# WalkingInfo

# Architecture Notebook

There is guidance within this template that appears in a style named InfoBlue. This style has a hidden font attribute that allows you to toggle whether it is visible or hidden in this template. Use the Microsoft® Word® menu **Tools > Options > View > Hidden Text** check box to toggle this setting. There is also an option for printing: **Tools > Options > Print**.

# Purpose

Esta aplicação tem como foco a usabilidade e conquistar o maior número de clientes possíveis. O primeiro ponto se deve ao fato de que o sistema será web, buscamos isso porque não queríamos que os usuários se fixem em uma só plataforma e ao mesmo tempo possam utilizá-lo em qualquer lugar. Já o segundo baseia-se muito forte no sucesso do primeiro, onde aqui, quanto mais pessoas colaborando, mais forte e conhecido será a aplicação.

# Architectural goals and philosophy

A aplicação será desenvolvida de forma totalmente nova, ou seja, sem utilizar sistemas antigos ou legados. Ao mesmo tempo, com base nas boas práticas de programação buscaremos diminuir o índice ou necessidade de um alto número de manutenção, mas a nossa maior preocupação serão os dados dos usuários e para isso utilizaremos um banco de dados com suporte oficial da Google, o Firebase.

# Assumptions and dependencies

Como dito no item 2, por preocupação com os dados do usuário, manter a velocidade alta por ser uma aplicação web e facilidade de uso utilizaremos o Firebase como banco. Para todo o desenvolvimento da aplicação optamos pela linguagem JavaScript pois tem crescido rapidamente no mercado já que possui boa performance, usabilidade, ferramentas, além do fato que nosso desenvolvedor é muito familiarizado com ela . Por fim, para parte visual e estruturação, teremos CSS e HTML, respectivamente.

# Architecturally significant requirements

* **Functional requirements:** Divididos em 10 casos uso, fazer login, publicar obstaculos, identificar obstáculos, denunciar publicações, remover publicações obsoletas, avaliar publicações, realizar registro fotográfico, realizar backup e importar backup respectivamente. Checar documentação do Analista, sobre as especificações de cada caso de uso.
* **Non-functional requirements:** No quesito de funcionamento, ignorando a parte do usário interagindo com o sistema e o tempo que ele leva para escolher cada opção, cada caso de uso deve levar no máximo 8 segundos para realizar o processo.
* **About tecnhology:**

<https://firebase.google.com/docs/web/setup>

<https://www.javascript.com/>

<https://facebook.github.io/react/>

# Decisions, constraints, and justifications

* Banco de Dados: Firebase porque é gratuito, tem boa perfomance e é um produto da Google.
* Linguagem de Programação: JavaScript porque é muito versátil, altamente utilizada e com ela ganhamos a possibilidade de utilizar REACT.

# Architectural Mechanisms

# Firebase

Banco de dados escolhido para a aplicação, atualmente o ambiente dele já está configurado.

# JavaScript

Linguagem escolhida para desenvolver a aplicação, atualmente está associada ao ambiente do banco de dados.

# Architectural views

O Firebase tem como propriedade ser um banco de dados hierárquico que utiliza como manipulação de arquivos um sistema de árvores hierárquica em *Json*.

## Recommended views

* **Logical:** Todos os dados do Firebase Realtime Database são armazenados como objetos JSON. O banco de dados atualmente possui uma árvore JSON hospedada na nuvem. Ao contrário de um banco de dados SQL, não há tabelas nem registros. Quando você adiciona dados à árvore JSON, eles se tornam um node na estrutura JSON com uma chave associada. Você pode fornecer chaves próprias, como códigos de usuário e nomes semânticos.
* **Operational:** O banco de dados usa uma árvore JSON, mas os dados armazenados nele podem ser representados como tipos nativos que correspondam a tipos JSON disponíveis. Assim, é possivel escreve códigos mais fáceis de atualizar.

# Layers or architectural framework

Para a nossa solução, utilizaremos o padrão arquitetural Model-View-Controller (MVC). É uma forma de quebrar uma aplicação, ou até mesmo um pedaço da interface de uma aplicação, em três partes: o modelo, a visão e o controlador.

Nesse padrão o Controller que interpreta as entradas do mouse ou do teclado enviado pelo usuário e mapeia essas ações do usuário em comandos que são enviados para o Model e/ou para a View para efetuar a alteração apropriada. Por sua vez o modelo (Model) gerencia um ou mais elementos de dados, responde a perguntas sobre o seu estado e responde a instruções para mudar de estado.

# Architectural views

## Recommended views

* **Logical:** Seguir a estrutura de pacotes do padrão arquitetural escolhido, MVC;
* **Operational:** Single-thread;
* **Use case:** Verificar o arquivo **“**diagramas.astah”, para encontrar os diagramas de cada caso de uso.
* **Conceptual:** Verificar imagem abaixo.

