PROJETO FINAL DE GRADUAÇÃO

APLICAÇÃO WEB DE BUSINESS INTELLIGENCE E ANALYTICS PARA SUPORTE NA TOMADA DE DECISÃO EM MANUTENÇÃO VIÁRIA NO DF

Marcos Guo Yan

Brasília, dezembro de 2017

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Redes de Comunicação

PROJETO FINAL DE GRADUAÇÃO

APLICAÇÃO WEB DE BUSINESS INTELLIGENCE E ANALYTICS PARA SUPORTE NA TOMADA DE DECISÃO EM MANUTENÇÃO VIÁRIA NO DF

Marcos Guo Yan

Relatório submetido como requisito parcial de obtenção de grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Dr. Georges Daniel Amvame Nze, _____
ENE/UnB
Orientador

Profª. Msc Beatriz Campos Santana de Araújo, _____
ENE/UnB
Examinador interno

Prof. Especialista Diego Martins de Oliveira , _____
IFB/UnB
Examinador externo

Brasília, dezembro de 2017

FICHA CATALOGRÁFICA

Yan, Marcos Guo
Aplicação WEB de Business Intelligence para suporte na tomada de decisão em manutenção viária no DF.

[Distrito Federal] 2017.

(FT/UnB, Engenheiro, Redes de Comunicação, 2017). Projeto Final de Graduação – Universidade de Brasília. Faculdade de Tecnologia, Departamento de Engenharia Elétrica.

REFERÊNCIA BIBLIOGRÁFICA

YAN, MARCOS GUO, (2017). Aplicação WEB de Business Intelligence para suporte na tomada de decisão em manutenção viária no DF. Projeto Final de Graduação em Engenharia de Redes de Comunicação, Publicação FT. Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF.

CESSÃO DE DIREITOS

AUTOR: Marcos Guo Yan

Aplicação WEB de Business Intelligence para suporte na tomada de decisão em manutenção viária no DF.

GRAU: Engenheiro ANO: 2017

Marcos Guo Yan

12/12/2017

Brasília, DF.

Dedicatória

Dedico este trabalho a Deus, à minha família, ao amor que tive sorte de encontrar, aos meus amigos e colegas da UnB.

Marcos Guo Yan

Agradecimentos

Agradeço primeiramente a Deus por me guiar mesmo em momentos em que me senti completamente perdido nessa longa caminhada que foi a graduação.

Agradeço aos meus pais pela paciência e confiança que sempre me depositaram. Minha mãe por estar sempre confiante e segura de que eu conseguiria atingir meus objetivos, mesmo quando eu não fazia ideia do que fazer para atingi-los. Meu pai por sempre me passar palavras de sabedoria e me inspirar a seguir essa jornada dentro da engenharia.

Agradeço à Monica Guo, irmã que sempre esteve ao meu lado e por quem tenho um carinho maior que o mundo. Obrigado por me motivar ou dar bronca para que eu estudasse e cumprisse minhas tarefas.

Agradeço à Mayanna Nogueira, minha companheira nas horas mais difíceis e a quem tudo posso confidenciar na certeza de que vou receber as mais sábias orientações. Obrigado pela paciência e pelo amor que é maior do que poderia imaginar. Te amo por tudo que me faz ser.

E o que seria da vida sem os amigos? Artur Araújo e Matheus Soares, obrigado por me inspirar a ser um homem melhor desde a época em que estudávamos no Ensino Médio - não sei se suportaria ser a pessoa que era e sou eternamente grato por me ensinar o que eu nunca aprenderia sozinho. Bernardo Klimkievicz, meu grande amigo de infância e que compartilha da minha vida até mesmo no outro lado do mundo - obrigado por ser um ombro amigo desde que me conheço por gente. Lucas Willian, a maior surpresa que tive durante a minha graduação e meu apoio durante as horas mais difíceis no final do curso.

Obrigado a todos colegas pelo imenso apoio, sem o qual eu não estaria onde estou hoje. O aprendizado maior que levo de todos esses anos na UnB é que na vida não existe linha de chegada. Então vamos aproveitar a jornada.

Marcos Guo Yan

RESUMO

Este trabalho dá continuidade ao Projeto Final de Graduação entitulado "Aplicativo Mobile de Parametrização e Precificação de Reparos de Buracos Viários no DF". O trabalho se propõe a facilitar o gerenciamento no que diz respeito a manutenção e reparo do corpo viário no Distrito Federal.

Um problema recorrente em todo o território nacional consiste no desenvolvimento, por parte dos órgãos públicos, de vias de baixa qualidade que consequentemente dão origem a buracos. Na última década, a frota de veículos presentes no Brasil praticamente dobrou [1], apontando para a necessidade de manutenção regular da pavimentação das pistas. No geral, os buracos trazem prejuízos significativos aos usuários das vias públicas, podendo até mesmo causar acidentes graves. Um segundo fator em análise é a falta de transparência quanto ao custo associado às operações de tapa-buracos, de sorte que poucas pessoas têm consciência do dispêndio de recursos no processo de repavimentação de uma fissura.

Propõe-se, então, um painel de *Business Intelligence* (BI) que consegue agilizar a tomada de decisão em um cenário orçamentário fixo e transparente, visando uma melhoria da qualidade viária geral. Serão utilizados os dados gerados no primeiro trabalho citado, de forma organizada e concisa de forma que o gerente de projeto de reestruturação viária consiga realizar seu trabalho de maneira totalmente automatizada: visualizar tickets gerados por usuários, responder tickets, ter métricas relacionadas aos buracos e usuários que alimentam a base de dados e também ter acesso ao painel financeiro que reflete toda a operação, no caso, pública.

Como resultado, apresenta-se uma plataforma de BI que faz o mínimo de requisições ao servidor de forma a otimizar o tempo de resposta e custos que são inerentes de um volume grande de dados. Além disso, o gerente tem um mapa em sua tela principal e todos os alertas de buracos criados pelos usuários estão disponíveis para consulta. Esse acesso visual cria um apelo muito grande em questão de gestão de prioridade e mapeia de forma clara quais regiões estão com maior número de informes. Assim, estudos futuros podem detalhar os motivos pelos quais algumas vias estão mais sujeitas a danos em um mesmo período de tempo, apesar de ter a mesma estrutura físico-química de outra via que necessita de menos reparos.

Palavras Chave: parametrização, buracos, rodovias, vias públicas, DF, Business Intelligence, BI

ABSTRACT

This project sequels the Undergraduate Final Project titled "Mobile Application for Parametrization and Pricing of Potholes on Repair Process in DF". This work comes with the purpose to facilitate management regarding maintenance and repair of the roads in *Distrito Federal*.

A recurring problem that affects all the national territory consists on the construction of low quality roads by public agencies. Consequently, the roads need constant repair. In the last decade the vehicle fleet in Brazil practically doubled [1], pointing out for the need to have a regular maintenance system. Generally, potholes brings significant damage to the users of the roads, even causing serious accidents. A second factor that is in analysis is that there is a serious lack of transparency regarding the expenditure of the repairing operations, in a sense that few people acknowledge how much is spent during the process of repaving.

Therefore a Business Intelligence panel is created to speed up the decision making process with a fixed and transparent budget, with the objective of improving the road quality in general. The data provided from the first project will be used in an organized and concise way, so that the restructuring project manager can work in a completely automated manner: view user created tickets, reply to tickets, have metrics that relates to potholes and users that feed the database and also have access to the financial panel - which reflects all public operation.

The BI panel can make minimal requests to the server so that can optimize the response time and budgets associated with a high data volume. Besides that, the manager will have access to a map in his main screen with all the alerts of potholes that the users created. This visual access can prioritize and show in a clear way which regions are the most affected by road damages. In that sense, future studies can detail why some areas are more sensitive to damage over time than others, even though their physical-chemical properties are the same.

Keywords: parametrization, potholes, roads, DF, Business Intelligence, BI

SUMÁRIO

LISTA DE FIGURAS

Capítulo 1

Introdução

1.1 Contextualização

Um desenvolvimento rodoviário e viário é essencial para o crescimento econômico e social de um país. Com uma malha de locomoção eficiente, é possível que serviços e produtos sejam prestados/entregues em um tempo hábil muito menor e garante uma economia de tempo que consequentemente gera uma economia de custos para empresas, pessoas físicas e o governo.

A importância das vias e rodovias no Brasil pode ser facilmente notada acompanhando os números de pesquisas que analisam o crescimento de seus usuários. De acordo com a pesquisa desenvolvida pelo Instituto Brasileiro de Geografia e Estatística (IBGE) [1], em 10 anos o número de carros no Brasil cresceu de forma expressiva, tendo mais que dobrado nesse período, passando de 24 milhões para aproximadamente 50 milhões de veículos. Tal crescimento deve ser acompanhado pelos órgãos governamentais a fim de que sejam providas pistas de qualidade para a crescente quantidade de usuários.

Uma prática que tem se tornado bastante comum nos processos de licitação para obras de infraestrutura urbana é o elevado preço dos projetos, os quais têm consumido recursos públicos de uma maneira alarmante. Diante da falta de informação e da falta de transparência, a população, em geral, não acompanha os projetos e tão pouco o dispêndio financeiro para os tais [2]. Operações tapa-buraco estão espalhadas em todas as cidades e isso motivou a continuação do projeto "Aplicativo Mobile de Parametrização e Precificação de Reparos de Buracos Viários no DF", visto que é importante que se tenha uma ferramenta capaz de realizar uma estimativa aproximada dos custos associados a cada buraco presente nas rodovias, de forma que haja um controle dos recursos públicos. Este projeto visa, acima de tudo, apresentar uma alternativa de

fiscalização por parte da população - além do Portal da Transparência [3] - e de gestão para órgãos que fazem a manutenção e revitalização das vias e rodovias públicas do DF especificamente neste caso.

1.2 Definição do problema

Como existe uma descentralização de informações por parte de cada município, Estado e Distrito, não existe uma diretriz de manutenção viária que se siga no país como um todo. Apesar de existir a norma ABNT NBR 9935/2005 [4], que diz respeito ao material utilizado na pavimentação e repavimentação, o conceito "material sem forma ou volume definido, geralmente inerte, de dimensões e propriedades adequadas para produção de argamassas e de concreto" deixa muitas brechas para que o material utilizado seja de baixa qualidade. Basicamente, o material descrito é usado como base para prover a estruturação de um pavimento, ou seja, ele é usado na base do revestimento.

Pegando o exemplo do DF, em 2016 e apenas seis meses da Operação Tapa-Buraco, foram gastos R$40 milhões para manutenções de vias, pavimentação e reparos na rede de drenagem pluvial. No mesmo período, R$ 1,337 milhão em massa asfáltica foi repassado às administrações regionais. Na época de chuvas, o repasse é de R$ 10 milhões por mês às empresas terceirizadas responsáveis pela manutenção. A Novacap produz, por dia, 500 toneladas de massa asfáltica para atender à demanda. O custo da contratação das terceirizadas, por sua vez, foi de R$ 26,3 milhões. De acordo com a Novacap, são sete equipes da empresa pública e onze terceirizadas atuam diariamente na cidade para a prestação do serviço [5].

1.3 Objetivos

Este projeto dá continuidade ao Projeto Final de Graduação entitulado "Aplicativo Mobile de Parametrização e Precificação de Reparos de Buracos Viários no DF"[2] e visa trazer um viés prático para a solução que surgiu como ferramenta colaborativa entre usuários das vias públicas no DF. O intuito é utilizar os dados obtidos pelos usuários através do aplicativo, para uma análise de dados e tomada de decisão mais eficiente por parte de gestores e gerentes de projetos de construção e manutenção viária e rodoviária. Com isso, espera-se poder aumentar a eficiência da operação e também agilizar os processos internos, trazendo uma acurácia maior em relação a prazos, gestão de recursos e de pessoas.

1.3.1 Objetivo Geral

O primeiro passo para que os processos referentes ao ato de manutenção e reparo sejam eficientes e unificados é organizar os dados. Com essa proposta, o projeto cria um ambiente centralizado de informações, em que os responsáveis pela qualidade viária tenham ferramentas que levantam métricas relevantes para a atuação decisiva sobre: onde realizar reparos de acordo com o número de reclamações, indicar às equipes técnicas a prioridade de atendimento, além de fiscalizar o compromisso orçamentário de forma clara.

1.3.2 Objetivos Específicos

A automação de processos que são descentralizados vai trazer uma autonomia maior para o(s) gestor(es) que está(ão) na operação, de forma que todo o cronograma, orçamento, número de agentes disponíveis e onde estão disponíveis estarão visíveis em uma só plataforma. É muito importante ressaltar que todas as ferramentas utilizadas são *open-source* e possuem uma comunidade dedicada para a iteração e melhoria da supracitada. Atualmente as demandas chegam para as equipes da Novacap [5] por meio da Ouvidoria-Geral do DF e a ideia é que utilizem o aplicativo desenvolvido no projeto que precede este, para que se possa modelar os dados de forma a serem úteis na gestão das equipes da Novacap.

Capítulo 2

Fundamentação teórica

Por ser um projeto que se utiliza de inúmeras ferramentas e tecnologias, esta seção tem o intuito de apresentar um panorama geral e definir exatamente os componentes e seus jargões necessários para que o sistema funcione.

2.1 Client-side

Este trabalho cita em diversas ocasiões se determinada funcionalidade está desenvolvida ou implementada no *client-side* ou *server-side*. Normalmente um cliente se trata da aplicação, então um *web browser* que o usuário roda no seu computador, *smartphone* ou qualquer dispositivo que se conecte ao servidor é considerado um cliente. Desta forma, o *client-side* se refere a todas as operações que são realizadas pelo usuário. No caso do painel de BI, o usuário pode enviar emails e realizar agendamentos, por exemplo. Em suma, o *client-side* deve ser uma visualização do que o usuário pode fazer em um sistema, facilitando tarefas e operações que possuem lógicas - já adiantando - no *server-side* [6].

2.2 Server-side

De forma análoga, refere-se ao *server-side* todas as operações que são realizadas pelo servidor. No escopo deste projeto foram definidas duas frentes: implementar um servidor remoto para acesso do aplicativo móvel Android e implementar o painel BI. O servidor remoto foi pri-

meiramente implementado em um *notebook* e o mesmo não poderia ficar desligado em nenhum momento. Toda vez que havia uma queda ou lentidão na conexão de internet, o serviço se deteriorava no lado do cliente porque não se estabelecia a transmissão e recebimento de dados. Um detalhe é que neste primeiro servidor o IP deveria ser mantido estático, de forma que quando se conectasse a uma nova rede, ou qualquer outro tipo de falha ocorresse, a comunicação fosse reeestabelecida em um mesmo endereço - tudo isso é explicado pelo fato do aplicativo definir um endereço padrão de requisições e caso um novo endereço fosse designado seria necessário mudar isso no aplicativo Android e gerar um novo APK, fato inviável para uma aplicação social deste porte. A figura abaixo representa bem simplificadamente as diferenças entre o lado do cliente e do servidor [7].

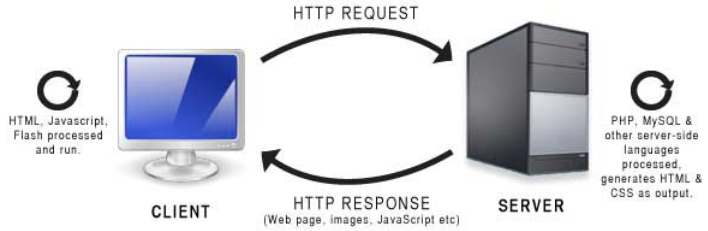


Figura 2.1: Diferença entre *server/client side* (Fonte: Tech Notes)

2.3 Banco de Dados

Um banco de dados é um conjunto de informações organizadas em forma de tabelas. Atualmente, a principal função de um banco de dados num ambiente corporativo/empresarial é o controle de operações e levantamento de métricas - que é exatamente a proposta desse projeto [8].

Existem diversos tipos de banco de dados e não é mérito descrever cada um individualmente. O foco é no banco de dados relacional, que é o mais comum de se utilizar pela sua flexibilidade e proposta de valor. É possível guardar dados estruturados e recuperá-los eficientemente com determinados critérios (*query*) utilizando a linguagem SQL, por exemplo [8].

Na modelagem dos dados, utiliza-se muito o processo chamado de normalização. Esse processo diminui a complexidade com a redução de redundâncias e oferece um acesso aos dados de forma mais rápida. Se existem dados repetidos ou uma cadeia de dependência entre múltiplas variáveis, a normalização basicamente vai eliminar a duplicação e garantir que as dependências façam sentido.

2.3.1 Banco de dados relacional

Para gerenciar um banco de dados relacional, utiliza-se ferramentas como um RDBMS (*Relational Database Management System*). Existem diversos disponíveis no mercado e o mais famoso é o MySQL, que é *open-source* e está disponível desde 1995 [9]. Escrito em C e C++, o MySQL é utilizado por companhias gigantescas de diversas áreas do mercado, entre elas o Facebook [10], Twitter e Youtube [9].

A instalação pode ser feita através de um binário, mas na pilha LAMP isso não é possível, então recomenda-se instalar através da linha de comando pelo repositório oficial do MySQL. Após isso, torna-se possível configurar todo o banco pela GUI, mas é possível fazer o mesmo através da linha de comando.

O MySQL define-se como um grande RDBMS no sentido de que os recursos de *hardware* são mínimos e suportam praticamente qualquer plataforma atual, com excelente desempenho e estabilidade. A sua documentação e comunidade são gigantes e provavelmente qualquer problema que um desenvolvedor usual venha a ter já tem uma solução muito bem descrita.

Já o SQL (*Structured Query Language*) é uma linguagem específica para utilização em manipulação de dados. São diversas as operações e comandos que se pode utilizar para alterar configurações do banco e seus dados, tais como inserir, alterar, deletar, consultar e disponibilizá-los. Existe uma interface visual para o MySQL com esses principais comandos, mas optou-se utilizar a linha de comando para um melhor controle e entendimento da ferramenta. Uma representação visual comparando os três RDBMS mais famosos é dada abaixo [11].

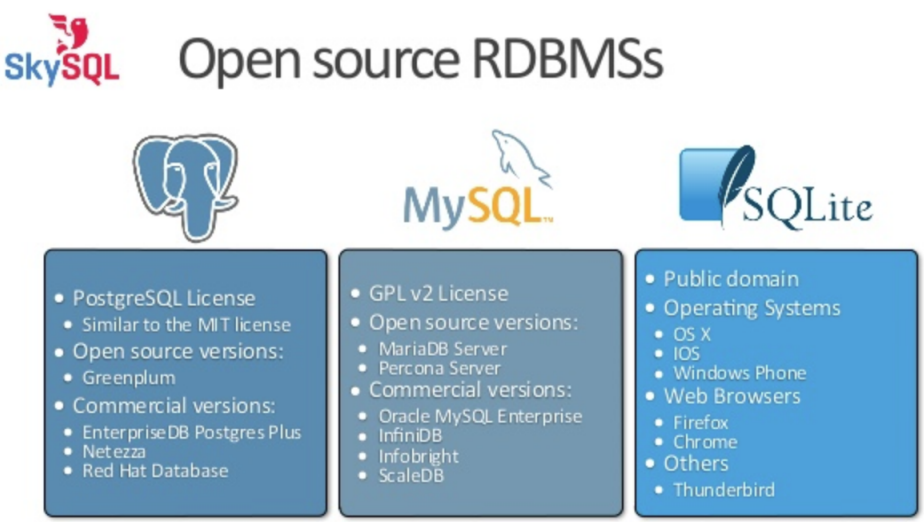


Figura 2.2: Comparativo entre RDBMS *open-source* (Fonte: Ivan Zoratti)

2.4 LAMP

LAMP é um acrônimo para Linux, Apache, MySQL, PHP podendo ter variantes em relação aos componentes de cada um. Essa chamada *stack* de softwares e serviços *open-source* é o pilar de toda a estrutura desse projeto. Essa junção dos quatro componentes é que segmenta uma base para construção de operações de mais alto nível a serem executadas para o que se deseja na aplicação [12]. Cada componente do LAMP será detalhado a seguir [13].

Figura 2.3: Modelo de servidor a ser seguido no projeto (Fonte: ProgrammableWeb)

2.5 Linux

Atualmente a intitulação de Linux serve para qualquer distribuição de sistema operacional construída com base no Linux *kernel*. Esse *kernel* teve seu primeiro sistema operacional lançado em 1991 [14] e é o sistema operacional que suporta os acessos e processamentos mais rápidos, sendo utilizado como *mainframe* dos computadores mais rápidos do mundo [15]. Além disso, 67% do total de servidores web no mundo utilizam o Linux [16] ou alguma distribuição do *kernel*. Esse número por si só já consegue mostrar a dimensão que o Linux tem, sinal de que seu desempenho e valor possuem um desempenho claramente superior à outros OS.

2.6 Apache

O servidor HTTP Apache é um projeto da *The Apache Software Foundation* com a proposta de desenvolver e manter um servidor HTTP *open-source* para sistemas operacionais modernos, incluindo UNIX e Windows [17]. O objetivo é prover um ambiente seguro, eficiente e extensível no sentido de que o servidor possa fornecer serviços HTTP de acordo com os padrões definidos atualmente pela *International Telecommunication Union* [18]. O projeto *The Apache*

HTTP Server (httpd) tem 92% de seu uso integrado em distribuições Linux [19]. O servidor não somente deve armazenar, processar e fornecer as páginas web para os clientes através do protocolo HTTP (*Hypertext Transfer Protocol*), como também receber dados de clientes através de formulários por exemplo.

2.7 PHP 7

O PHP é uma linguagem *server-side* de *script* que pode ser embarcada no lado do cliente para comunicação com o servidor. Por ser uma linguagem que pode ser usada para uso geral, é claro que possui diversos outros usos, porém é tão famoso por estabelecer essa comunicação entre cliente e servidor, que até mesmo o Facebook foi escrito em PHP e até hoje o utiliza [20]. Será utilizada a última versão estável do PHP, que é o PHP 7. Em benchmark recente baseado em plataformas WordPress, foi observada uma performance 100% superior nos quesitos de *caching* e compactação de estruturas de dados [21]. Existem duas maneiras de se adicionar o suporte PHP a um servidor web: como um módulo de servidor web nativo ou como um executável CGI. Para este projeto, se faz proveito do módulo de interface direto ao Apache.

2.8 Desenvolvimento Web

Em um ambiente de desenvolvimento *web*, é importante ressaltar que existem várias tecnologias como bibliotecas (jQuery, React) e *frameworks* (Angular, Bootstrap) e todas têm como base o *output* nas três bases de uma apresentação *web*: HTML, CSS e JS [22]. Cada uma dessas tecnologias será apresentada a seguir.

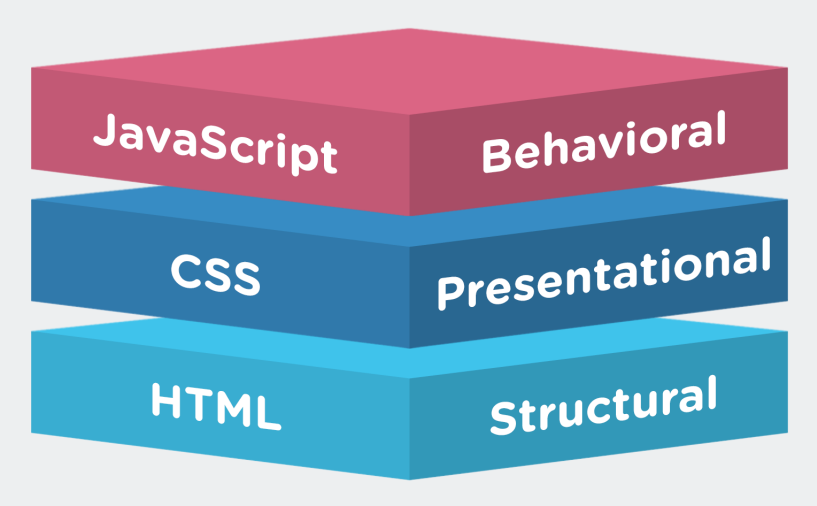


Figura 2.4: Pilares do desenvolvimento *web* (Fonte: 1Training.org)

2.8.1 HTML

HTML é um acrônimo para *Hypertext Markup Language* e é a linguagem de marcação padrão para criação de páginas e aplicações web. Os navegadores que usuários utilizam em seus computadores recebem o HTML de um servidor web e renderizam uma forma visual de multimídia. Mais amplamente, pode ser considerada uma linguagem de dados. HTML apenas encapsula dados e descreve o que fazer com eles, não como fazer. Ela não é linguagem de programação porque não é *Turing complete*. Ou seja, ela precisaria ter algumas características específicas para poder "programar um dispositivo". Não é possível executar o HTML. Para se ter essa complitude de Turing, o HTML deveria fazer cálculos, mudar informações contidas em algum tipo de memória, tomar decisões e mudar o fluxo de execução. Entretanto, é possível embutir linguagens de programação dentro do HTML, tal como o Javascript, que será descrito adiante [23].

2.8.2 CSS

Um outro pilar no desenvolvimento web, o *Cascading Style Sheets* (CSS) é uma linguagem de estilo utilizada para descrever a apresentação de um documento, geralmente, HTML. Essa divisão entre estrutura (HTML) e apresentação (CSS) ajuda muito na organização durante o processo de codificação e desenvolvimento. A apresentação engloba itens como *layout*, cores e fontes. No caso deste projeto, a codificação dos estilos está nos arquivos .css e realmente se tem uma flexibilidade muito maior em relação ao reuso de código, diminuindo significativamente a duplicação do mesmo [24].

Cada navegador usa sua própria *layout engine* que basicamente faz a renderização do CSS, o que pode trazer diferenças notórias em uma comparação de um mesmo sistema em sistemas/navegadores diferentes. Para efeito de padronização, utilizou-se o Google Chrome e Mozilla Firefox para os testes.

2.8.3 Javascript

Javascript (JS) é uma linguagem de programação interpretada de alto nível. Juntamente com o HTML e o CSS, o Javascript forma a estrutura básica de um sistema web e 94.8% dos sites utilizam esse formato [6].

Utilizado para dar características interativas e mais dinâmicas aos sites estáticos, o Javascript permite, inclusive, suporte para jogos online. Apesar de ter sido implementado para ser uma linguagem que opera no *client-side*, é possível embarcar outros tipos de software que manipulam as instruções no nível do servidor. Apesar do nome, o Javascript é muito diferente do Java. O Javascript é uma linguagem de programação de *script* orientada ao objeto, sendo o Java

uma linguagem de programação orientada ao objeto. Assim, o Java precisa rodar em uma VM (máquina virtual), e não consegue fazê-lo em um navegador como o Javascript [6].

Assim, a Figura 2.4 consegue ilustrar claramente o que se descreveu em 2.8.1, 2.8.2 e 2.8.3. O HTML é estrutural, então vai responder "o que isso significa?"para o usuário, enquanto o CSS diz "como vai ser a apresentação?"e o JS se preocupa em "como cada elemento se comporta?".

2.9 Ajax

Diante das três ferramentas mais básicas para o desenvolvimento web citadas acima, apresenta-se o Ajax. Em diversas ocasiões, faz-se o carregamento de uma página web e a informação que nela está contida já foi mudada. Essa característica é de programação síncrona, em que existe um cascateamento de operações e funções sendo executadas. Muitas vezes se faz essencial um carregamento paralelo e que atualize o sistema sem necessidade de novas requisições ao banco. Isso representaria um gasto enorme de *queries* e impactaria no custo de uma operação como um todo (custos de servidor, principalmente). O Ajax traz essa característica de carregamento em *background*, sendo por isso um acrônimo de *Asynchronous Javascript and XML* [25].

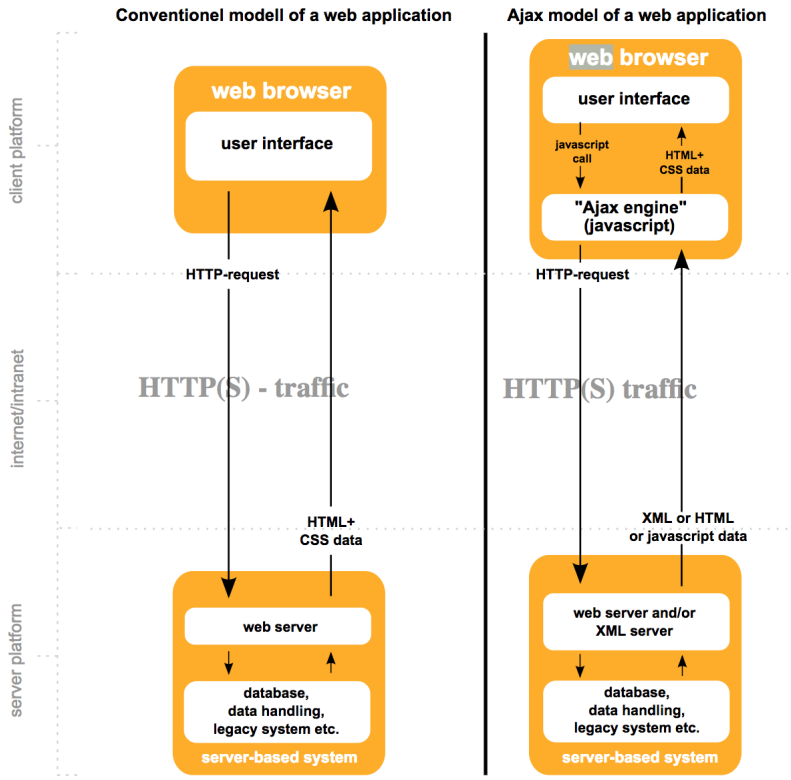


Figura 2.5: Fonte: Arquitetura web com Ajax (Fonte: SegueTech)

Um exemplo do uso do Ajax seria a atualização ou envio de informações do cliente para o servidor sem precisar carregar a página inteira - imagine um usuário do Netflix: sempre que ele avalia um filme, não se recarrega a página toda. Pelo contrário, a nota é enviada sem precisar gerar um volume de dados alto para os dois lados operantes. Consequentemente, o Ajax também é uma linguagem que acelera muito o desempenho de sistemas, consistente com o aumento no grau de satisfação dos usuários.

2.10 GPS e Assisted GPS

Amplamente utilizado e embarcado em diversas aplicações e *hardware*, o *Global Positioning System* é um sistema de navegação baseado no uso de uma constelação de 24 satélites espaçados igualmente em seis planos orbitais para triangulação de coordenadas terrestres e devem funcionar 24 horas por dia sem parar, em quaisquer condições climáticas e sem taxas de assinatura ou configuração para ser utilizado [26].

Os satélites GPS orbitam precisa e completamente a Terra duas vezes ao dia, a uma altura de 20,200 quilômetros. Cada um transmite um sinal único de parâmetros orbitais que permitem aos dispositivos GPS decodificar e computar a localização precisa do satélite que transmitiu a informação original. Basicamente, o dispositivo vai calcular a distância em relação a cada satélite e o tempo qu se levou para receber aquele sinal. Dependendo da precisão que se deseja no dispositivo pode-se utilizar mais ou menos satélites para obter uma média mais precisa. Atualmente os dispositivos têm uma média de precisão de 10 metros [27]. De acordo com o grau de precisão e taxa de atualização do dispositivo, utiliza-se mais ou menos sinais dos satélites para a finalidade.

Todavia, o GPS perde muita precisão ou até mesmo deixa de funcionar quando um ambiente possui muitos obstáculos que interferem diretamente na qualidade de transmissão e penetração das ondas que transmitem o sinal [28]. Para contornar essa deficiência do GPS e permitir que usuários consigam utilizar a tecnologia, foi desenvolvida uma nova arquitetura chamada de *Assisted* GPS, ou simplesmente AGPS.

Segundo Goran Djuknic e Robert Richton da Bell Laboratories, o AGPS oferece uma maior precisão, disponibilidade e cobertura por um preço mais em conta. Para isso, conta com uma arquitetura que inclui um dispositivo com receptor GPS, um servidor AGPS com a referência do receptor e que faz a comunicação com a constelação de satélites e uma infraestrutura de rede sem fio que consiste em uma estação rádio base (ERB). A figura a seguir ilustra cada componente descrito acima na configuração de um sistema operante [28].

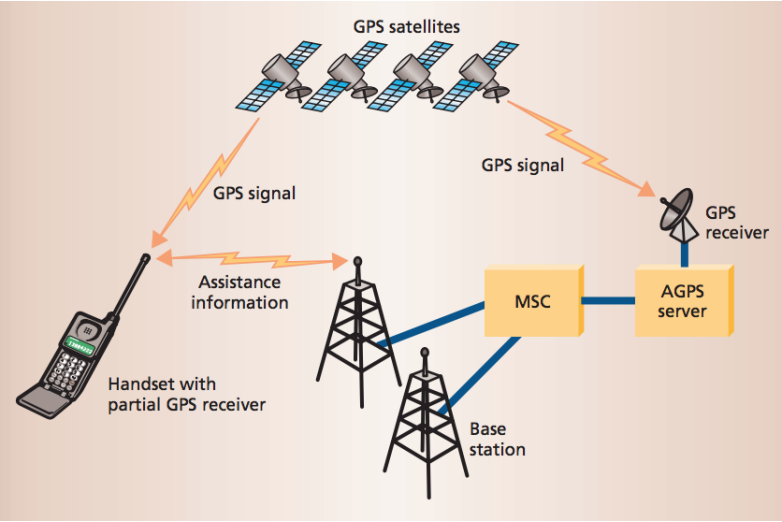


Figura 2.6: Arquitetura do *Assisted*-GPS.

Os componentes principais do sistema são: um *handset wireless* com recptor GPS, um servidor AGPS com referência para o receptor, uma estrutura de rede *wireless* consistindo em estações de rádio-base e um MSC (*Mobile Switching Center*). Com o auxílio de ERBs, essa nova arquitetura permitiu que o GPS fosse utilizado de maneira escalável em nível global. Praticamente qualquer *smartphone* fabricado atualmente tem suporte à tecnologia. Por este motivo, aliou-se essa ferramenta poderosa em um aplicativo para mostrar em tempo real a situação de precatoriedade das vias públicas no DF. Isso contribui para um colaborativismo social e econômico enorme, além de se ter métricas e dados em relação a períodos de tempo a partir de agora.

2.11 Web API

De acordo com David Emery, uma *Application Programming Interface* (API) é um conjunto de funções, rotinas, subrotinas e protocolos que auxiliam no desenvolvimento de uma aplicação. No início da criação de processos para programação e desenvolvimento ágil, percebia-se uma grande quantidade de duplicação de código. Para aproveitar o reuso e facilitar a vida dos desenvolvedores, muitos fornecedores de serviços como *software* passaram a disponibilizar APIs [29]. O propósito é simples - tornar a vida de desenvolvedores mais ágil e produtiva. É criada uma simplificação das camadas mais abstratas de um serviço e quando se escreve o código é possível pensar somente nas camadas mais superiores e palpáveis, como objetos e ações. Em suma, supondo que o programador quer realizar uma ação com função já disponível em uma API, ele pode chamar essa função sem entender o funcionamento da mesma - ele só precisa saber a entrada e saída dela.

No caso de uma Web API, a comunicação se dá entre interfaces de uma empresa públi-

ca/privada ou projetos *open-source* e aplicações que utilizam seus serviços/produtos. Tipicamente, esse tipo de API se constrói com base em requisições e respostas HTTP que têm uma estrutura em XML ou JSON, por exemplo. As instruções de comunicação de mais baixo nível não precisam ser executadas ou requisitadas diretamente pelo desenvolvedor. Esse projeto irá utilizar APIs *open-source* para criação de gráficos e modelagem de dados em mapas utilizando informações coletadas por AGPS e armazenadas em um banco de dados.

2.12 Javascript Object Notation

A IETF (*Internet Engineering Task Force*) publicou um memorando em 2006 sobre a estrutura oficial do *Javascript Object Notation* (JSON). Nele, definiu-se o JSON como um tipo de arquivo baseado em texto com uma formatação com um conjunto de regras com finalidade de deixá-lo mais compacto [30]. A seguir é apresentado um exemplo de formato JSON utilizado neste projeto para leitura e ordenação facilitada dos dados recebidos.

```
[
  {
    "LagoSul": "4"
  },
  {
    "Sudoeste": "3"
  },
  {
    "Taguatinga": "1"
  }
]
```

Esse tipo de formato é amplamente utilizado em sistemas web que utilizam comunicação assíncrona entre navegador e servidor. Apesar de ter sido desenvolvido para ser um subconjunto do Javascript, o JSON acabou se tornando um padrão internacional de formatação de dados sendo uma linguagem independente [31].

2.13 XML

Assim como o HTML, o XML é uma linguagem de marcação que é acrônimo para *Extensible Markup Language*. O interessante do XML, assim como JSON, é que é uma linguagem facilmente interpretada tanto por máquina quanto por humano. Essa facilidade visual que as duas linguagens trazem com suas respectivas organizações com toda certeza foi um fator para sua popularização e ampla utilização. É, então, uma formatação de texto para dados e pode até ser utilizada para representação de estruturas de dados.

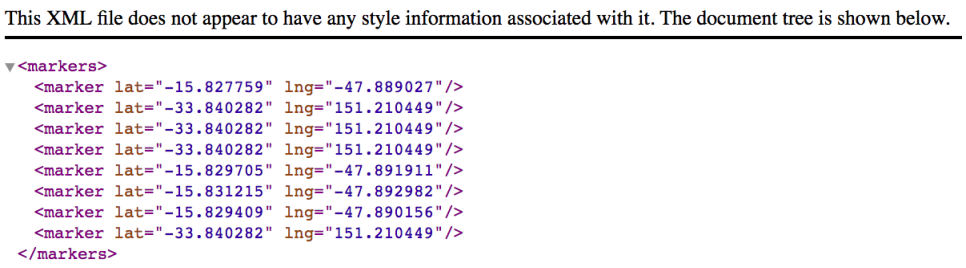


Figura 2.7: Formato XML. (Fonte: pesquisa)

O XML permite um tratamento da comunicação a partir do DBMS que facilita a manipulação dos dados. Essa manipulação será detalhada de forma mais clara na descrição da metodologia de desenvolvimento.

2.14 Levantamento de requisitos

O conceito de levantamento de requisitos é aplicável a qualquer modelo que tenha um fornecedor de produto ou serviço e um cliente que deseja fazer uso daquele produto ou serviço. No caso deste trabalho, quando se fala de levantamento de requisitos, refere-se ao contexto de *software*.

O levantamento de requisitos de *software* baseado em uma arquitetura de modelagem de negócios é apoioado em um conjunto de métodos que trazem a questão do *Unified Process*, que nada mais é do que unificar processos. Quatro modelos de engenharia de *software* têm sido amplamente discutidos: o ciclo de vida clássico (ou cascata), a prototipação, o modelo espiral e as técnicas de Quarta Geração [32]. O ponto em comum de todos esses modelos é a existência de uma fase de análise de requisitos. Entende-se, então, que essa etapa é uma das mais importantes dentro de um projeto - se mal executada, pode custar tempo e dinheiro, algo que empresas não podem se dar ao luxo de desperdiçar de forma alguma.

O fluxograma abaixo ilustra os passos necessários dentro de um levantamento de requi-

sitos que leva o nível de ambiguidade ao mínimo e aumenta a coesão em relação às definições do que o cliente quer do seu desenvolvedor [32]. Esse conceito é muito importante por trazer a característica de definir o que se planeja desenvolver e se é isso que realmente deve ser feito - tudo através de uma organização de documentação e registros.

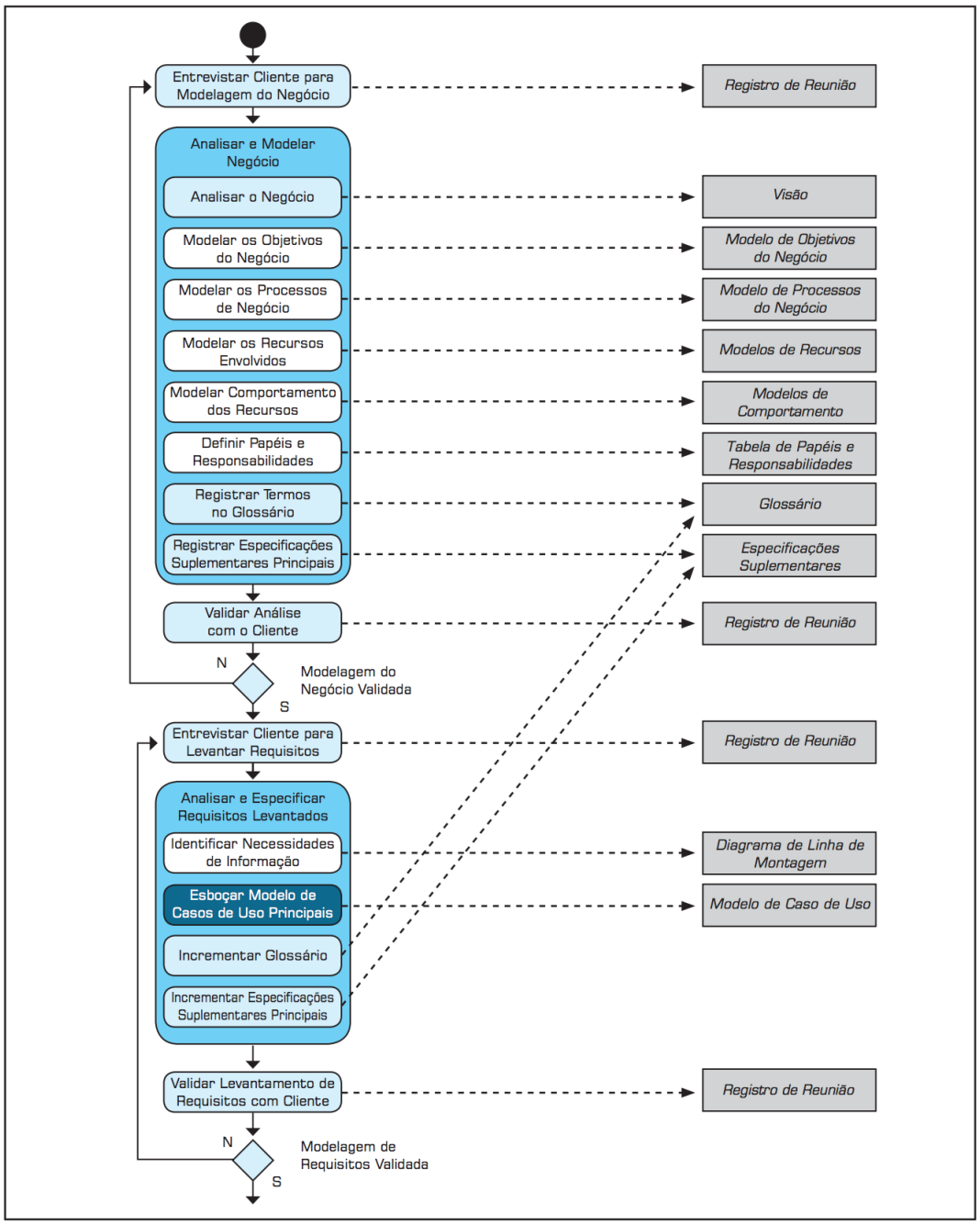


Figura 2.8: Modelo de etapas do levantamento de requisitos.

2.15 UML

Segundo Marlon Dumas da Universidade de Tecnologia de Queensland [33], diagramas de atividade UML (*Unified Modeling Language*) servem para se modelar tanto processos computacionais quanto organizacionais em processos de trabalho. Apesar de não sei um método de desenvolvimento na sua construção, o UML dá todo o apoio e base que um projeto precisa para que sua equipe, ao visualizar sua documentação, consiga ter uma noção geral do *design* e do objetivo final determinado.

O UML consiste de basicamente três itens, sendo eles: diagrama de atividades, diagrama de casos de uso e diagrama de classes.

2.15.1 Diagrama de atividades

O diagrama de atividades representa o fluxograma de como as atividades de um sistema operam e transacionam entre si. Analogamente, o diagrama de atividades segue uma estrutura de compilação procedural se for comparada a uma linguagem de programação. Esse diagrama em especial consegue explicitar visualmente os caminhos que o sistema toma ao longo da sua execução [34].

2.15.2 Diagrama de casos de uso

Quando um cliente especifica o que deseja que quer que seja construído, ou seja, quando a implementação do projeto ainda não foi iniciada e está na fase de levantamento de requisitos, imagina-se todos os possíveis cenários da ferramenta a ser desenvolvida. Esse diagrama é um dos mais importantes do ponto de vista de um planejamento empresarial, dado que uma especificação mais clara do escopo do projeto consegue delimitar a quantidade de trabalho a ser desempenhada naquela situação. Definir claramente o escopo do projeto é o trabalho mais importante desse diagrama e seu uso e registro - se bem feitos - não deixam margem para que fábricas de *software* se aproveitem e lancem horas extra com funcionalidades não requisitadas e que podem até implicar em um uso dificultado da aplicação [34].

2.15.3 Diagrama de classe

Os diagramas de classe já têm uma característica bem mais técnica do ponto de vista de desenvolvimento de *software*. Esse diagrama realmente descreve as classes, funções, métodos e como estes se relacionam com os atributos. De maneira alguma esse tipo de documentação fornece uma visão geral do projeto. Em desenvolvimento de *software* é muito comum que os

programadores não tenham uma noção do produto final, mas sim de módulos que ele e sua equipe estão entregando [34].

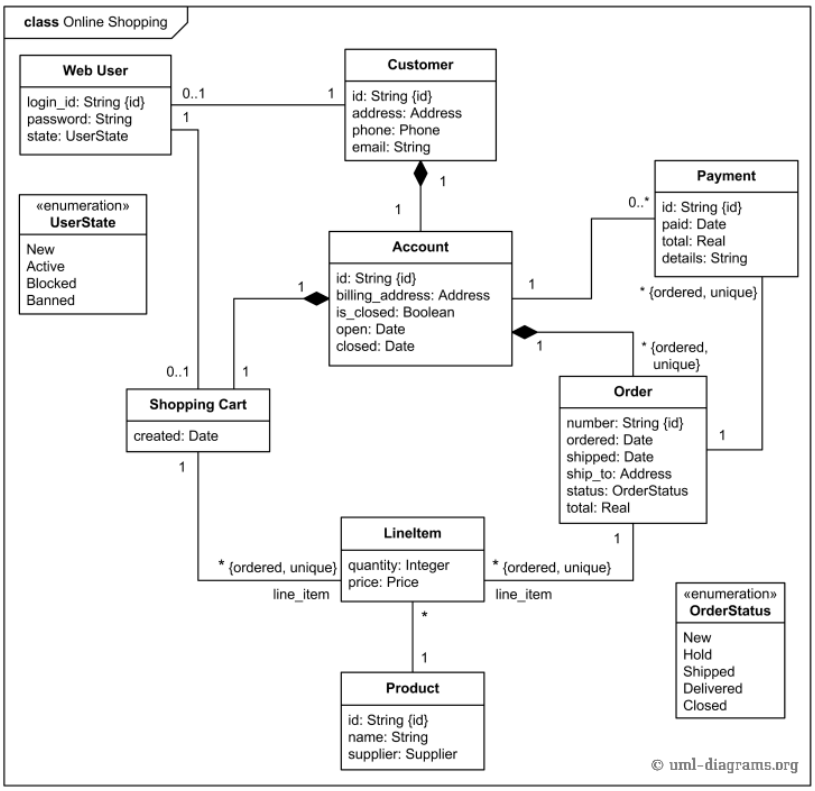


Figura 2.9: Exemplo de diagrama de classe para um sistema de compra online.

Na figura acima pode-se observar o relacionamento das classes de um sistema de compras online [33]. Esse nível de apoio visual contribui para uma maior coesão da parte dos desenvolvedores, tendo em vista que aquele determinado projeto ou módulo tem uma estrutura bem desenvolvida e sem pontas soltas.

2.16 Business Intelligence e Analytics

O Business Intelligence surgiu como área de estudo importante tanto na área acadêmica quanto comercial, refletindo a magnitude do impacto das aplicações que trabalham com dados e com elas levantam gráficos de desempenho, fazem projeções mais realistas de mercado e outras métricas em geral.

Ainda em 2008, a fase em que a tecnologia estava começando a ter sua ascensão e pegar tração de forma muito mais firme, o então diretor financeiro da Google, Hal Varian, disse que dados estavam ficando mais baratos de se obter e a análise completa esses dados [35]. Mais ainda,

Hal Varian deu a recomendação para que as pessoas estudassem mais ferramentas de análise de dados: banco de dados, *machine learning*, econometria, estatística, dentre outros relacionados à área. O *McKinsey Global Institute* publicou em 2011 uma projeção de que em 2018 os Estados Unidos teriam um déficit de profissionais da área de análise de dados na ordem de 140.000 a 190.000 pessoas [36].

Business Intelligence e Analytics (BI&A) são centrados em dados: desde seu armazenamento, gerenciamento, extração e análise através de ferramentais. O interessante é que de acordo com Chaudhuri [37], grande parte da indústria que utiliza o BI&A utiliza um RDBMS para armazenamento de dados. Isso mostra como as ferramentas se relacionam e os motivos que explicam isso serão detalhados com maior clareza no Capítulo 3. Para lidar com mais intuitividade e precisão, um analista de dados conta com ferramentas ETL - que fazem extração, transformação e carregamento (ETL - *extraction, transformation, load*) - em seu dia a dia. Muitos conceitos novos se ramificam a partir da necessidade de se extrair informações de uma coleção de dados, como *big data*, *big data analytics*. Por isso, muitas vezes conceitos podem parecer repetidos ou que estão sendo mesclados com outros.

A figura abaixo ilustra como a evolução do BI&A contribuiu para novos estudos e metodologias de abordagem em relação à utilização de dados, além de exemplos de aplicações e cenários em que é possível se obter esse dados. Por exemplo, plataformas de BI&A 2.0 que são baseadas em formas web podem coletar dados referentes a clientes que fazem compras online em determinado fornecedor, gerando uma necessidade de se desenvolver técnicas mais aprimoradas para este segmento - daí surge o *web analytics*. Essas ferramentas e metodologias aplicadas ao mercado se completam e dão sentido às ramificações que surgem, descritas a partir do BI&A e seus segmentos [38].

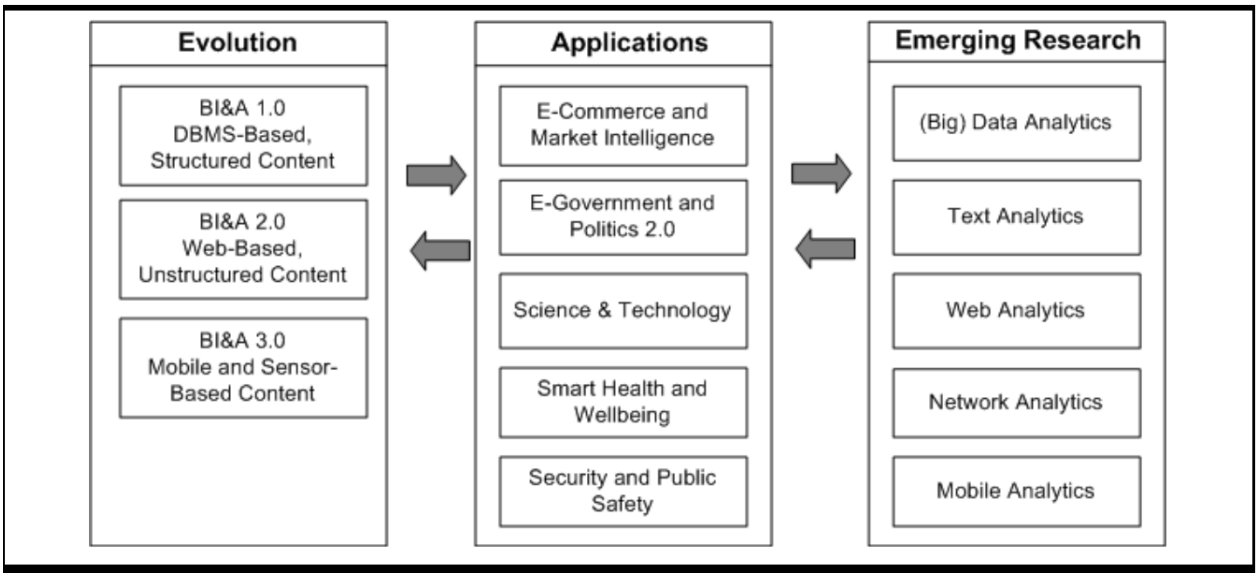


Figura 2.10: Quadro evolutivo do BI&A e suas aplicações de acordo com pesquisas.

A grande expansão do estudo e aplicação do BI&A foi indiscutivelmente iniciada e impulsionada pela indústria do *e-commerce*. No caso deste projeto, o foco é em soluções governamentais e empresas que fornecem serviços para o governo. É uma prática comum da política brasileira que se façam obras corridas perto de épocas de eleição. Sabendo-se disso, é de suma importância garantir que, mesmo com um prazo curto de implementação das obras, se tenha uma estrutura organizada de gestão e controle de recursos. Ter controle e um cronograma flexível de execução dentro de um projeto normalmente necessita de um alto grau de organização e uma gerência firme. Trazendo essa ferramenta de BI&A, pode-se ter uma maior liberdade no que diz respeito ao controle e registro de atividades sendo atualizados a todo momento. Uma ferramenta de BI&A pode ser colaborativa e facilitar o trabalho de todos os envolvidos.

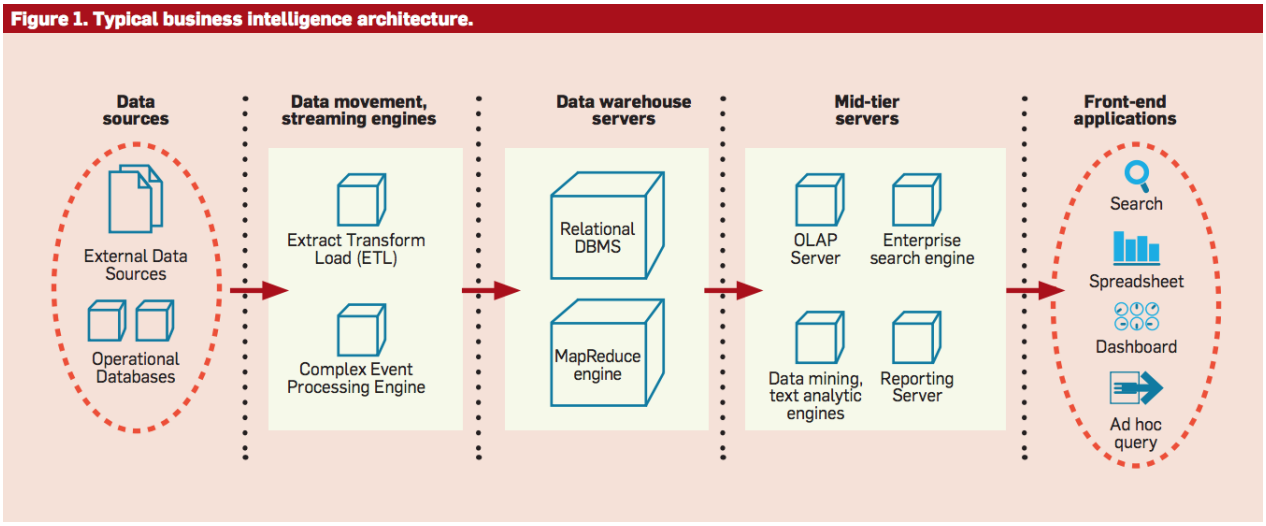


Figura 2.11: Arquitetura típica de um sistema de BI&A.

De acordo com a figura acima [39], para o caso deste projeto, a ferramenta ETL se resume ao aplicativo Android de parametrização de buracos, que captura todas as informações necessárias para serem armazenadas de forma estrutural no RDBMS. Um *mid-tier server* rodando no computador local irá realizar a comunicação com o *front-end* para disponibilizar os dados em um *dashboard* com todas as métricas definidas.

2.16.1 BI&A 1.0

Empresas do varejo e que precisam ter uma relacionamento mais próximo com o cliente passaram a obter dados e tipicamente utilizam bancos de dados relacionais (RDBMS) para armazenamento [38]. As técnicas analíticas desenvolvidas são baseadas em uma matemática estatística muito avançada e baseada em modelos da década de 1970, como o *data mining*.

O BI&A 1.0 é centrado em obter dados e armazená-los para que, quando essa coleção seja grande o suficiente, uma análise matemática possa ser coerente e eficaz. Esse cenário precisa de

uma força tarefa na obtenção desses dados que talvez seja pouco interessante do ponto de vista de escala. Além de requerer agentes na coleta dos dados (cadastrando clientes no supermercado por exemplo), não existe uma automação alta para análise. Todo esse processo de se validar ideias e metrificar o que se foi feito precisa de técnicos, matemáticos e estatísticos utilizando um *dashboard* para visualização [38].

	Key Characteristics	Gartner BI Platforms Core Capabilities	Gartner Hype Cycle
BI&A 1.0	DBMS-based, structured content • RDBMS & data warehousing • ETL & OLAP • Dashboards & scorecards • Data mining & statistical analysis	• *Ad hoc* query & search-based BI • Reporting, dashboards & scorecards • OLAP • Interactive visualization • Predictive modeling & data mining	• Column-based DBMS • In-memory DBMS • Real-time decision • Data mining workbenches
BI&A 2.0	Web-based, unstructured content • Information retrieval and extraction • Opinion mining • Question answering • Web analytics and web intelligence • Social media analytics • Social network analysis • Spatial-temporal analysis		• Information semantic services • Natural language question answering • Content & text analytics
BI&A 3.0	Mobile and sensor-based content • Location-aware analysis • Person-centered analysis • Context-relevant analysis • Mobile visualization & HCI		• Mobile BI

Figura 2.12: Características chave do BI&A 1.0 na primeira coluna da esquerda para a direita.

2.16.2 BI&A 2.0

O BI&A 2.0 já foi uma evolução natural do 1.0 porque tem como base o crescimento em ritmo acelerado da era web. Realmente existiu uma revolução cultural desde os anos 2000 com o advento de novas tecnologias e ferramentas voltadas para usuários comuns. Duas ferramentas foram essenciais para a construção dessa nova ramificação do BI: Google e Yahoo. As ferramentas de busca na internet substituíram a forma como as pessoas passaram a procurar por serviços e produtos. Se antigamente as famosas páginas amarelas eram a maneira de se divulgar uma empresa com panfletos e impressões pagas, agora o mundo passa a utilizar uma nova vitrine - a vitrine virtual.

Só no primeiro trimestre de 2017, a Google faturou U$21.4 bilhões em propagandas online [40]. Fica claro com esse dado que a indústria realmente engatou no modelo de propaganda através da web para tentar fazer o *onboarding* ou retenção de clientes, abandonando o marketing offline nesse sentido.

Técnicas para conhecer e saber mais dos clientes surgiram, como registros de ações em sites através de *cookies*, *heatmap* das ações do usuário em um site, *server logs*, dentre outras. Faz

sentido então que surjam novas técnicas de análise, como o *web intelligence* e o *web analytics*, bem como pesquisas que começaram a ser desenvolvidas baseadas em um contexto web [41]. O BI&A 1.0 é centrado nos dados em sua essência, diferentemente do 2.0 que é centrado em texto, *web crawling*, *web mining*, análise de redes sociais e utilizando as ferramentas estatísticas já utilizadas no 1.0.

Esse é, então, um novo mercado criado a partir da necessidade de se entender melhor como o cliente consumia, como consome e como gostaria de consumir produtos e serviços. Essa provavelmente foi a grande evolução que se teve do BI&A 1.0 para o 2.0 - tirar o foco da análise dos dados de consumo já existentes para a análise do que o mercado quer consumir. Evidencia-se que projeções mais realistas e claras de consumo são apresentadas em relatórios de grandes consultorias como resultado direto dessa nova abordagem de interação [36].

2.16.3 BI&A 3.0

A revolução da web tomada a partir dos anos 2000 rapidamente tomou um novo formato: dispositivos *wireless* e portáteis. Um artigo da *The Economist* em 2011 ressaltou que o número de *smartphones* e *tablets* (480 milhões de unidades no ano) havia ultrapassado o número de *laptops* e PCs (380 milhões de unidades no ano) pela primeira vez na história [42]. Apesar do número de PCs ter ultrapassado a barreira de 1 bilhão de unidades já em 2008, esse mesmo artigo projetou que o número de dispositivos móveis conectados até 2020 iria ultrapassar a margem de 10 bilhões. Essa é, então, uma era de jogos digitais em aparelhos portáteis, de redes sociais e aplicativos embarcados em equipamentos sofisticados e com processamento tão rápido quanto computadores medianos.

Áreas como *mobile design*, UX e UI ganharam destaque. As pessoas hoje em dia estão pagando serviços como *Netflix*, *Spotify* para trazer uma *ad-free experience*. Ou seja, essa revolução cultural que a tecnologia trouxe também veio com um novo modelo de marketing - agressivo e que quer se impor a todo momento para o consumidor e as pessoas estão pagando por serviços que tiram completamente a propaganda e permitem um experiência *ad-free* e *on-demand* completa.

Com tantas ferramentas de gerenciamento de pessoas, recursos, ações e projeções, o BI&A 3.0 - apesar de ainda estar sendo uma área estudada e explorada empiricamente - traz um viés de *cross-platform* para que as informações de diferentes fontes que foram extraídas em pequenos blocos possam se complementar e unir as informações em uma grande central. Essa é a ideia mais geral, mas ainda se tem muito o que ganhar e diferentes arquiteturas podem aparecer.

2.17 Tomada de decisão

O último passo dentro de uma cadeia de processos analíticos em uma empresa retorna à tomada de decisão. Todos os processos estruturados, desde como se obter dados, armazenamento, modelagem e processamento, têm como objetivo chegar no que importa de verdade, decidir um rumo, um posicionamento e estratégia que a empresa deve tomar em relação ao que se observou nos estudos de dados coletados.

A tomada de decisão é precedida pelo processo de análise. Essa fase deve se voltar ao conceito principal de toda operação: reconhecer com clareza qual a problemática ou propósito principal de acordo com os objetivos inicialmente definidos. Por exemplo, se uma empresa do varejo decide analisar os dados coletados sobre a venda de sorvete por período no ano, ela pode perceber que durante o inverno existe uma menor demanda em relação ao verão. Esse dado pode culminar na decisão de se estocar menor quantidade do produto durante o inverno, de forma a economizar valores absurdos com refrigeração e manutenção de equipamentos, bem como recursos humanos responsáveis pela logística de entrega e recepção. Esse foi um exemplo bem lógico e simples, mas muitas vezes em um ambiente e estrutura mais complexos, correlacionar essas variáveis pode ser uma tarefa muito mais complicada.

2.18 Data mining

Após apresentar o panorama do que é o BI&A, é muito importante se entender como e de onde as informações sobre as pessoas, locais, padrões estão sendo extraídas. Com conjunto de dados cada vez maiores sendo disponibilizados para domínio público e privado, observa-se um padrão de previsão muito mais preciso para determinados eventos.

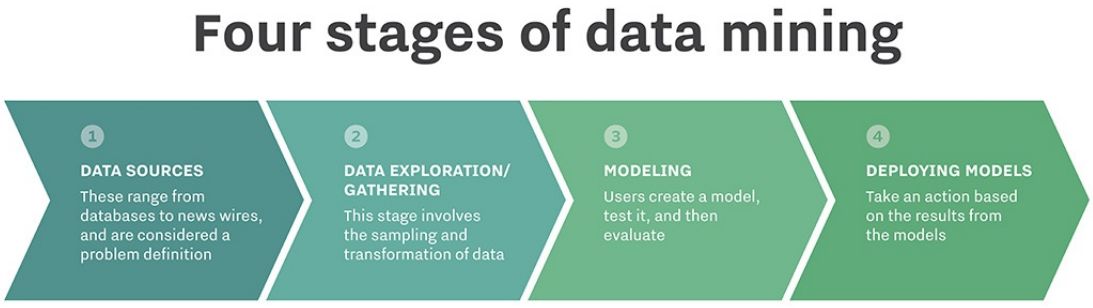


Figura 2.13: Processo de Data Mining. Fonte: Dragon1

Pesquisas sobre como e onde fazer a mineração de dados sobre comportamentos, hábitos, interesses, produtos e serviços em destaque estão no centro da atenção de agências de marketing, de

corporações e gigantes da tecnologia, bem como o varejo. O mercado se adaptou a ser inovativo e estar sempre à frente do consumidor em relação a ele mesmo [43]. Para conseguir tal feito, muitas ferramentas de ETL estão sendo utilizadas em redes sociais. O melhor exemplo de caso é o Facebook, que atingiu o número de 2 bilhões de usuários ativos mensalmente [44]. Uma rede social dessa dimensão e que tem um ambiente apropriado para coleta de dados - possui API de cadastro e *login* que praticamente qualquer plataforma com o mínimo de bom senso utiliza - é a mina de ouro para as empresas.

Segundo Nettleton do Departamento de Ciência da Computação da Universidade de Barcelona [45], a melhor representação de forma organizada da coleta de dados em redes sociais é através de grafos. Os usuários podem ser representados através de nós contendo uma série de informações como nome, idade, email, interesses, histórico do que fez, amizades, dentre outros.

Quando se fala em *data mining* em um contexto web, já se fala em *web mining*. Um mercado gigantesco de *ads* e ferramentas de SEO (*Search Engine Optimization*) se formou [40] e visa auxiliar o aumento da visibilidade orgânica e paga dos *links* de empresas, companhias, governos que queiram aparecer melhor classificadas na ordem de busca sobre determinadas palavras-chave.

Capítulo 3

Metodologia

Este capítulo elucida todo o planejamento e implementação da ferramenta de BI&A para apoio à tomada de decisão em ambientes de manutenção viária e rodoviária.

3.1 Projeto sucessivo

Este projeto sucede o Projeto Final de Graduação "Aplicativo Mobile de Parametrização e Precificação de Reparos de Buracos Viários no DF"[2]. Utilizando o aplicativo desenvolvido no supracitado como um ETL, a ideia é gerar um volume de dados grande o suficiente para obter métricas como por exemplo os gastos que se têm por período, regiões mais afetadas pelo desgaste do pavimento, padrão de uso do aplicativo dos usuários, dentre outros.

O foco, então, ficará sobre o desenvolvimento, manutenção e suporte a todo *back-end* da aplicação ETL. Além disso, utilizando um servidor *mid-tier* local para processamento dos dados será possível modelar métricas e disponibilizá-las em um painel BI&A no *front-end*, que é o produto final deste projeto.

O intuito é que exista uma equipe de análise de dados para estar utilizando o sistema de forma a dar apoio à tomada de decisão - seja financeira, estratégia ou de gestão. Serão disponibilizados dados como bairros mais afetados por degradação das vias, um mapa que consegue trazer a informação visual de onde estão sendo reportados os buracos, o número de buracos reparados, um painel para visualizar agendamentos, apresentar o histórico de manutenção, mostrar a quantidade de horas, recursos humanos e materiais gastos por períodos. É possível também trabalhar com um sistema de *ticket*, em que usuários podem entrar em contato e os agentes responderem por um

sistema interno. Essa transparência é essencial para uma melhor gestão em todos os aspectos, já que o foco são as vias e rodovias mantidas com recursos públicos.

Figura 3.1: Fluxograma das etapas de desenvolvimento do projeto. (Fonte: autor)

Para efeito de facilidade na linguagem, define-se neste projeto que o responsável pela utilização, manipulação e distribuição dos dados e suas consequências é chamado de **gerente de projeto**.

3.1.1 Levantamento de requisitos e definição da arquitetura

Para o levantamento de requisitos de sistema e operação, o ideal realmente seria ter uma reunião ou até mesmo passar um período de tempo observando a forma como agências como a Novacap trabalham. Dessa forma, se poderia entender a melhor forma de otimizar os processos que já são institucionais e conhecidos dentro da cultura de trabalho do órgão. Porém, este projeto traz uma visão do contexto social e que foge do padrão já existente, podendo atingir uma disruptura do sistema de gestão que pode ser aplicado a outros órgãos públicos no futuro dependendo do nível de sucesso atingido. Grandes nomes da indústria surgiram de forma disruptiva e sem ter um *know-how* sobre a área de atuação que stão entrando. Um exemplo disso é Elon Musk, que

ajudou a fundar a PayPal no início dos anos 2000 [46] sem ter uma formação na área de bancos, investimentos ou pagamentos. Anos depois ele fundou a Tesla [47] e a SpaceX [48], duas gigantes nas áreas automotiva verde e espacial, respectivamente - todas totalmente descorrelacionadas e pode-se dizer que não entender do negócio pode ter sido um impulsionador do sucesso dessas companhias por conta da visão e perspectiva de negócio de fora.

Do ponto de vista da arquitetura do sistema, trabalha-se com duas frentes principais. A primeira é um módulo de comunicação entre a aplicação Android e o banco de dados, de forma a fornecer a base para alimentar informações. Já a segunda frente seria realmente extrair esses dados e fazer a modelagem dos dados para apresentação no *front-end* para o usuário do sistema. A arquitetura de BI utilizada é a mesma apresentada anteriormente na Figura 2.6 e a arquitetura simplificada geral é apresentada na figura a seguir.

Figura 3.2: Arquitetura simplificada da ferramenta de BI&A. (Fonte: autor)

A aplicação para extração dos dados neste projeto é um aplicativo desenvolvido em Android nativo e coleta as seguintes informações do usuário e as armazena em uma tabela chamada *user*: nome, email, altura e tem um ID atribuído para controle.

```
marcosguo — mysql -u root -p — 80×24
~ — mysql -u root -p
Database changed
mysql> show tables;
+--------------------+
| Tables_in_pothole |
+--------------------+
| photos             |
| user               |
+--------------------+
2 rows in set (0.00 sec)

mysql> describe user
    -> ;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| nome   | varchar(80) | YES  |     | NULL    |                |
| senha  | varchar(24) | YES  |     | NULL    |                |
| email  | varchar(30) | YES  |     | NULL    |                |
| altura | float(4,2)  | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql>
```

Figura 3.3: Tabela de informações do usuário. (Fonte: autor)

O usuário então alimenta uma tabela chamada *photos*. Ela contém o conjunto de informações sobre os buracos em vias públicas e é descrita na figura abaixo através de: id do usuário,

um nome randômico para a foto, o *path* (ou caminho de registro da foto), a latitude, longitude e o custo de reparo do buraco. Assim como o usuário, é atribuído um ID único para cada buraco.

```
mysql> describe photos;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_id   | int(11)      | YES  |     | NULL    |                |
| name      | varchar(60)  | YES  |     | NULL    |                |
| path      | varchar(200) | YES  |     | NULL    |                |
| latitude  | float(10,6)  | NO   |     | NULL    |                |
| longitude | float(10,6)  | NO   |     | NULL    |                |
| cost      | float(10,2)  | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
7 rows in set (0.00 sec)
```

Figura 3.4: Tabela de informações do buraco. (Fonte: autor)

Essas informações são comunicadas pela aplicação e armazenadas no servidor rodando em LAMP. O interessante seria conseguir rodar essa *stack* em um servidor dedicado e durante a implementação deste projeto foi feita uma abertura de pedido para instalar o servidor dentro do Laboratório de Tecnologias da Tomada de Decisão, um dos laboratórios da Universidade de Brasília.

A comunicação entre a aplicação e um servidor local LAMP foi estabelecida com sucesso já de antemão ao início deste projeto, mas está dentro do escopo de desenvolvimento do mesmo. Após essa primeira parte da arquitetura estar funcional, pode-se começar a trabalhar com a extração e modelagem dos dados obtidos. Para isso utiliza-se um servidor que possui permissão de leitura ao dados e roda *scripts server-side* e já dá o *output* fazendo uma conexão direta com o *front-end*. Tanto o PHP quanto o Ajax foram usados com essa finalidade na modelagem, com o Javascript, HTML e CSS trabalhando para fornecer uma formatação intuitiva e de fácil visualização para o usuário no *front-end*.

3.1.2 Alimentar informações no banco de dados

O intuito do projeto predecessor realmente foi de trazer um colaborativismo por parte dos usuários, auxiliando tanto os próprios usuários a ficarem atentos quanto do seus trajetos pela cidade quanto para quem cuida da manutenção e saúde das vias tenham uma ferramenta visual quanto dos locais que precisam de atenção. No entanto não foi feito um trabalho no sentido de se popularizar o aplicativo para uso em massa por dois motivos; primeiro o tempo para executar e conscientizar as pessoas iria tomar muito tempo e fugir do escopo do primeiro projeto; o outro motivo é que a licensa para colocar um aplicativo na Google Play (loja oficial de aplicativos Android) tem um custo de U$25 [49] e um tempo de aceite dentro da loja até esteja disponível para *download*.

Uma outra opção seria fazer uma distribuição do APK do aplicativo, que analogamente seria como um instalador de extensão .exe para distribuições binárias. Essa divulgação seria feita

em grupos da Universidade de Brasília em redes sociais, porém iria exigir um nível médio de manipulação do celular para permitir a instalação e muitos usuários se sentiram desinibidos para efetuar tal ação. Dessa forma, o que se fez foi mapear através do Waze [50] os buracos dentro da região do Plano Piloto e entorno. Com esse mapeamento em mãos, passou-se por esses locais e foi utilizado o aplicativo para alimentar a base. Esse realmente foi um trabalho exaustivo e pouco eficiente. Contudo, não havia tempo hábil para se distribuir o APK e esperar que os usuários reais do aplicativo o utilizassem para coleta de dados.

3.1.3 Extração e modelagem de dados

A extração dos dados armazenados no banco de dados é feita através de *server-side scripting*, principalmente com o uso de PHP e uso de Ajax em algumas ocasiões (apesar de Ajax ser considerado um tecnologia orientada ao *client-side* assim como o Javascript). O PHP é responsável pela grande maioria das *queries* (se não todas) do banco - que é o MySQL Community Server GPL Versão 5.7.20 [51].

Da modelagem de dados, são usadas diversas funções em PHP que vão fazer o tratamento do *output* para serem chamados por um HTML. No caso dos gráficos, foi utilizada a ChartsJS [52] que possui suas próprias funções de controle de *input* (dados puros do banco) e *output* (saída da função).

O *snippet* do código abaixo faz parte do projeto, servindo para diversos propósitos dentro do painel BI&A, dentre eles fornecer dados para a API do Google Maps disponibilizar a localização dos buracos e também mostrar a localização dos buracos na conta do aplicativo Android por usuário. O *connect.php* requerido já na primeira linha do código é um *script* que tem uma única funcionalidade: dar permissão de conectividade para leitura do banco para outros *scripts* que tiverem uma dependência dele.

```
<?php


 require("connect.php"); // Permissao de acesso 777 ao banco de dados


$mysql_qry = "select latitude, longitude, cost from photos where user_id like
    '$user_id';"; // Faz a query na variavel mysql_qry para ler o dado de latitude,
    longitude e cost da tabela photos
```

```
if (mysqli_multi_query($conn,$mysql_qry)) // Verifica conexao
{
   $superstring = "sucesso \n"; // Variavel para concatenar os resultados
  do
    {
    if ($result=mysqli_store_result($conn)) {

 // Varre todas as linhas da tabela photos
      while ($row=$result->fetch_assoc())
        {
        $superstring .= $row['latitude'].' '.$row['longitude']. " \n";
        }
      mysqli_free_result($result);
      }
    } while (mysqli_next_result($conn));
  }
  echo $superstring; // Resultado com a localizacao

?>
```

O retorno dessa função foi feito para o aplicativo Android, que pega a primeira palavra da $superstring que é "sucesso"para o caso de ter feito a *query* corretamente e entende que o comportamento e saída foi adequado para o que se esperava da manipulação do dado. Além disso o uso de APIs, como da Google Maps e ChartJS, facilitou muito a implementação do sistema, reaproveitando código e diminuindo a duplicação do mesmo.

3.1.4 Disponibilizar informações no front-end e server-side scripting

Essa parte do escopo do projeto já deve ser tratada com um visão mais funcional de como um analista, técnico ou gerente de projeto perceberia os dados de maneira mais clara e sem deixar brechas à interpreção pessoal. O painel que contém os dados deve ser assertivo e existe

uma preocupação com o *design* - talvez um pouco pela questão estética, mas a prioridade é a simplicidade e exibir as informações de maneira totalmente descomplicada. Esse ponto é crucial para que um novo produto se desenvolva com sucesso - simplificação. Apesar dos esforços que a comunidade de engenharia de *software* faz trazendo modelos e especificações para boas práticas durante a execução do que se levantou no planejamento, um número cada vez maior de projetos que não cumprem os critérios pré-definidos aparece a todo momento [53]. Segundo Joseph Barjis, da Universidade de Tecnologia de Delft, na Holanda, as empresas precisam ter mais atenção e clareza em duas coisas no processo de desenvolvimento de *software*: modelagem dos processos internos daquela instituição e uma linguagem de ação simples. Isso quer dizer que trazer informações de uma forma visualmente clara pode beneficiar uma gestão sobre determinado produto ou serviço.

O *front-end* vem com o intuito de se trazer uma ferramenta que não seja influenciadora de decisões, mas sim um apoio às mesmas. Dessa forma, até o padrão de cores apresentado para o gestor do projeto foi pensado de forma a não tirar a atenção sobre os dados. Até mesmo o uso da fonte *Work Sans* foi pensado para que as linhas não sejam grosseiras. Esse minimalismo dá leveza para tomadas de decisões que geralmente são difíceis e duras. Afinal, são decisões que vão moldar o futuro daquele projeto.

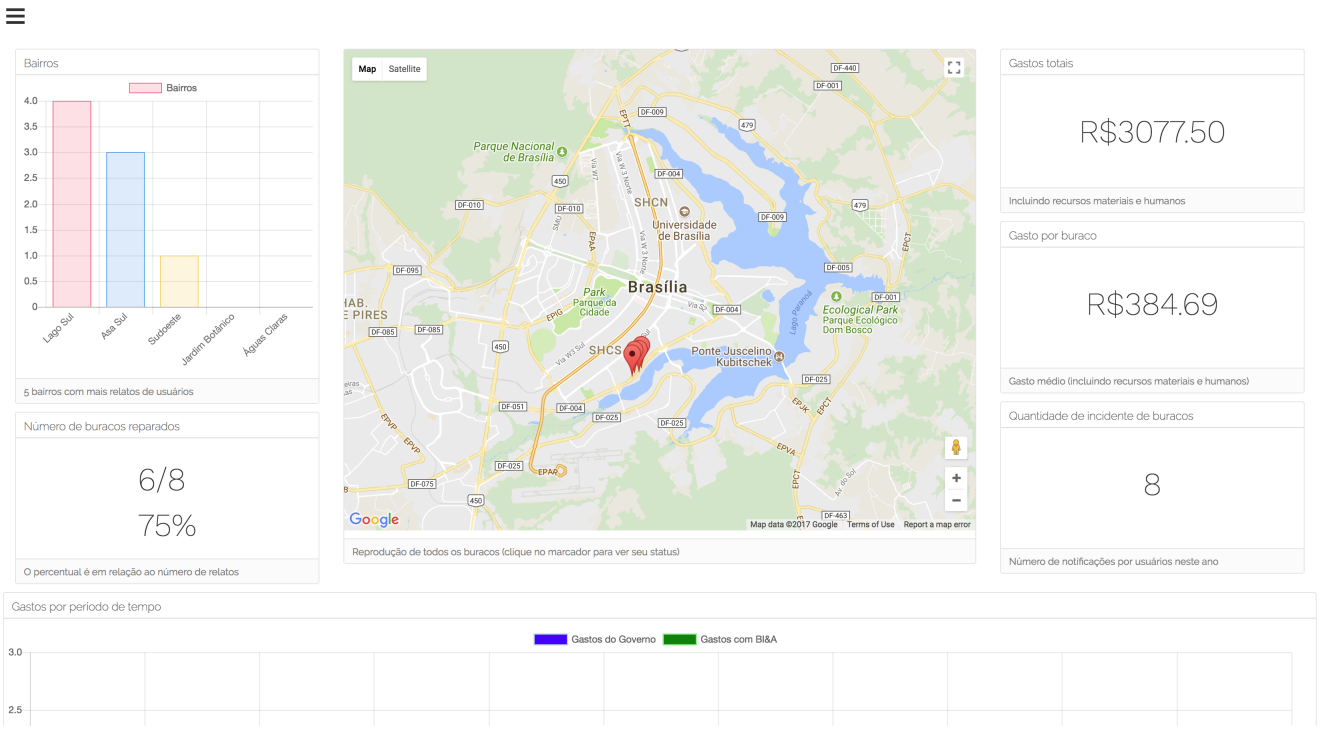


Figura 3.5: Painel de BI&A para suporte à tomada de decisão na manutenção viária do DF. (Fonte: autor)

O que se percebe imediatamente é que os dados são o que entram em destaque já na primeira visualização que se tem do painel. O único destaque em cores fica por conta do mapa

dinâmico feito através da Google Maps API. A localização de cada ponto é pega através do banco de dados com *queries* feitas por PHP e manipuladas para o mapa com Javascript. Aliás, todos os dados presentes no painel são trazidos dessa forma - não necessariamente com JS para o *front-end*, mas utilizando *server-side scripting* com o PHP e manipulando os dados para obter porcentagens, gráficos comparativos de custos notificações por bairros. A apresentação fica por conta do HTML, CSS e JS. A seguir apresenta-se a maneira como se foi feita a disponibilização dos dados.

Bairros: traz um gráfico em barras que mostra a quantidade de notificações criadas por usuários em determinadas regiões, todas dentro do DF no escopo deste projeto. A foto abaixo descreve a tabela *photos* do banco de dados *pothole*.

```
mysql> select * from photos
    -> ;
+------------+---------+-------+-------+------------+------------+---------+-----------+--------+
| id         | user_id | name  | path  | latitude   | longitude  | cost    | bairros   | status |
+------------+---------+-------+-------+------------+------------+---------+-----------+--------+
|  469952655 |      11 | foto1 | /user | -15.827759 | -47.889027 |   10.50 | AsaSul    |      1 |
|  681653864 |       3 | foto1 | /user | -33.840282 | 151.210449 |   11.50 | AsaSul    |      1 |
|  989863604 |       3 | foto1 | /user | -33.840282 | 151.210449 |   11.50 | Sudoeste  |      1 |
| 1259197692 |       3 | foto1 | /user | -33.840282 | 151.210449 |   11.50 | LagoSul   |      1 |
| 1376545851 |      11 | foto1 | /user | -15.829705 | -47.891911 |   10.50 | LagoSul   |      1 |
| 1856441659 |      11 | foto1 | /user | -15.831215 | -47.892982 |   10.50 | LagoSul   |      1 |
| 1958482391 |      10 | foto1 | /user | -15.829409 | -47.890156 | 3000.00 | AsaSul    |      0 |
| 2024404472 |       3 | foto1 | /user | -33.840282 | 151.210449 |   11.50 | AsaSul    |      0 |
+------------+---------+-------+-------+------------+------------+---------+-----------+--------+
8 rows in set (0.00 sec)
```

Figura 3.6: Descrição da tabela que contém informações acerca dos bairros com maior número de notificações de usuários (Fonte: autor)

É feita uma *query* na coluna de bairros e armazena-se todas as *strings* obtidas em um *array*. O *script* **bairros.php** se encarrega disso e também de fazer o *encoding* do *array* em um formato JSON, que é necessário no JS **bairros.js**. Este último fica, então, responsável por pegar o dado no formato JSON disponibilizado pelo *server-side scripting* através do método GET (HTTP) e utiliza então a API ChartJS para fazer a chamada da função de representação gráfica. Fez-se um tratamento de erro para disponibilizar o gráfico com os dados mesmo que se envie uma *string* incorreta (sendo esta descartada) ou se não existir um buraco em determinada região, o que é bom senso mas não foi tratado pela API gráfica ChartJS. O resultado da *query* feita pelo *server-side scripting* em PHP resulta em um *array* no seguinte formato:

Array ([0] => Array ([bairros] => AsaSul) [1] => Array ([bairros] => AsaSul) [2] => Array ([bairros] => Sudoeste) [3] => Array ([bairros] => LagoSul) [4] => Array ([bairros] => LagoSul) [5] => Array ([bairros] => LagoSul) [6] => Array ([bairros] => AsaSul) [7] => Array ([bairros] => AsaSul))

A partir desse *array* que pega a coluna **bairros** da tabela **photos**, é necessária uma contagem de cada um dos elementos que resulte em algo do tipo:

[{"AsaSul":"4"},{"LagoSul":"3"},{"Sudoeste":"1"}]

Esse formato acima é um formato JSON válido e a leitura do arquivo **bairros.js**, res-

ponsável por organizar o número de buracos por bairro e disponibilizá-los em forma gráfica.

O dado de **número de buracos reparados** também vem essa mesma tabela descrita na Figura 3.6. Dessa vez utiliza-se a informação da coluna *status*. Usando uma lógica Booleana binária para definir se a notificação dos usuários foi atendida ou não, pode-se facilmente pegar a informação do número de buracos que foram reformados. Se definiu que um buraco com *status* 0 não foi reparado e com *status* 1 já foi feito o reparo. Dessa forma, é feita uma contagem nessa coluna de forma algébrica, esquecendo da lógica Booleana previamente pensada. O resultado final dessa soma evidentemente resulta no número de buracos que se deu suporte. Para trazer um pouco mais de contexto à informação, foi adicionado um contador no mesmo código **quantidadebairros.php** que faz essas *queries* do número de buracos consertados. Esse contator simplesmente varre a tabela contando o número de linhas, que na prática se refere ao número total de notificações geradas pelos usuários - o número total de buracos. Claro, múltiplos usuários podem reportar um mesmo buraco e esse painel ainda não foi especificado para tratar de quão próximo uma notificação precisa estar para que aconteça um *merge* entre elas e serem consideradas um único buraco. Essa é uma inteligência muito importante para trabalhos futuros. A informação da quantidade é, então, disponibilizada na forma de fração, com o numerador indicando o número de reparos e o denominador o número total de buracos notificados. Finalmente, essa fração é também mostrada em formato percentual para um cálculo rápido de como se está gerindo no momento a saúde das vias.

O **mapa de buracos** possui uma lógica bem mais complexa que a maioria das informações até então. São basicamente três etapas: *query* do banco para pegar as informações de latitude de longitude, um tratamento dos resultados dessa *query* para passar os dados para o formato XML e, finalmente, a chamada da API do Google Maps Javascript [54]. Explicando cada etapa detalhadamente, a *query* é feita através do *script* **xml.php** que também inclui o segundo passo de fazer um *parse* da formatação do resultado para XML. Sem estar no formato XML, a API da Google Maps Javascript não aceita o *input* e, portanto, não disponibiliza os marcadores de localização. Fez-se o *embed* do código em JS a seguir dentro do próprio HTML da página. Isso não é uma prática recomendada dentro de desenvolvimento de *software*, porém esse *snippet* não faz nenhum tipo de carregamento em tempo real, não precisa de nenhum tipo de função de *callback* e é rodada apenas uma vez durante o carregamento da página. Isso em questão de custos de servidor para uma empresa é algo muito interessante, dado que não é necessário que se faça muitas requisições a todo instante de forma a minimar os custos nesse sentido. Para se ter uma ideia, a AWS (Amazon Web Services) que é um provedor de plataformas de *cloud computing on-demand*, teve um crescimento trimestral enorme nesse ano de 2017. Isso significa que cada vez mais os provedores de serviços e produtos estão automatizando seus sistemas e utilizando cada vez mais requisições aos seus servidores [55].

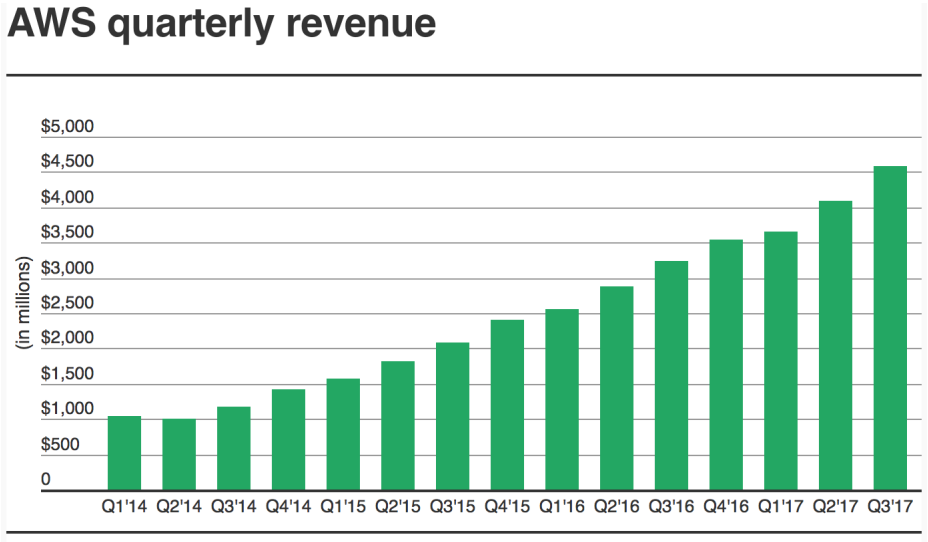


Figura 3.7: Relatório trimestral de receita da AWS (Fonte: TechCrunch)

O Ajax faz a atualização dos gráficos sem precisar atualizar a página toda. Isso é benéfico no sentido de utilização de *queries*, mas não é tão eficiente porque não tem uma taxa de atualização definida. O ideal seria ter uma capacidade de servidor que permitisse que as novas informações sempre entrassem sem que se precisasse fazer a requisição manualmente. Porém, para efeito de simular uma redução de custos, essa funcionalidade não foi feita. Não que fosse algo difícil de se implementar - se fosse adicionada o *snippet* abaixo dentro do corpo do Ajax em **linegraph.js** e **bairros.js** esse feito já seria realizado nos dois gráficos de gastos por período de tempo e dos 5 bairros mais notificados, respectivamente.

```
$(document).ready(function(){

  refreshTable(); // Chamada da funcao refreshTable() dentro do escopo do JS

});


function refreshTable(){

    $('#tableHolder').load('buracos.php', function(){ // Deve chamar o php que

                                  // faz as queries das info de cada painel

       setTimeout(refreshTable, 5000); // Seta o tempo de atualizacao a cada 5s

    });

}
```

Para os **gastos totais** foi utilizada a mesma lógica: é feita a *query* na tabela **photos** e dessa vez os dados da coluna **cost** que são utilizados. Eles estão no formato float(10,2) para dois dígitos de precisão decimais. É realizada a seguinte operação dentro do mesmo *script* - **gastostotais.php** - que faz as *queries*

$$Total(material) = \sum_{n=0}^{\infty} cost$$

de forma que a variável *Total* armazena o valor em flot(10,2) do somatório de custo de cada buraco. Esse valor é calculado pela aplicação Android [2] e este trabalho apenas aceita o valor de referência que foi preocupação do trabalho anterior. A partir desse somatório já se tem o custo total dos materiais utilizados na recuperação dos buracos, sem contar com a parte de recursos humanos.

Cada integrante da equipe de recuperação e revitalização viária possui um *id* único associado, que descreve também seu cargo (*cargo*), nome (*nome*), o nome do time do qual faz parte (*time*) e a remuneração (*remuneracao*). Essas informações estão descritas na figura abaixo.

```
mysql> describe rh;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| id          | int(11)     | YES  |     | NULL    |       |
| cargo       | varchar(20) | YES  |     | NULL    |       |
| nome        | varchar(30) | YES  |     | NULL    |       |
| time        | varchar(20) | YES  |     | NULL    |       |
| remuneracao | float(10,2) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figura 3.8: Tabela de **rh** descrevendo os recursos humanos do projeto (Fonte: autor)

Nesse sentido, o cálculo que descreve o custo global da operação seria o seguinte:

$$Total(global) = \sum_{n=0}^{\infty} cost + \sum_{n=0}^{\infty} remuneracao$$

Dessa forma, o **gasto médio por buraco** é definido como o *Total(global)* dividido pela quantidade de buracos. Para efeitos práticos, mostra-se o valor médio de cada buraco somente com o custo material e também o custo com recursos humanos. Isso facilita a visualização de quanto a questão humana pesa dentro de uma operação de infraestrutura urbana como esta. O valor de recursos humanos vem da tabela **rh** descrita abaixo.

$$Média(Material) = \frac{1}{n}(\sum_{n=0}^{\infty} cost)$$

$$Média(Total) = \frac{1}{n}(\sum_{n=0}^{\infty} cost + \sum_{n=0}^{\infty} remuneracao)$$

O painel BI&A fica no formato da figura seguinte após essas alterações, incluindo as médias, custos totais e o percentual a mais que os recursos humanos impactam na operação. Apesar de não estar tão mais simples, a visualização ainda é fácil para efeitos analíticos, dado o volume de informações que são totalmente relevantes do ponto de vista operacional e funcional da tomada de decisão.

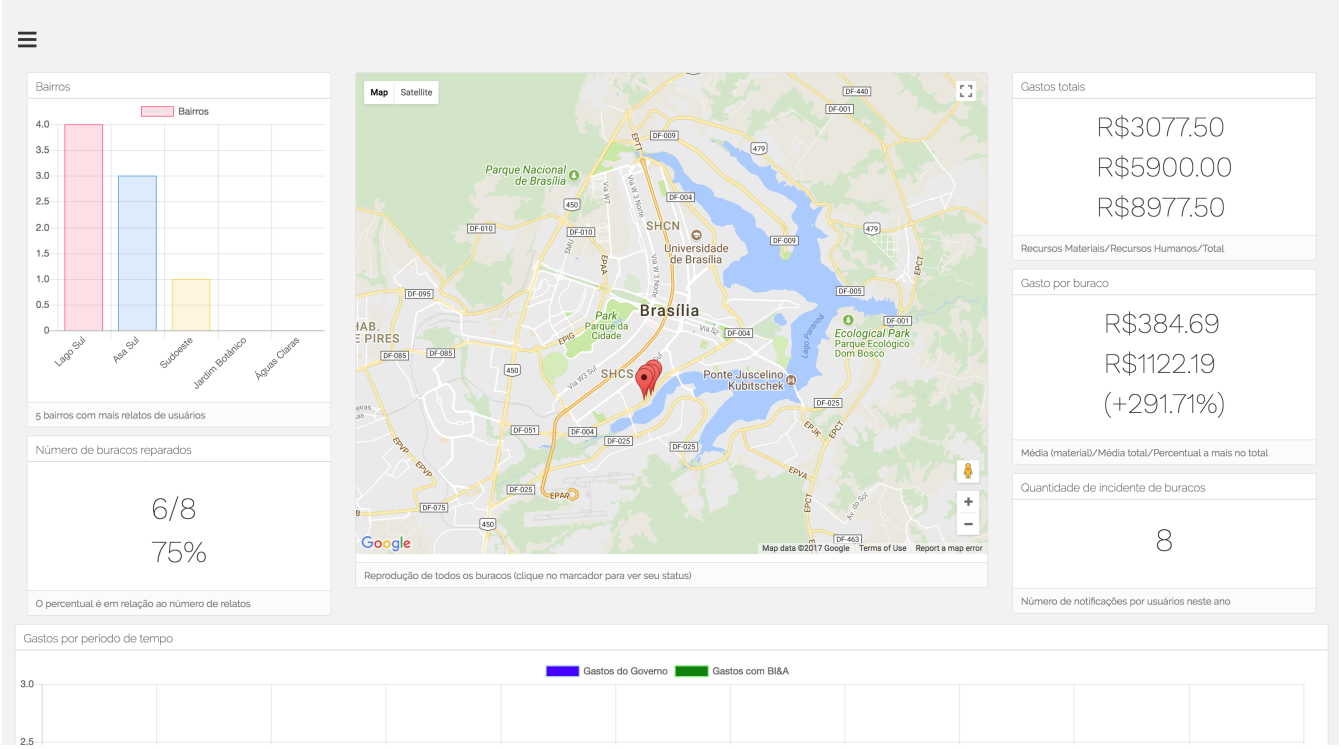


Figura 3.9: Painel BI&A com adição de métricas (Fonte: autor)

Da **quantidade de buracos**, que indica o número de incidentes relatados por usuários, se obtêm a informação simplesmente pelo número de *queries* obtidas da tabela **photos** pelo *script* **quantidadeburacos.php**.

Agora, para o gráfico de linha de **gastos por período de tempo**, a complexidade se dá na atribuição de cada ponto por período. Os valores apresentados como "Gastos do Governo"foram obtidos através de pesquisas e reportagens acerca do gasto governamental terceirizando o serviço para agências como a Novacap [5], além de ter como base o Portal da Transparência [56] que mostra gráficos e dados acerca de despesas e investimento em cada área socio-econômica e órgãos institucionais. O *script* que pega os valores da tabela **financeiro** (descrita abaixo) é o **linha.php**.

```
mysql> describe financeiro;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| governo | float(10,2) | YES  |     | NULL    |       |
| analise | float(10,2) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Figura 3.10: Tabela de **financeiro** descrevendo os gastos do governo e os com suporte à gestão

(Fonte: autor)

Como dito anteriormente, não existe inteligência no apoio à tomada de decisão. São várias informações importantes distribuídas em gráficos e a inferência para suporte na tomada de decisão cabe ao analista, gerente de projeto ou quem quer que seja monitorando o painel. Por exemplo, quando um incidente de buraco é reportado pelo usuário, deve se criar uma regra interna dentro da operação que define um tempo máximo para atendimento daquela solicitação de reparo. Dessa forma, *tickets* de reparo seriam criados e designados às equipes de recuperação viária de forma automática de acordo com a ociosidade fazendo uma distribuição. Além disso, métricas de nível de trabalho e metas entregues poderiam ser criadas para avaliar cada time, de forma a criar sistemas de recompensa e promoções na corporação.

Abaixo está descrito o custo total da operação no DF referente ao mês de janeiro do ano de 2017. As informações são encontradas no site do DNIT [57], porém as informações são disponibilizadas somente até Maio de 2017 [57] e não existe um histórico no *site* para os anos anteriores. Dessa forma, fez-se uma estimativa dos meses de Junho até Novembro de 2017.

DNIT **CGCIT**

SISTEMA DE CUSTOS REFERENCIAIS DE OBRAS - SICRO — Distrito Federal — FIC 0,01606
Custo Unitário de Referência — Janeiro/2017 — Produção da equipe 0,56 m³
4915757 Tapa buraco com serra corta piso — *Valores em reais (R$)*

A - EQUIPAMENTOS		Quantidade	Utilização		Custo Horário		Custo Horário Total
			Operativa	Improdutiva	Operativo	Improdutivo	
E9556	Compactador manual de placa vibratória - 3 kW	1,00000	0,25	0,75	4,5350	1,4090	2,1905
E9591	Serra para corte de concreto e asfalto - 10 kW	1,00000	0,16	0,84	7,7395	0,6064	1,7477
						Custo horário total de equipamentos	3,9382
B - MÃO DE OBRA		**Quantidade**	**Unidade**		**Custo Horário**		**Custo Horário Total**
P9824	Servente	6,00000	h		13,2018		79,2108
						Custo horário total de mão de obra	79,2108
						Custo horário total de execução	83,1490
						Custo unitário de execução	148,4804
						Custo do FIC	2,3846
						Custo do FIT	-
C - MATERIAL		**Quantidade**	**Unidade**		**Preço Unitário**		**Custo Unitário**
M3507	Material retirado da pista - revestimento asfáltico	1,00000	m³		-		-
						Custo unitário total de material	-
D - ATIVIDADES AUXILIARES		**Quantidade**	**Unidade**		**Custo Unitário**		**Custo Unitário**
4900000	Mistura betuminosa	1,00000	m³		-		-
						Custo total de atividades auxiliares	-
						Subtotal	150,8650
E - TEMPO FIXO		**Código**	**Quantidade**	**Unidade**	**Custo Unitário**		**Custo Unitário**
M3507	Material retirado da pista - revestimento asfáltico - Caminhão basculante 6 m³	5915476	2,40000	t	43,4300		104,2320
						Custo unitário total de tempo fixo	104,2320
F - MOMENTO DE TRANSPORTE		**Quantidade**	**Unidade**		**DMT**		**Custo Unitário**
					LN	RP	P	
M3507	Material retirado da pista - revestimento asfáltico - Caminhão basculante 6 m³	2,40000	tkm		5914314	5914329	5914344	
4900000	Mistura betuminosa - Caminhão basculante 6 m³	1,50000	tkm		5914314	5914329	5914344	
							Custo unitário total de transporte	
							Custo unitário direto total	255,10

Figura 3.11: Custo de Jan/2017 da Operação Tapa-Buraco (Fonte: DNIT Centro-Oeste)

Atualmente a forma como a Novacap recebe as notificações e reclamações da população acerca de problemas estruturais nas vias públicas é através da Ouvidoria do Distrito Federal [58]. Como mostrado na figura abaixo, o procedimento é altamente burocrático e, se convém dizer, muito bem escondido. Realmente só o cidadão muito interessado vai ir atrás de informações sobre como proceder e fazer a solicitação de reparo. Após fazer a solicitação, não existe um cronograma em que o agente encaixe aquele pedido - ao trazer a solução do colaborativismo com um sistema mais transparente e de fácil gerenciamento é possível simplificar a vida dos agentes e de quem utiliza as vias públicas.

Figura 3.12: Procedimento para registrar um pedido de reparo viário (Fonte: Ouvidoria do GDF)

A ideia geral então é automatizar todo o processo para que a comunicação não se deteriore e fique perdida entre um registro de manifestação até o gerente das equipes de reparo de determinado dia da semana, por exemplo. Ao trazer essa automação em que os próprios usuários das vias reportam a **localização precisa** e que os gerentes de projeto tenham uma ferramenta que define as datas de reparo em uma tabela que também inclui dados como o dia em que o usuário reportou o problema, o gerente responsável por aquele reparo (que pode ser definido por região por exemplo) e o seu *status*. Isso é um auxílio visual muito poderoso perto do processo que se tem atualmente e a transparência que a ferramenta dá para os dois lados do máquina é essencial na prestação de contas cada vez mais cobrada.

ID	Bairro	Data de entrada	Data de reparo	Responsável	Finalizado
1	Lago Sul	08/10/2017	15/10/2017 Alterar	Georges Alterar	N Alterar
2	Lago Sul	01/11/2017	20/10/2017 Alterar	Jonas Alterar	S Alterar
3	Asa Sul	21/10/2017	05/11/2017 Alterar	Georges Alterar	S Alterar

Figura 3.13: Tabela de agendamentos de reparo (Fonte: autor)

Esse sistema de agenda está implementado em complemento ao sistema de BI&A. Além de existir uma análise gráfica e visual dos dados coletados a partir do colaborativismo que os usuários das vias trazem, o painel possui um sistema inteligente que cria uma entrada para toda notificação de usuário. Pela figura acima, percebe-se que é possível se alterar três entradas de cada agendamento: data de reparo, responsável e o *status* de solicitação atendida ou não. A data de reparo que é definida inicialmente é de 15 dias após notificação. Esse tempo, claro, pode ser ajustado futuramente de acordo com as regras e processos dentro do órgão responsável pelas manutenções. Como não houve resposta acerca do tempo de demora padrão para cada atendimento aos registros de manifestação da Ouvidoria do GDF, definiu-se esse tempo, que foi julgado como suficiente para reparo e que não fosse longo o suficiente para causar danos a longo prazo para quem trafega na região.

Do desenvolvimento, foi necessário se trabalhar com *server-side scripting* e modelagem dos dados para apresentação. O JS fica novamente com o cargo de fazer as chamadas de funções a partir do *front-end* modificar os dados. Primeiramente foi criado um script chamado **agenda.php** que fica responsável por fazer um **SELECT** de todos os dados da tabela agendamentos.

```
mysql> describe agendamentos;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| responsavel | varchar(30) | YES  |     | NULL    |                |
| local       | varchar(20) | YES  |     | NULL    |                |
| dataentrada | varchar(20) | YES  |     | NULL    |                |
| dataagenda  | varchar(20) | YES  |     | NULL    |                |
| finalizado  | varchar(3)  | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Figura 3.14: Descrição dos campos da tabela **agendamentos** (Fonte: autor)

O **agendamento.js** usa Ajax para fazer a chamada de **agenda.php** e faz iterações para coleta dos dados que o *script* envia e os armazena em um formato JSON. Esse formato JSON é então todo destrinchado em chaves chamadas *key values* para que se manipule os dados de cada linha individualmente. Isso é feito através de uma chamada específica no dado: se o dado JSON de todas as linhas de agendamentos está na variável *data*, a chamada da primeiro ID de buraco seria

feita através de *data.[key].id* com leitura na primeira linha.

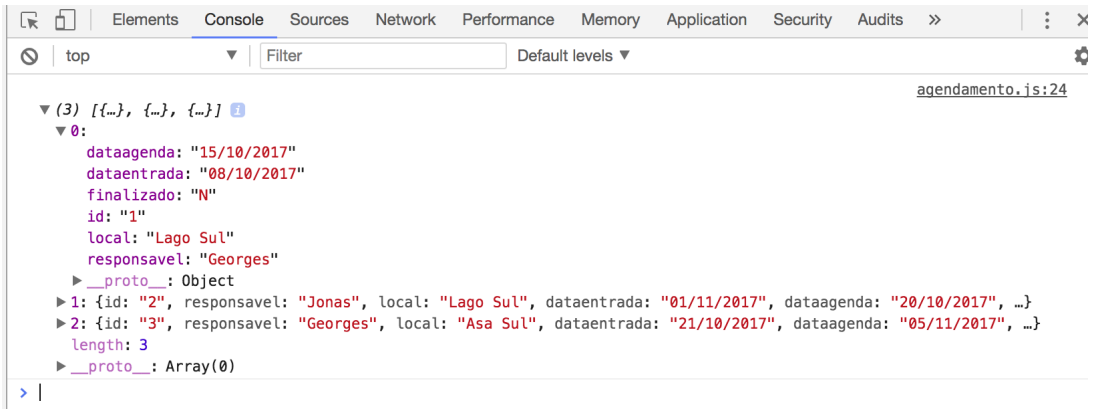


Figura 3.15: Formato de dado que o **agendamento.js** captura do **agenda.php** (Fonte: autor)

Através de um *console.log(data)* é possível ver o *array* na figura acima em formato JSON dos dados. Neste caso são três elementos, cada um discriminado de forma bem organizada e intuitiva tanto para leitura de máquina quando humana - que é a proposta do JSON. Abriu-se o primeiro elemento *0* para visualização e nele pode-se ver todas colunas de entrada da tabela que contém informações dos agendamentos. Com esse grupamento de informações, é necessário então fazer um tratamento visual. Isso é atingido utilizando HTML e CSS. A grande dificuldade aqui foi utilizar o ***Bootstrap Modal*** que é um *plugin* de *dialogs*. O *Bootstrap* é um *framework open-source* para *front-end web*. Ou seja, ele é um *software* que permite a manipulação do usuário para atender a tarefas e utilidades que lhe sejam interessantes [59]. A dificuldade não se faz pelo uso do *framework* em si, mas a forma como os elementos em HTML, CSS e JS precisam se encaixar para que se tenha o funcionamento adequado da plataforma. A organização de código em HTML foi crucial para que a execução fosse correta. Inicialmente se misturou muito código JS dentro do HTML através de *tags* do tipo **<script></script>** e isso gerava comportamentos errôneos por chamadas de *callback* quando não se era necessário e vice-versa. O que se fez então foi retirar *scripts* e JS do HTML e criar vários arquivos individuais, fazendo a chamada de cada um somente quando se fosse necessário. Isso é interessante do ponto de vista de desenvolvimento de código porque diminui muito a duplicação de código. Por exemplo, várias *queries* são realizadas com o mesmo código, então esse código é utilizado para outras chamadas que precisam desse *input* de dados. Um exemplo disso é o conjunto de *scripts* **muda-reparo.php**, **muda-responsavel.php** e **muda-status.php**, que inerentemente realizam o mesmo processo de pegar o novo *input* do *modal* e passar para o *back-end* da aplicação através de um *script* com um **UPDATE** no MySQL.

O processo de ajuste para verificar se o comportamento de determinado *script* e formatação para o *front-end* com o uso de um JS por exemplo estava correto era basicamente utilizar um servidor local para testar se o *output* está correto - seja imprimindo o dado diretamente no *browser*, abrindo *pop-up* ou imprimindo no console do *browser*. No caso deste projeto se utilizou o Google Chrome 62.0.3202.94 (64-bit) por ter uma característica mais universal com 54,53% do

market share de navegadores no mundo [60]. Finalmente, as cores escolhidas foram todas pensadas de acordo com o que por padrão conseguem trazer um foco maior e consequentemente diminuir as distrações do que esteja na tela. O fundo padrão é um cinza #e2e2e2 que traz um certo tom monótono necessário para que se foque no dado apresentado. As pesquisas acerca de qual cor ajuda na concentração são divergentes e por isso se seguiu a opinião majoritária em questionamentos feitos a um grupo de designers [61]. Em questões práticas, esse sistema de gerenciamento dos agendamentos de reparo traz uma facilidade enorme na manipulação dos dados. Claro, a ideia central não é mudar a logística ou processos internos dos órgãos que fazem esse trabalho de manutenção viária, sendo esse painel de BI&A um suporte para tomada de decisão e gestão de recursos humanos e materiais. Como não houve uma metodologia à risca para o levantamento de requisitos juntamente a um gerente ou coordenador da Novacap, não foi possível se desenvolver um sistema totalmente de acordo com o que eles gostariam. Nesse sentido, é importante deixar claro que é totalmente possível fazer ajustes no sistema e que as operações mais importantes estão implementadas.

Ao se realizar uma alteração de dado no sistema de agendamento, o botão dentro do *modal* faz as requisições para o servidor e, utilizando um comando de *server-side* é possível se retornar à página anterior **header("location:javascript://history.go(-1)");**. Isso acontece porque o botão está encapsulado em um formulário HTML do tipo **<form action="http://localhost/muda-reparo.php"method="post»** por exemplo. Assim, o navegador literalmente carrega o *script* que, pela velocidade de processamento, não é perceptível ao usuário no *front-end*. O *header* redireciona então para a página anterior. Utilizando *tags* nas **divs** HTML que contém as instruções do *modal*, é utilizado o JS para fazer o *reload* da página e atualizar o dado para o usuário. É algo trivial que acrescenta para uma melhor experiência de uso. Realmente é possível se fazer muitas funcionalidades em cima da base que se foi construída.

```
// Atualiza a pagina depois de fechar o modal
$('#reparo').on('hidden.bs.modal', function () {
  location.reload();
})
// Atualiza a pagina depois de fechar o modal
$('#responsavel').on('hidden.bs.modal', function () {
  location.reload();
})
// Atualiza a pagina depois de fechar o modal
$('#finalizado').on('hidden.bs.modal', function () {
  location.reload();
})
```

Figura 3.16: Snippet para *reload* da página, aumentando usabilidade (Fonte: autor)

À primeira vista o sistema pode parecer muito simples, mas a estrutura lógica que sedimenta a base para toda a plataforma é bem robusta e possibilita um número enorme de *queries* e *data storage*. Por ser totalmente construído em ferramentas *open-source*, não conta com custos de licenciamento - que são enormes. O governo brasileiro gasta mais de R$3,7 bilhões com *software* de outros países segundo o diretor executivo da Linux Internacional, Jon Hall. Segundo uma auditoria realizada pela Controladoria Geral da União, se iniciativas de *software* livre fossem mais utilizadas, seria possível ter uma economia de cerca R$600 milhões, valor significativamente

grande e que poderia auxiliar em outros setores públicos [62].

Agora que se tem o contexto geral do projeto apresentado, se faz necessário trazer visualmente a maneira como todos os *scripts* se relacionam com o HTML, CSS e JS.

3.2 Coleta de dados

Agora que o ambiente está todo configurado e preparado para disponibilizar dados reais ao invés dos dados de teste que se tinha, é hora de estudar uma metodologia para coleta de dados. O intuito desde o início do projeto sempre fora que o cidadãos utilizassem o aplicativo *mobile*, criando um ecossistema colaborativista e, dessa forma, diminuísse o gasto de recursos públicos. Trazendo essa característica, é possível fazer com que o uso de recursos humanos para levantamento do estado das vias e rodovias do DF seja menor e as informações mais completas - dado que seria necessário um contingente de trabalhadores muito grande para fazer esse mapeamento no DF todo, que conta com uma área de 5.779 km² e 31 regiões administrativas [63]. As linhas em cor preta na figura abaixo [64] representam a rede rodoviária do DF na proporção 1:150000. As linhas viárias nessa proporção se tornam tão vascularizadas que se mesclam e dificultam uma visualização que faça a representação justa. Para tal, é interessante até se usar o Google Maps [54];

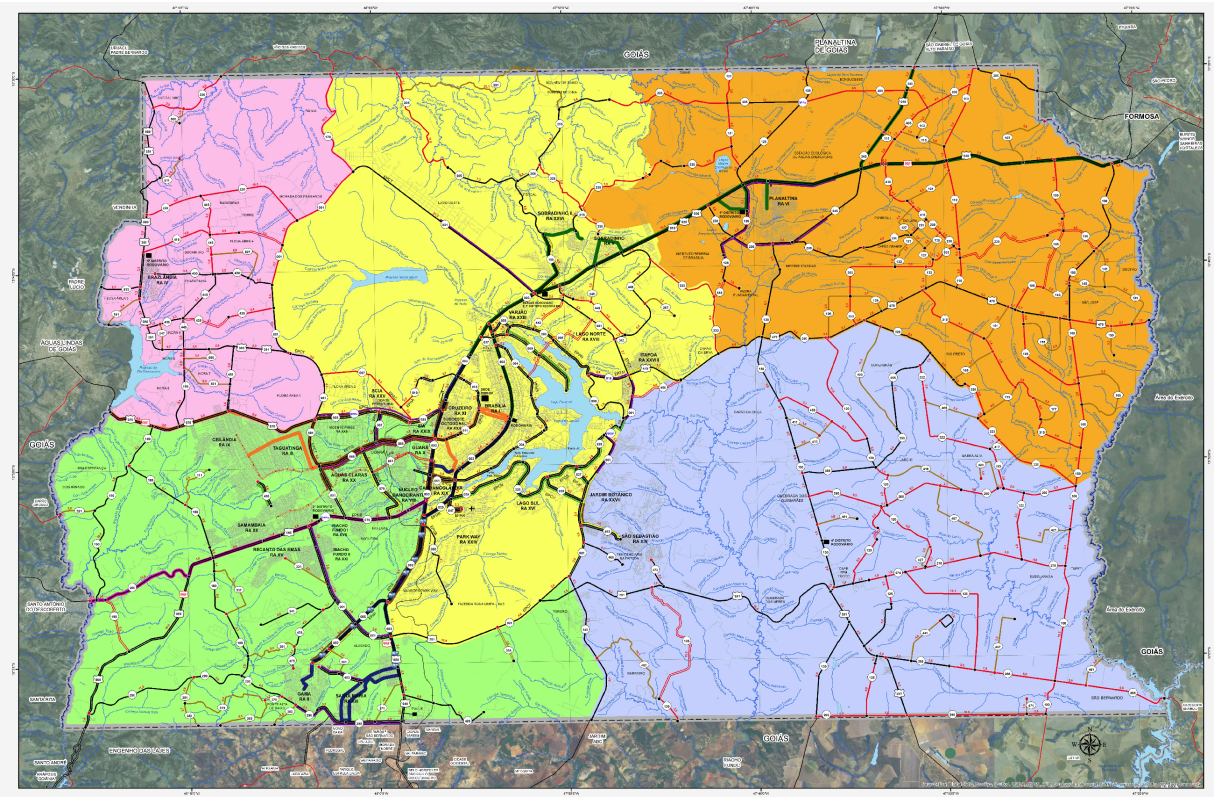
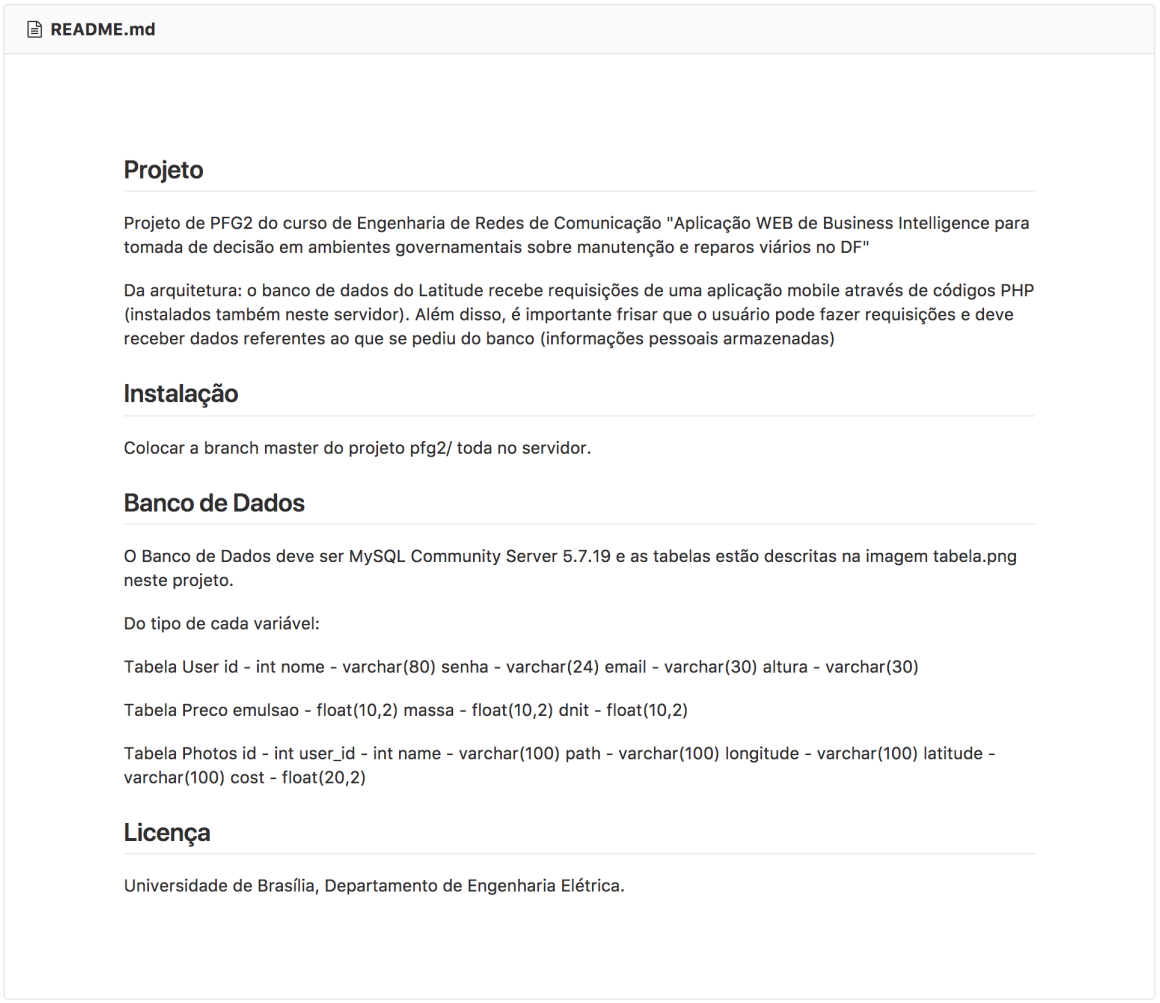


Figura 3.17: Mapeamento rodoviário do DF (Fonte: DER-DF)

Para fazer a coleta dos dados através da rede de usuários das vias, é necessário então um servidor dedicado à operação e que comporte o RDBMS para ter conectividade com os aparelhos móveis e fazer a parte de receber e enviar dados. Para isso foi aberto um *ticket* de número #000.538 no Laboratório de Tecnologias da Tomada de Decisão - LATITUDE/UnB para que esse segmento do projeto fosse nele instalado. Após isso, foi necessário expor as variáveis de conexão ao banco de dados em um arquivo de configuração para que fossem feitas alterações desses valores para o banco de homologação, e assim, a aplicação funcione corretamente.



README.md

Projeto

Projeto de PFG2 do curso de Engenharia de Redes de Comunicação "Aplicação WEB de Business Intelligence para tomada de decisão em ambientes governamentais sobre manutenção e reparos viários no DF"

Da arquitetura: o banco de dados do Latitude recebe requisições de uma aplicação mobile através de códigos PHP (instalados também neste servidor). Além disso, é importante frisar que o usuário pode fazer requisições e deve receber dados referentes ao que se pediu do banco (informações pessoais armazenadas)

Instalação

Colocar a branch master do projeto pfg2/ toda no servidor.

Banco de Dados

O Banco de Dados deve ser MySQL Community Server 5.7.19 e as tabelas estão descritas na imagem tabela.png neste projeto.

Do tipo de cada variável:

Tabela User id - int nome - varchar(80) senha - varchar(24) email - varchar(30) altura - varchar(30)

Tabela Preco emulsao - float(10,2) massa - float(10,2) dnit - float(10,2)

Tabela Photos id - int user_id - int name - varchar(100) path - varchar(100) longitude - varchar(100) latitude - varchar(100) cost - float(20,2)

Licença

Universidade de Brasília, Departamento de Engenharia Elétrica.

Figura 3.18: Arquivo de configuração do banco de dados (Fonte: autor)

Existe uma preocupação, claro, por parte do LATITUDE/UnB em questão de segurança da rede e de dados. Neste ano de 2017 fora implementado um sistema de *deploy* automatizado em que é possível que através de um ambiente compartilhado em homologação, é possível que este projeto seja atualizado constantemente pelo https://git.redes.unb.br/marcos.yan/pfg2 sem que passe por uma revisão manual de alguém responsável do Latitude/UnB. Ou seja, é permitido um acesso via SFTP para o servidor Web do laboratório para enviar os *scripts* e o conteúdo estático. Para isso, é liberada uma base de dados no servidor de banco de dados, não sendo necessário

o acesso às máquinas. Isso significa que não pode é possível executar comandos ou *scripts* no servidor via SSH, mas sim conectar no banco de dados usando as credenciais dadas. A aplicação será instalada automaticamente a partir do repositório do git.

Como essa funcionalidade de *deploy* automático foi lançada recentemente, foi necessária a criação de um *branch* no git do repositório com o nome "staging", provavelmente por uma adequação interna que lê a *branch* com esse nome. Infelizmente, a aplicação não está em produção. Segundo o engenheiro que fez o atendimento, Jorge Guilherme, apesar do servidor funcionar 24/7, esse ambiente é de homologação e, portanto, não tem as garantias de desempenho, disponibilidade e *backup* que se tem na produção. O git do LATITUDE/UnB possui uma GUI amigável e intuitiva, mas foi utilizado a CLI para fazer *merges, pushes e pulls*. Como o ambiente já estava configurado, não foi preciso seguir todos os passos de criação de repositório ou adicionar um <filename>. Simplesmente se clonou o um *path* para o repositório da aplicação local e se fez o seguinte pela CLI a partir da pasta com *path* para o git:

1. *git add .*
2. *git commit -m "update referente a tal coisa"*
3. *git push origin master*

Sem uma licença de desenvolvedor da Google, não é possível disponibilizar o aplicativo na Google Play [49]. Então decidiu-se que um instalador de extensão APK seria distribuído em um grupo de rede social da Universidade de Brasília que conta com mais de 40 mil membros para que sistemas Android pudessem baixar e ter a plataforma disponível. Essa solução não teve a adesão esperada e nenhum dado foi gerado e isso será explicado na seção de resultados. Com tempo e recursos de divulgação limitados, não foi possível trazer usuários para coleta de dados através do aplicativo. As métricas disponíveis para o gerente de projeto precisam ter uma validação dos dados disponibilizados, utilizando entradas reais de buracos que utilizem o algoritmo de dimensionamento do aplicativo *mobile*. Assim, se faz necessária uma nova abordagem.

Os dados não foram coletados desde o começo do projeto por um simples motivo: sabia-se que a arquitetura do banco iria mudar, que existiriam novas entradas que só seriam pensadas no decorrer do projeto e é exatamente isso que aconteceu. O RDBMS ficou mais robusto em questão de tabelas e informações nelas contidas. De apenas duas tabelas (**user**, **photos**) passaram a ter cinco (**user**, **photos**, **rh**, **financeiro**, **agendamentos**) e as entradas de todas as que existiam foram aumentadas para se obter mais informações sem precisar fazer nenhuma alteração no aplicativo Android. A descrição das tabelas ao final da implementação do sistema está descrita pelas figuras abaixo.

```
mysql> show tables;
+-------------------+
| Tables_in_pothole |
+-------------------+
| agendamentos      |
| financeiro        |
| photos            |
| rh                |
| user              |
+-------------------+
5 rows in set (0.00 sec)

mysql> describe photos;
+-----------+--------------+------+-----+---------+----------------+
| Field     | Type         | Null | Key | Default | Extra          |
+-----------+--------------+------+-----+---------+----------------+
| id        | int(11)      | NO   | PRI | NULL    | auto_increment |
| user_id   | int(11)      | YES  |     | NULL    |                |
| name      | varchar(60)  | YES  |     | NULL    |                |
| path      | varchar(200) | YES  |     | NULL    |                |
| latitude  | float(10,6)  | NO   |     | NULL    |                |
| longitude | float(10,6)  | NO   |     | NULL    |                |
| cost      | float(10,2)  | YES  |     | NULL    |                |
| bairros   | varchar(20)  | YES  |     | NULL    |                |
| status    | int(2)       | YES  |     | NULL    |                |
+-----------+--------------+------+-----+---------+----------------+
9 rows in set (0.00 sec)
```

Figura 3.19: Topologia final do RDBMS (Fonte: pesquisa)

```
mysql> describe user;
+--------+-------------+------+-----+---------+----------------+
| Field  | Type        | Null | Key | Default | Extra          |
+--------+-------------+------+-----+---------+----------------+
| id     | int(11)     | NO   | PRI | NULL    | auto_increment |
| nome   | varchar(80) | YES  |     | NULL    |                |
| senha  | varchar(24) | YES  |     | NULL    |                |
| email  | varchar(30) | YES  |     | NULL    |                |
| altura | float(4,2)  | YES  |     | NULL    |                |
+--------+-------------+------+-----+---------+----------------+
5 rows in set (0.00 sec)

mysql> describe rh;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| id          | int(11)     | YES  |     | NULL    |       |
| cargo       | varchar(20) | YES  |     | NULL    |       |
| nome        | varchar(30) | YES  |     | NULL    |       |
| time        | varchar(20) | YES  |     | NULL    |       |
| remuneracao | float(10,2) | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)

mysql> describe financeiro;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| governo | float(10,2) | YES  |     | NULL    |       |
| analise | float(10,2) | YES  |     | NULL    |       |
+---------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> describe agendamentos;
+-------------+-------------+------+-----+---------+----------------+
| Field       | Type        | Null | Key | Default | Extra          |
+-------------+-------------+------+-----+---------+----------------+
| id          | int(11)     | NO   | PRI | NULL    | auto_increment |
| responsavel | varchar(30) | YES  |     | NULL    |                |
| local       | varchar(20) | YES  |     | NULL    |                |
| dataentrada | varchar(20) | YES  |     | NULL    |                |
| dataagenda  | varchar(20) | YES  |     | NULL    |                |
| finalizado  | varchar(3)  | YES  |     | NULL    |                |
+-------------+-------------+------+-----+---------+----------------+
6 rows in set (0.00 sec)
```

Figura 3.20: Topologia final do RDBMS (Fonte: autor)

Por conta do sistema de segurança do LATITUDE/UnB, não foi possível fazer o *debug* para verificar o motivo pelo qual o acesso e fornecimento de dados não estava acontecendo entre o

aplicativo e o *back-end*. O que se fez foi basicamente alterar o endereço IP e a porta de acesso dentro do código do aplicativo para que o mesmo acessasse o servidor do laboratório assim como estava acessando um servidor local rodando em um *notebook*. Dessa forma, manteve-se a arquitetura originalmente criada para o tráfego de dados. Finalmente, para a coleta de dados foi feito um trabalho manual. Com um *notebook* rodando o servidor e conectado à rede através de um *hotspot* fornecido por um celular com conexão 4G e o celular rodando o aplicativo com direcionamento de IP neste servidor local, foi feito um mapeamento de maneira totalmente ineficiente em prol de um retorno de dados mais rápido afim de se testar integralmente a plataforma. Apesar da ideia inicial ser um mapeamento de todas regiões do DF, para este método focou-se na região do Plano Piloto e regiões adjacentes: Lago Norte, Lago Sul, Noroeste e Sudoeste.

O relacionamento entre os *scripts*, o RDBMS e o apresentação *web* é descrita pelo diagrama a seguir.

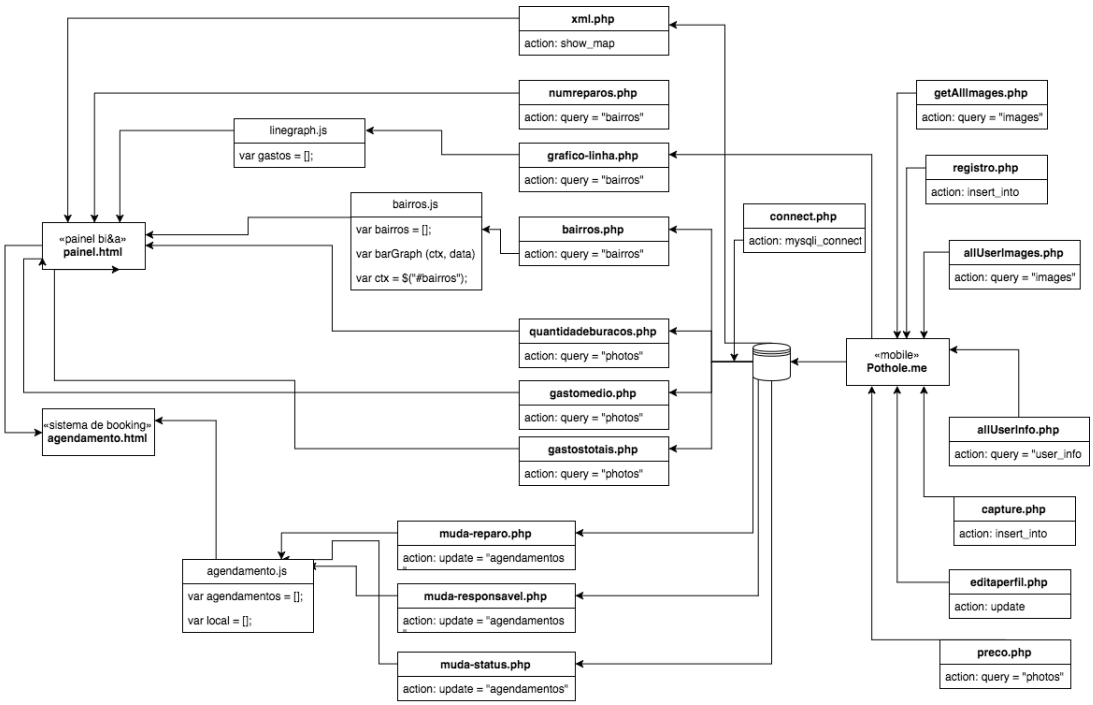


Figura 3.21: Estrutura de relacionamento dos *scripts* e *front-end* (Fonte: autor)

Capítulo 4

Resultados

Este capítulo traz os resultados obtidos através da metodologia e cronograma dispostos durante o desenvolvimento deste projeto

Como a grande maioria dos projetos, o resultado não fora exatamente de acordo com o planejamento inicial. Alguns incrementos foram feitos na plataforma da perspectiva preliminar de concepção, enquanto algumas funcionalidades e detalhamentos não foram inclusos. A partir das ações e decisões tomadas a partir da metodologia detalhada na seção anterior, aqui será explicitado cada resultado obtido através das execuções no projeto.

4.1 Levantamento de requisitos e arquitetura do projeto

Na metodologia do projeto a ideia era fazer o levantamento de dados com alguém interno à Novacap e que essa ação possibilitasse um melhor entendimento e compreensão das necessidades mais urgentes dentro da gestão de processos da companhia. As tentativas de contato através da Ouvidoria do GDF [58] e no email da Novacap (novacap@novacap.df.gov.br) não foram efetivas, foi feita uma tentativa de contato via telefone e o que se percebeu foi que a hierarquia era muito rígida dentro da instituição. Os atendentes são preparados para somente atender chamados e dar informações específicas. O contato com os gestores dos projetos e programas governamentais do GDF é bastante restrito e o Direto de Urbanização, Daclimar Castro, não pôde receber a proposta para um estudo de caso deste projeto. Dessa forma, fez-se um mapa mental com tudo o que se imaginou necessário e crucial para que a operação e seus processos fossem mais fluidos e organizados durante a execução - desde o recebimento da chamada até o fechamento da mesma.

Figura 4.1: Mapa mental de levantamento de requisitos. (Fonte: autor)

A implementação deveria seguir então as funcionalidades descritas através do mapa mental acima. A primeira tarefa então é definir a arquitetura e topologia do sistema. Apesar da metodologia descrever que o banco de dados deveria ter um IP fixo e operar em um servidor dedicado do LATITUDE/UnB, isso não foi possível devido ao tempo escasso para verificação das

barreiras de segurança que não permitiram conectividade com a aplicação *mobile*. Nesse sentido se fez necessário então voltar a uma modelagem que se tinha no projeto original e é representada abaixo.

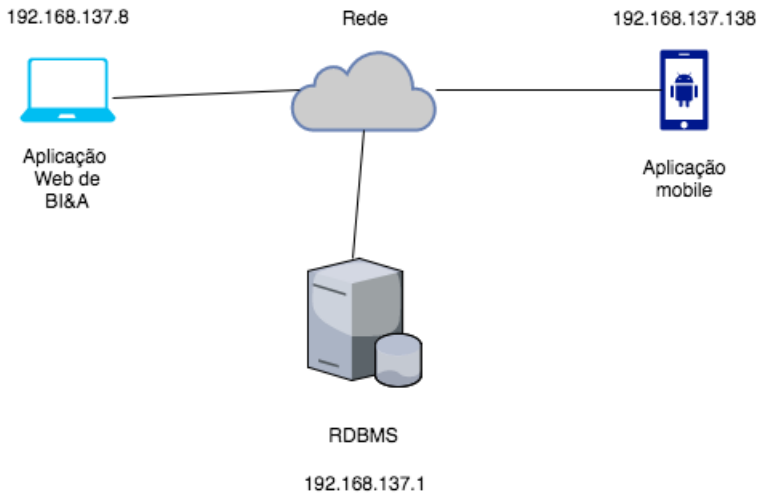


Figura 4.2: Configuração do ambiente de rede. (Fonte: autor)

A arquitetura é a mesma do projeto anterior [2] e os endereços estáticos foram mantidos para estar de acordo com a distribuição do APK e a aplicação Android, consequentemente. O aplicativo sempre fica com o IP 192.168.137.138 mesmo sem estar configurado estático. Isso porque o *hotspot* que está rodando no IP 192.168.137.1 (que é o servidor com o banco de dados) faz a distribuição desse IP para o aparelho conhecido. Já a aplicação *web* de BI&A pode ficar em qualquer IP, já que apenas um *script* é responsável por fazer a comunicação: **connect.php**. Esse *script* é requisitado por todos os outros que fazem comunicação com o RDBMS. Então, como os outros IPs estão estáticos, só é preciso deixar configurado uma única vez o endereçamento. A topologia originalmente pensada é a mesma, mas o RDBMS estaria com um IP do LATITUDE/UnB.

4.2 Coleta e extração de dados

A coleta de dados realmente foi feita de forma manual sem o suporte colaborativo inicialmente planejado. A licença de desenvolvedor Android para publicação na Google Play não foi adquirida por contar com uma taxa de U$25 (cerca de R$82), além de existirem várias políticas acerca de plágio no contrato para licenciamento [49]. Como o projeto não possui um aval governamental ou uma patente no Instituto Nacional da Propriedade Industrial (INPI), decidiu-se por manter uma distribuição não-licenciada por um APK para futuros usuários da plataforma.

Da extração de dados, foi realmente criada uma ferramenta de ETL (*extract*, *transform*, *load*) através de *server-side scripting*. Apesar de não ser realmente uma ferramenta com coman-

dos de interface CLI ou GUI, apresenta todas as características necessárias para se enquadrar como ferramenta ETL. Basicamente são feitas *queries* de toda tabela que se deseja manipular as informações e fazer o carregamento do *output* para a aplicação *web*. Toda representação e fluxo de como isso é feito segue exatamente a descrição do Capítulo 3.

Com a coleta de dados de 28 buracos e um mapeamento feito nas regiões Asa Sul, Asa Norte, Lago Norte, Lago Sul, Noroeste e Sudoeste, foi possível se obter métricas interessantes, descritas abaixo.

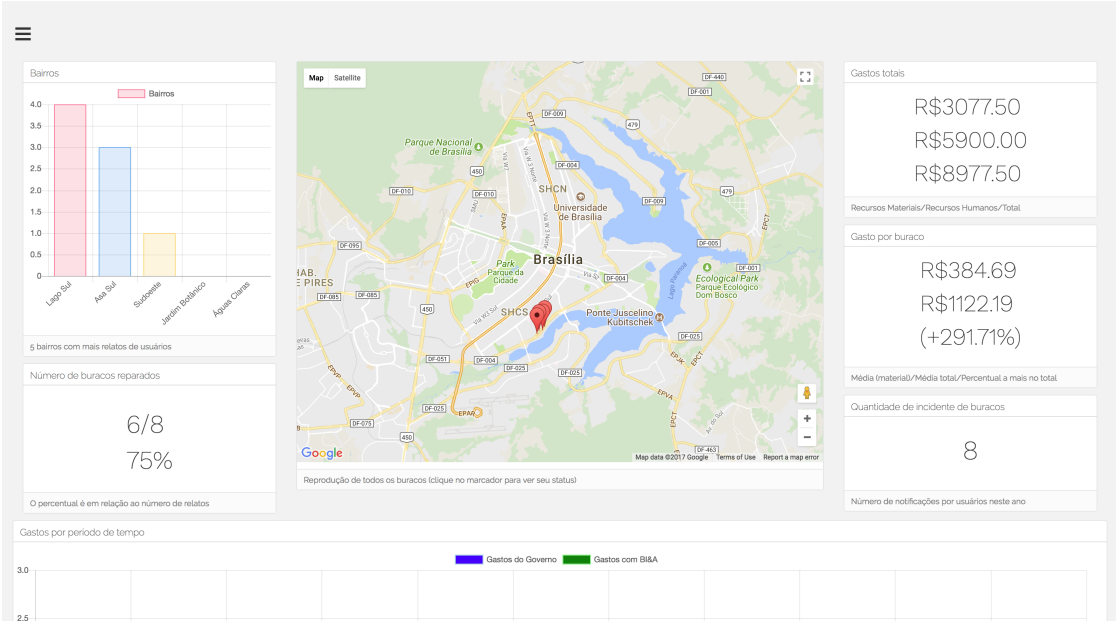


Figura 4.3: Painel funcional do BI&A. (Fonte: autor)

No gráfico de "Gastos por período de tempo", foi utilizada uma média do mês de maio de 2017 adiante. Isso porque o DNIT não liberou os dados desse período. De qualquer forma, espera-se que exista um gasto mais controlado, dado que o sistema é de ponta a ponta, garantindo uma maior transparência e gerência adequada para a operação. De acordo com Zenor [65], um gerenciameto adequado dentro das operações pode aumentar o lucro em 28% dentro de uma empresa. No caso deste projeto o lucro seria a economia obtida a partir de uma administração apropriada.

4.3 Sistema de agendamento

Cada entrada nova no sistema gera um *ticket* novo dentro do sistema de agendamento criado como complemento ao painel BI&A para que o sistema fosse realmente robusto ao nível de um gerente de projeto ter que utilizá-lo diariamente para um boa gestão. Essa questão de acostu-

mar o usuário à uma nova plataforma é essencial para que boas práticas sejam implementadas no futuro. O que mais se observa são sistemas muito robustos e com diversas funcionalidades sendo apresentadas de maneira imediata após execução do projeto e não sendo utilizados. Isso acontece porque a mudança de hábitos de trabalho são difíceis de se implementar dentro de uma equipe grande como a que se tem na Novacap, por exemplo. A usabilidade no geral está muito boa, o gerente de projeto não precisa ter nenhum conhecimento técnico relacionado a desenvolvimento de *software*. No entanto, para fazer qualquer alteração, é necessário se referenciar ao buraco que se quer fazer modificações de dados. Um exemplo é dado a seguir.

Figura 4.4: Linha antes da modificação de dado. (Fonte: autor)

Quando uma alteração é feita, é necessário que se referencie à linha, ou seja, o ID do buraco que é uma chave primária no RDBMS que possui um auto-incremento a cada entrada nova no banco. É uma maneira bem rústica de se ter uma correspondência ao número da linha que deve ser alterada, mas a manipulação do JSON a partir do HTML não foi possível no tempo hábil. O corpo o JSON está tão encapsulado no *framework* Bootstrap Modal que o retorno de determinada linha está obscura demais para uma fácil extração.

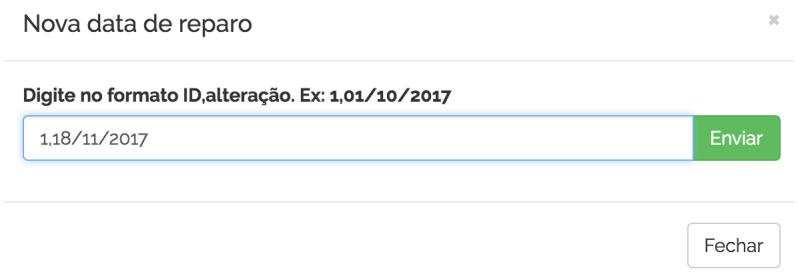


Figura 4.5: Padrão de referenciamento necessário para alteração de dados. (Fonte: autor)

Dessa forma, é necessário que a alteração tenha a seguinte formatação: "**numID,DadoNovo**", sem aspas. Como mostrado na figura abaixo, o novo dado é dia 18/11/2017 e referencia o ID número 1.

Figura 4.6: Linha após alteração. (Fonte: autor)

Como se pode perceber, a alteração foi feita - e sem precisar de uma atualização manual na página. O próprio modal foi encapsulado dentro de *listener* que checa quando ele foi clicado e atualiza a

página após alteração do dado. Isso agiliza o trabalho do gerente de projeto, bem como melhora a experiência de uso.

Do ponto de vista dos dados, todos estão de acordo com o banco, então não existe nenhum erro no *parsing* das informações. Isso também vale para os gráficos e métricas apresentados no painel de BI&A, em que todos os cálculos e algoritmos de modelagem e manipulação dos dados foram minuciosamente verificados. Por ser um sistema relativamente compacto, não existe muito espaço para dúvidas e nem *bugs*. Diversos casos de uso foram testados (como ir e voltar na página, clicar no botão de enviar do modal duas vezes rapidamente, tentar ficar dando *refresh* repetidamente para causar uma obstrução na transferência entre o banco e o *front-end*) e nenhum causou algum *crash* ou falha no sistema.

Capítulo 5

Conclusões

5.1 Perspectivas futuras

Este sistema apresentado como Projeto Final de Graduação é apenas uma das inúmeras frentes possíves para o projeto como um todo. Tudo se iniciou com o desenvolvimento de uma aplicação *mobile* que faz a detecção, parametrização e precificação de buracos a partir de fotos tiradas pelo celular. A partir disso, os dados foram modelados neste projeto e existem várias linhas para se seguir daqui em diante.

5.1.1 Integração veicular

Em 2015, a Google registrou uma patente de tecnologia para mapear buracos nas ruas. A patente que permitiria à empresa utilizar a localização do carro do usuário por GPS e sensores para detectar buracos nas ruas e armazenar a informação em um banco de dados. Assim, os serviços da empresa poderiam sugerir a motoristas caminhos menos esburacados com uma integração do Google Maps ou outro serviço de GPS [66]. Isso mostra que existe uma preocupação a um nível mundial acerca da questão de qualidade da manutenção viária.

Como se foi desenvolvido um algoritmo para fazer essa detecção [2], seria interessante fazer a integração deste com um veículo, de forma que o mapeamento fosse mais rápido e não dependesse do colaborativismo da população. Essa ação traria medidas mais ágeis por parte dos responsáveis pela manutenção, bem como teria um gasto mínimo.

5.1.2 Mapeamento com *drones*

A ideia aqui é de se utilizar veículos aéreos não-tripulados, *drones*, para fazer o mapeamento das vias. Embarcando uma câmera e um sistema de processamento ao *drone*, fazendo possível o uso do algoritmo de parametrização, poderia se utilizar a tecnologia para cobrir uma determinada área muito mais rapidamente em comparação com a integração veicular descrita acima.

5.1.3 Aumentar a robustez do painel BI&A

Este projeto teve uma preocupação de consolidar uma base consistente em que se fosse possível escalar a aplicação seguindo boas práticas de desenvolvimento. Os componentes estão bem detalhados e o código está bem modularizado, de forma que o reuso pode ser empregado. Houve um aumento de informações dentro das tabelas que já existiam para que fosse possível criar o sistema de agendamento e controle financeiro pelo painel. O aumento da robustez do painel BI&A pode ser feito para projetos futuros de forma bem simples, dado que o banco é um RDBMS fácil de ser modelado e suporte um volume muito grande de dados. O foco de um próximo projeto nessa linha seria realmente desenvolver métricas que gerem uma previsibilidade maior sobre a situação dos buracos.

5.2 Conclusões

Finalmente, o projeto foi bem sucedido dentro da definição do seu escopo. Foi dada uma visibilidade dos dados coletados pelos usuários que não existia para a população e para o gerente de projeto responsável pela manutenção viária. A robustez e transparência do sistema foi aumentada de forma que até mesmo os gerenciamentos de reparos estão disponíveis em um sistema de agendamento e os dados coletados são, agora, modelados para que suportem tomadas de decisão dentro das instituições incumbidas da tarefa de conservação das vias do DF.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] (IBGE), I. B. de Geografia e E. Frota de veículos no brasil. *cidades.ibge.gov.br/painel/frota.php*, 2016.

[2] WILLIAN, L. Aplicativo mobile de parametrização e precificação de reparos de buracos viários no df. *Universidade de Brasília*, 2017.

[3] BRASILEIRO, G. Portal da transparência. *http://www.transparencia.df.gov.br//despesas/orgao*, 2017.

[4] ABNT. Nbr 9935. *Agregados - Terminologia*, 2010.

[5] BRASíLIA, A. de. Novacap intensifica operação tapa-buraco. *https://www.agenciabrasilia.df.gov.br/2016/11/24/novacap-intensifica-operacao-tapa-buraco-por-causa-das-chuvas/*, 2016.

[6] CORPORATION, N. C. Server-side javascript guide. *http://docs.oracle.com/cd/E19957-01/816-6411-10/contents.ht*, 1998.

[7] IDIOCY, T. Java server and client time zone difference problem solved. *http://techidiocy.com/java-server-and-client-time-zone-difference-problem-solved/*, 2015.

[8] W3TECHS. Javascript no mundo. *https://w3techs.com/technologies/details/cp-javascript/all/all*, 2017.

[9] ORACLE. Mysql about. *https://www.mysql.com/about/*, 2017.

[10] CALLAGHAN, M. Mysql at facebook. *https://www.youtube.com/watch?v=Zofzid6xIZ4*, 2015.

[11] ZORATTI, I. The evolution of open-source databases. *OpenWorld Forum in Paris*, 2014.

[12] TECHTARGET. Lamp (linux, apache, mysql, php). *http://searchenterpriselinux.techtarget.com/definition/* 2009.

[13] RAJPUT, M. What is the mean stack and why is it better than lamp? *https://www.programmableweb.com/news/what-mean-stack-and-why-it-better-lamp/analysis/2015/12/22*, 2015.

[14] OPERATING Systems — Introduction to Information and Communication Technology. *http://openbookproject.net/courses/intro2ict/system/os$_i$ntro.html*, 2001.

[15] TECHFAE. Linux is running on almost all of the supercomputers. *https://www.techfae.com/linux-running-almost-supercomputers/*, 2016.

[16] W3TECHS. Usage of os for websites. *http://w3techs.com/technologies/overview/operating$_s$ystem/all*, 2015.

[17] FOUNDATION, T. A. S. Http server project. *http://httpd.apache.org/about*, 2017.

[18] ITU. Basic texts of the international telecommunication union. *http://www.itu.int/net/about/basic-texts/index.aspx*, 2009.

[19] SPACE, S. Os/linux distributions using apache. *https://secure1.securityspace.com/s$_s$urvey/data/man.*2017

[20] WONG, Y. Why hasn't facebook migrated away from php? *https://www.quora.com/Why-hasn-t-Facebook-migrated-away-from-PHP*, 2016.

[21] NET, P. Php: rfc:phpng. *https://wiki.php.net/rfc/phpng*, 2017.

[22] 1TRAINING. How can you use html, css and javascript for web development? *https://www.1training.org/how-can-you-use-html-css-and-javascript-for-web-development/*, 2017.

[23] W3. Html 5.1 2nd edition. *https://www.w3.org/TR/html/*, 2017.

[24] NETWORK, M. D. Css developer guide. *https://developer.mozilla.org/en-US/docs/Web/Guide/CSS*, 2017.

[25] SEGUETECH. What is ajax and where is it used in technology? *https://www.seguetech.com/ajax-technology/*, 2010.

[26] GARMIN. What is gps? *http://www8.garmin.com/aboutGPS/*, 2017.

[27] GOVERNMENT, U. Gps accuracy. gps: The global positioning system. *https://www.gps.gov/systems/gps/performance/accuracy/*, 2017.

[28] DJUKNIC, G. M.; RICHTON, R. E. Geolocation and assisted gps. *http://www.cs.columbia.edu/ drexel/CandExam/Geolocation$_a$ssistedGPS.pdf*, 2001.

[29] EMERY, D. Standards, apis, interfaces and bindings. *http://www.acm.org/tsc/apis.html*, 2002.

[30] FORCE, I. E. T. The application/json media type for javascript object notation (json). *https://tools.ietf.org/html/rfc4627*, 2006.

[31] INTERNATIONAL, E. The json data interchange format. *http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf*, 2013.

[32] CAMPOS, D. P. D. A. J. R. D. Definição de requisitos de software baseada numa arquitetura de modelagem de negócios. *http://www.redalyc.org/pdf/3967/396742032003.pdf*, 2008.

[33] DUMAS, M.; HOFSTEDE, A. H. ter. Uml activity diagrams as a workflow specification language. *http://bit.ly/2A3j1c2*, 2001.

[34] OMG/UML. Uml usage. *http://www.uml.org/*, 2015.

[35] GOOGLE. Hal varian answers your questions. *http://www.freakonomics.com/2008/02/25/hal-varian-answers-your-questions/*, 2008.

[36] MANYIKA J., C. M. B. B. B. J. D. R. R. C.; BYERS, A. H. Big data: The next frontier for innovation, competition, and productivity. *http://www.mckinsey.com/insights/mgi/research/technology$_a$nd$_i$nnovation/big$_d$ata$_t$he$_n$ext$_f$rontier$_f$or$_i$nnovation*, 2012.

[37] CHAUDHURI S., D. U.; NARASAYYA. An overview of business intelligence technology. *Communications of the ACM (54:8), pp. 88-98*, 2011.

[38] STOREY, H. C. R. H. L. C. V. C. Business intelligence and analytics: From big data to big impact. *http://bit.ly/2zVRYi9*, 2012.

[39] CHAUDHURI, U. D. S.; NARASAYYA, V. An overview of business intelligence technology. *Communications of the ACM p. 1-20*, 2011.

[40] SHINAL, J. Google is grabbing more and more ad revenue from partners. *https://www.cnbc.com/2017/04/27/alphabets-google-unit-grabbing-ever-more-ad-revenue-from-partners.html*, 2017.

[41] DAVENPORT, T. H. Competing on analytics. *Harvard Business Review (84:1), p. 98-107*, 2006.

[42] ECONOMIST, T. Beyond the pc. *http://www.economist.com/ node/21531109*, 2011.

[43] KULIKOV, O. Data mining for social networks. *Technische Universitat Munchen*, 2009.

[44] CONSTINE, J. Facebook now has 2 billion monthly users... and responsibility. *https://techcrunch.com/2017/06/27/facebook-2-billion-users/*, 2017.

[45] NETTLETON, D. Data mining of social networks represented as graphs. *Computer Science Review 7, Universitat Pompeu Fabra, Barcelona, Spain, pp.2-9*, 2013.

[46] KLEBNIKOV, S. 8 innovative ways elon musk made money before he was a billionaire. *Time, http://time.com/money/4883868/8-innovative-ways-elon-musk-made-money-before-he-was-a-billionaire/*, 2017.

[47] TESLA. Elon musk. *https://www.tesla.com/elon-musk*, 2017.

[48] SPACEX. Elon musk. *http://www.spacex.com/elon-musk*, 2017.

[49] GOOGLE. App licensing. *https://developer.android.com/google/play/licensing/index.html*, 2017.

[50] WAZE. About us. *https://www.waze.com/about*, 2017.

[51] ORACLE. Download mysql community server. *https://dev.mysql.com/downloads/mysql/*, 2017.

[52] OPEN-SOURCE. Getting started to chartjs. *http://www.chartjs.org/docs/latest/*, 2017.

[53] BARJIS, J. The importance of business process modeling in software systems design. *Delft University of Technology, http://www.sciencedirect.com/science/article/pii/S0167642308000038*, 2008.

[54] MAPS, G. Javascript api reference. *https://developers.google.com/maps/documentation/javascript/*, 2017.

[55] MILLER, R. Aws continues to rule the cloud infrastructure market. *TechCrunch, https://techcrunch.com/2017/10/30/aws-continues-to-rule-the-cloud-infrastructure-market/*, 2017.

[56] DF, G. do. Receitas por órgão. *Portal da Transparência, http://www.transparencia.df.gov.br//receitas/por-orgao*, 2017.

[57] DNIT. Custos e pagamentos. *Centro-Oeste, http://www.dnit.gov.br/custos-e-pagamentos/sicro/centro-oeste/centro-oeste*, 2017.

[58] DF, O. do. Registre sua manifestação. *http://www.ouvidoria.df.gov.br/registre-sua-manifestacao.html*, 2017.

[59] W3SCHOOLS. Bootstrap get started. *https://www.w3schools.com/bootstrap/bootstrap$_g$et$_s$tarted.asp*, 2017.

[60] STATCOUNTER. Top browsers per country. *http://gs.statcounter.com/browser-market-sharemonthly-201707-201707-map*, 2017.

[61] DESIGN, G. What would be a less distracting background for white? *https://graphicdesign.stackexchange.com/questions/93533/what-would-be-a-less-distracting-background-for-white*, 2017.

[62] TECMUNDO. Governo temer vai abandonar software livre para comprar produtos microsoft. *https://www.tecmundo.com.br/microsoft/111214-governo-temer-abandonar-software-livre-comprar-produtos-microsoft.htm*, 2016.

[63] FEDERAL, G. do D. Geografia do df. *http://www.brasilia.df.gov.br/333/*, 2017.

[64] FEDERAL, D. de Estradas de Rodagem do D. Mapa rodoviário. *http://www.der.df.gov.br/o-der/mapa-rodoviario.html*, 2017.

[65] MJ, Z. The profit benefits of category management. *MJ Zenor - Journal of Marketing Research p1-11*, 1994.

[66] DIGITAL, O. Google registra patente de tecnologia para mapear buracos nas ruas. *https://olhardigital.com.br/noticia/google-registra-patente-de-tecnologia-para-mapear-buracos-nas-ruas/50794*, 2015.

6. PROGRAMAS UTILIZADOS

6.1 ShareLaTeX

ShareLaTeX é um editor *online* de LaTeX que permite compilação em tempo real do código e foi utilizado para a escrita do relatório deste projeto.

6.2 Sublime Text 2

Sublime é um simples editor de texto que possui comandos e atalhos que facilitam a vida do desenvolvedor. Todos os códigos foram escritos pelo Sublime e os que precisavam ser compilados/interpretados eram feitos então pela linha de comando através de compiladores próprios, como o PHP.

6.3 Google Chrome

Foi utilizada a versão 62.0.3202.94 (Official Build) (64-bit), principalmente pela facilidade em fazer *debug* no código através do console que informa *status* da rede, se conexões e requisições foram feitas com sucesso. Esse detalhe é fundamental por conta da ligação que é feita entre o *front-end* e o *back-end*.