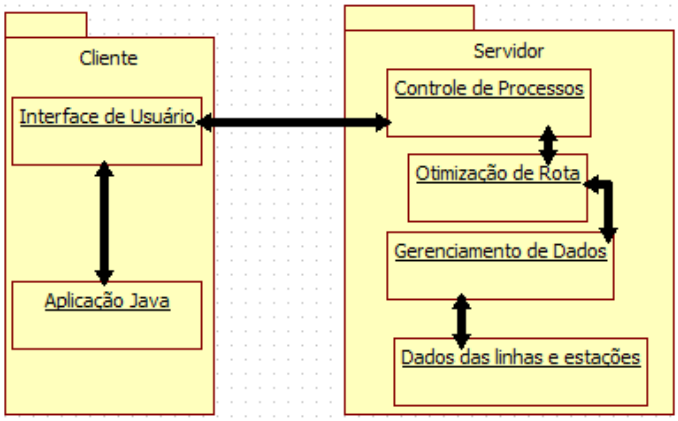

Figura 6. Diagrama de arquitetura do "Caminhos dos Trilhos SP".

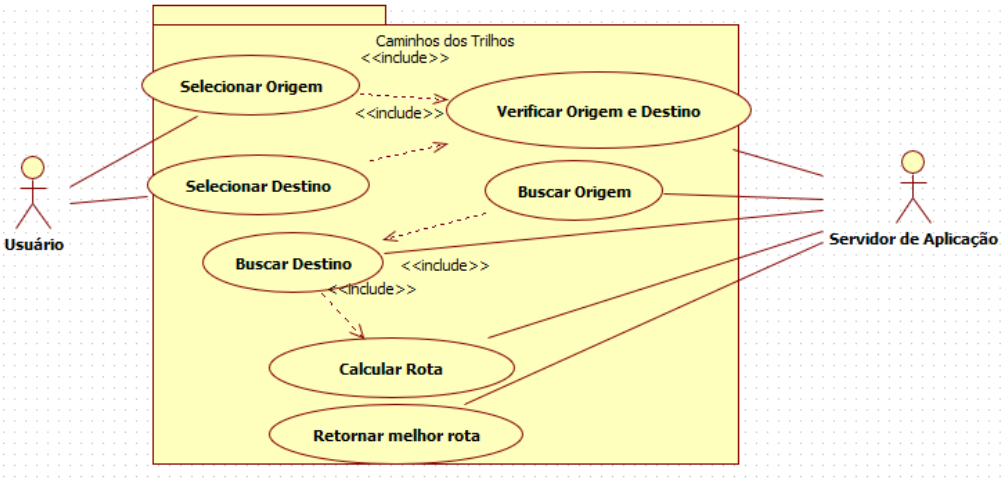

Figura 7. Diagrama de casos de uso do "Caminhos dos Trilhos SP".

4.3. O Protótipo do Aplicativo

Como limitações, neste primeiro momento, o aplicativo funcionará apenas em aparelhos Android, por estar sendo desenvolvido utilizando o Android Studio, um programa de criação de aplicativos voltados unicamente para este sistema operacional. No futuro, será feita uma versão desse mesmo aplicativo para o sistema iOS, usado por celulares Apple. Para esta versão, será usada a linguagem Swift, uma extensão da linguagem C++ própria para aplicativos para iOS.

Na tela inicial do aplicativo, serão pedidas as estações de origem e de destino do usuário, para que seja computada a melhor rota. As estações pedidas serão escolhidas em caixas de seleção que, ao ser tocadas, abrirão uma lista contendo todas as estações de metrô e dos trens da CPTM. Após isso, aparecerá um botão "Calcular Rota", que, antes de cumprir tal função, perguntará ao usuário se aquelas realmente serão as estações desejadas. Caso o usuário toque em "Não", continuará na tela inicial, podendo escolher as estações novamente. Caso a opção escolhida seja "Sim", o aplicativo iniciará sua função principal, procurando nas Tabelas *Hash* de cada linha a chave que representa a

estação de origem. Ao encontrá-la, percorrerá a tabela que a possui e verificará se a estação de destino também está nela.

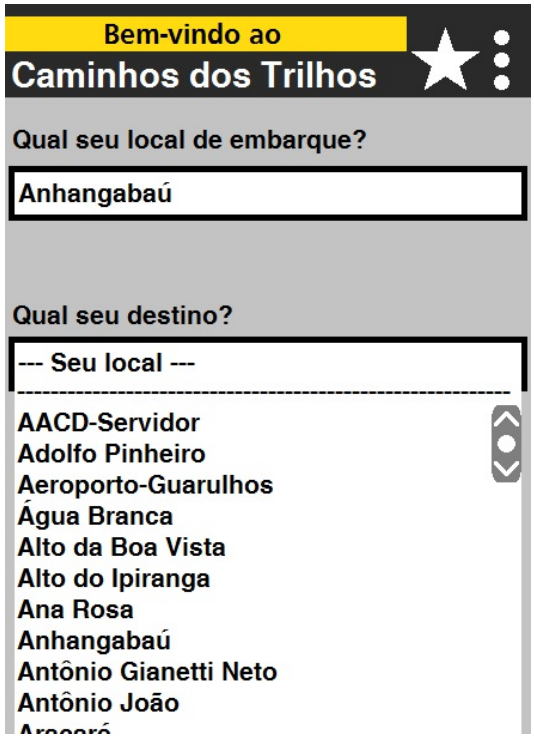

Figura 7. Esboço da tela inicial do "Caminhos dos Trilhos SP".

Independentemente deste resultado, a função passará a percorrer a tabela de adjacências a partir do nó da estação de origem, verificando os vértices um por um, suas possíveis conexões e seus respectivos custos, e acumulando os custos de tempo no vetor de custos enquanto acumula os vértices já verificados em outro vetor, marcando-os. O processo se repete até que a função chegue ao vértice de destino.

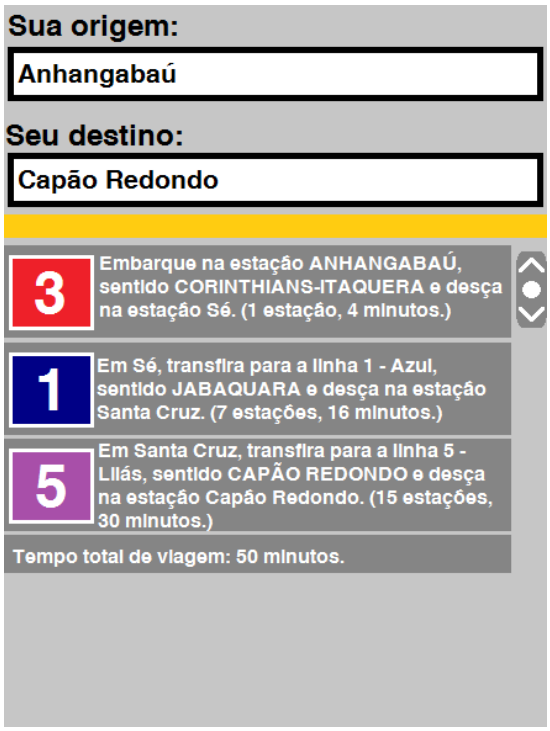

Figura 8. Esboço da tela do "Caminhos dos Trilhos SP" após calcular a menor rota de Anhangabaú a Capão Redondo.

Feito isso, a função irá comparar os custos totais de cada possibilidade calculada, e apenas aquela que tiver a menor soma será mostrada ao usuário em forma de instruções passo a passo. Havendo transferências de grafos ao longo da função, estas serão mostradas ao usuário como transferências de linhas, e seus custos de tempo também serão contados, bem como a quantidade de nós/estações percorridos em cada tabela/linha.

5. Conclusões

Definida a arquitetura como sendo a cliente-servidor, baseado na proposta de (AZLIANOR et al, 2008), no algoritmo de Dijkstra utilizado por (NELSON et al, 2018) e detalhado por (DE CARVALHO, sem data) e (BARROS et al., 2007), e na tabela *Hash* explicada por (BUNGAMA, 2016), foi alcançado o objetivo de propor uma arquitetura e um protótipo de aplicativo voltado para o sistema ferroviário de São Paulo.

Como trabalhos futuros, será feita a implementação do próprio aplicativo, que também passará a mostrar quais as estações mais próximas de pontos de interesse das cidades atendidas pelo sistema ferroviário paulista e como chegar até eles usando suas linhas. Além disso, futuramente, o grafo também será ampliado para tratar da rede de ônibus da capital paulista, mostrando mais opções de rotas entre seus bairros.

Foi sugerido que o aplicativo também tratasse da situação das linhas ("Operação Normal", "Velocidade Reduzida", "Operação Parcial", etc.), visto que suas falhas interferem, não apenas no tempo de viagem, mas, caso seja uma interferência séria,

como obras, na própria viagem, podendo ter estações interditadas. Para isso, será necessário permissão para o uso das APIs “Direto do Metrô e “Direto da CPTM”, que avisam aos usuários do sistema ferroviário sobre possíveis falhas ou interdições. Com a permissão obtida, o aplicativo dependerá de conexão de internet para receber esses dados, fornecidos pelas próprias companhias ferroviárias (Metrô, CPTM, ViaQuatro e ViaMobilidade).

Também foi sugerido um projeto voltado para a rede de ônibus da cidade de São Paulo, o que será aplicado em um trabalho futuro, que mostrará ao público as mudanças nas linhas previstas pela recente licitação de ônibus de São Paulo, como cortes, prolongamentos e extinções de linhas e as alternativas para essas alterações. Essas informações são debatidas entre grupos específicos sobre mobilidade urbana, mas não tão discutidas com o público em geral, que corresponde à maioria dos usuários de transporte público.

Para o desenvolvimento desse novo projeto, além disso, acatando outra sugestão, será criado um *site* HTML em que um novo aplicativo mobile será vinculado. Este receberá do usuário um número de linha de ônibus e chamará uma função, implementada no site supracitado, que verificará a existência de linha com o número recebido, emitindo uma mensagem de erro se não existir. Se o resultado for positivo, retornará ao aplicativo e ao seu usuário o futuro da linha recebida, se ela será prolongada, encurtada ou extinta, e, nos dois últimos casos, suas alternativas, que podem ser outras linhas de ônibus, ou o uso de metrô ou trem. Esse método de implementação perde a não-dependência de conexão de internet buscada no primeiro projeto, mas facilita o desenvolvimento do aplicativo, podendo torná-lo multiplataforma, para que seja usado tanto em computadores, como em celulares iOS e Android.

6. Referências Bibliográficas

AZLIANOR, A. A. et al. A Mobile web application architecture for generating destination-oriented LRT Route. In: Proceedings of the WSEAS International Conference on Applied Computing Conference. World Scientific and Engineering Academy and Society (WSEAS), 2008. p. 242-244.

Barros E. A. R., Pamboukian S. V. D., Zamboni L. C. (2007). Algoritmo de Dijkstra: apoio didático e multidisciplinary na implementação, simulação e utilização computacional. In International Conference on Engineering and Computer Education (ICECE). Academic Press.

Bungama P. A. (2016). Um repositório chave-valor com garantia de localidade de dados. In Universidade Federal do Paraná (UFPR). Academic Press.

De Carvalho B. M. P. S. (sem data). Algoritmo de Dijkstra. Universidade de Coimbra. Academic Press.

JEON, Hyeyoung; RHEW, SungYul. Mobile Software Architecture Technique and Application for Using Business Logic and View. In: International Conference on Hybrid Information Technology. Springer, Berlin, Heidelberg, 2012.p.458-464

Nelson Q., Steffernsmeier D. & Pawaskar S. (2018). A simple approach for sustainable transportation systems in smart cities: a graph theory model. In IEEE Conference on Technologies for Sustainability (SusTech). Academic Press.