Chico d'Água - doc

-Descrição do problema

O Brasil tem uma porcentagem considerável da quantidade de água potável mundial, mas com uma área de proporções continentais, e com uma deficiência de distribuição igualitária dentro do território, seu uso consciente nos locais em que se faz disponível é de extrema importância. Dentre todas as áreas que utilizam-se deste recurso a irrigação é a que mais consome, e assim naturalmente foi-se desenvolvidos por meio de estudos diversas técnicas que resultam em um uso eficiente da água.

As técnicas desenvolvidas em que se obtém maior acurácia depende diretamente de medições físicas tais como velocidade do vento, umidade relativa do solo, dentre outros. A obtenção desses parâmetros se faz por meio de equipamentos disponíveis no mercado para que produtores possam obter e usufruir de tudo que o equipamento oferte.

Tais equipamentos normalmente estão associados a um alto valor financeiro imobilizando de imediato pequenos e médios produtores que desejam utilizar da água de maneira eficiente, como obter um grau técnico para o desenvolvimento de suas plantações extraindo assim o melhor de seu plantio. Correndo ao lado das técnicas que obtém alta acurácia a partir de parametrização física, há os métodos que trocam um pequeno grau de acurácia em sua estimativa pelo bem do uso de parâmetros empíricos como temperatura.

Um desses métodos que se utilizam de parâmetros empíricos para sua estimativa é o método de Hargreaves-Samani, um método que utiliza a temperatura do dia anterior, a fase em que se encontra a cultura (ou um coeficiente de cultura) e algumas informações sobre a cultura em si para fornecer um coeficiente de evapotranspiração, por meio do qual a partir de algumas transformações, podemos calcular a quantidade de tempo que um sistema de irrigação necessita estar ligado.

Com o uso do método de Hargreaves-Samani podemos suprir uma demanda no nicho de pequenos e médios produtores oferecendo uma ferramenta de baixo custo e com bom grau de acurácia que dispensa equipamentos caros.

-Análise de requisitos

Para resolver, ou ao menos, remediar parcialmente o problema, é proposto o desenvolvimento de um app para dispositivos móveis que se utilize do método de Hargreaves-Samani para estimar a quantidade de tempo necessário que um sistema de irrigação necessita estar ligado para suprir as necessidades hídricas de determinada cultura.

Este app, para que consiga desempenhar o papel desejado, deve cumprir alguns requisitos, os quais estão dispostos a seguir.

Requisitos Funcionais

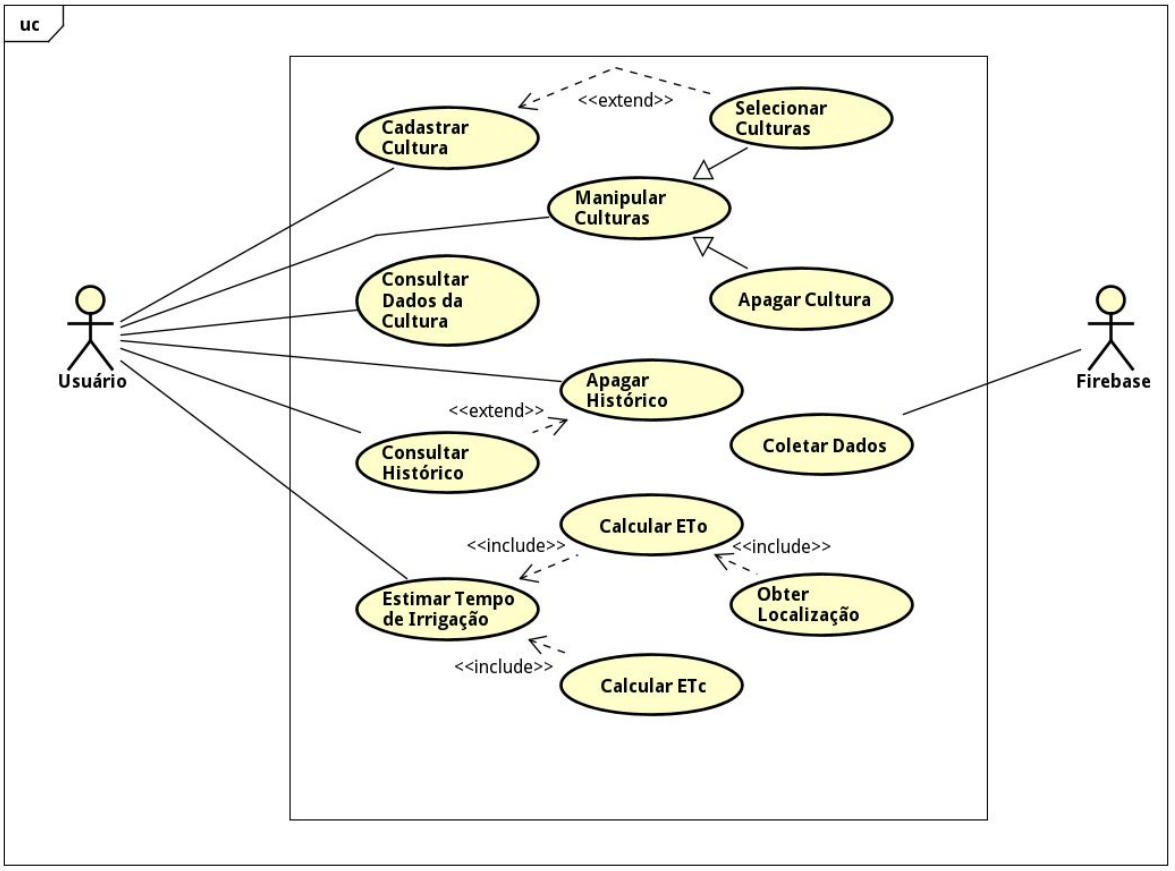
[RF001] O app deve coletar a localização do usuário.
[RF002] O usuário deve fornecer dados sobre sua cultura e equipamentos de irrigação.
[RF003] O usuário deve fornecer dados de temperatura diária.
[RF004] O app deve calcular o coeficiente de evapotranspiração de referência.

[RF005] O app deve calcular o coeficiente de evapotranspiração de cultura.
[RF006] O usuário deve informar o coeficiente de cultura ou fase atual da cultura.
[RF007] O usuário deve cadastrar uma cultura ao utilizar o app pela primeira vez.
[RF008] O app deve fazer backup em nuvem dos resultados de estimativa a cada novo cálculo de tal.
[RF009] O app deve comparar a localização coletada do usuário com localidades pré-carregadas no app, para efeito de calibração de coeficientes.
[RF010] O app deve fornecer a opção "Outra" na ocasião de cadastro de cultura, para o caso onde a cultura desejada não esteja contemplada entre as opções pré-carregadas no app.

Requisitos Não-Funcionais

[RNF001] O app deve fornecer ao usuário uma estimativa no padrão "*x* minutos" ou "*y* horas e *z* minutos".
[RNF002] O usuário deve ser capaz de cadastrar culturas adicionais.
[RNF003] O app deve mostrar os dados da cultura em uso em determinado momento.
[RNF004] O usuário pode ter acesso a um histórico das estimativas calculadas.
[RNF005] Não deve ser possível identificar o usuário com o dado de backup em nuvem.
[RNF006] O usuário deve estar ciente da coleta da sua localização pelo app.

O diagrama de casos de uso apresentado a seguir busca complementar o entendimento sobre as partes do app e seus agentes, além de especificar o relacionamento entre determinadas funcionalidades específicas.



-Ferramentas

Para o desenvolvimento do app é utilizado o framework ***Flutter***, ferramenta desenvolvida pela Google para desenvolvimentos de apps híbridos para ***Android*** e ***iOS***, e futuramente para *Web* e *Desktop*. ***Flutter*** é orientado ao design e seu motor foi desenvolvido para que se obtenha desempenho de app nativo, possui versão estável e um bom número de bibliotecas em seu repositório.

Flutter atua em conjunto com a linguagem de programação ***Dart***, também desenvolvida pela Google, o que em termos integração, obter um ótimo desempenho no geral. Dado o seu alcance uma vez codificado, seu desempenho e também a facilidade para desenvolvimento, ***Flutter*** tornou-se a escolha para o desenvolvimento do app Chico d'Água.

Se faz necessário que alguns dados da estimativa gerada, em conjunto com os dados que colaboraram para o contexto de tal sejam colhidos, para fins de estatística, sendo assim é utilizado o ***Firebase***, um banco de dados de tempo real na nuvem fácil de usar e bem integrado ao ***Flutter***. O ***Firebase*** em si oferece alguns produtos, aqui estaremos usando o *RealtimeDatabase* como local de backup destes dados.

Até este ponto no desenvolvimento do app está sendo utilizada a IDE ***Android Studio*** que possui plugin para o ***Flutter*** porém nada impede a utilização de outra IDE. Por mais que o código seja híbrido, há apenas releases para a plataforma ***Android***, a qual está majoritariamente presente nos aparelhos do público alvo do app, que são pequenos e médios produtores.

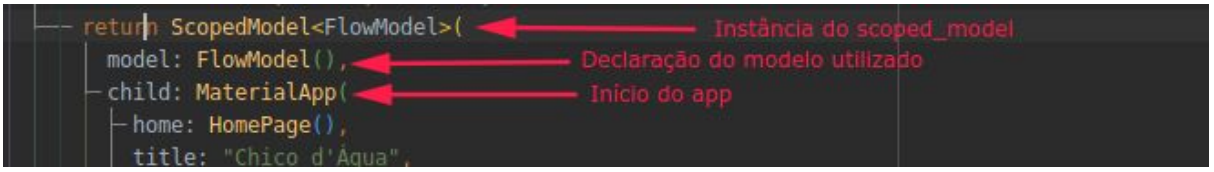
Para os diagramas, seja atividades ou de casos de uso, foi utilizada a ferramenta ***Astah***, o diagrama de classes foi auto-gerado por meio de uma biblioteca disponível no repositório, mas isso será discutido mais adiante, pois há alguns usos do repositório que merecem algumas palavras sobre. As configurações para de dependências podem ser encontradas no arquivo "*pubspec.yaml*" na raiz do projeto.

Algumas dependências

-scoped_model

Como neste app utilizamos dados que devem sempre estar à disposição, pois configuram variáveis nos cálculos, e seria bastante incômodo realizar a leitura do arquivo com os dados todas as vezes, utilizamos o *scoped_model*. A função do *scoped_model* é manter dados sempre a disposição em variáveis que estão disponíveis para consulta em todo o âmbito do app, flexibilizando assim a consulta a tais.

Para sua utilização é necessário, em geral, dois passos: 1 - criar um modelo de dados para a aplicação, ou seja, criar uma classe com as devidas variáveis e métodos necessários; e 2 - envolver todo o código do app com ele, assim ele sempre pode ser chamado de qualquer ponto do app, este "envolver" consiste em criar uma instância do *scoped_model*, setar o modelo de dados previamente criado, e colocar todo o resto (código do app) como filho desta instância.



-dartdoc

Esta biblioteca é utilizada para "auto-gerar" documentação para o app, quando gerada essa biblioteca renderiza uma página web que consiste em todos as classes, métodos e variáveis do projeto apresentando as devidas dependências de cada qual e suas devidas descrições.

As classes, métodos e variáveis do framework possuem sua descrição, porém as que o desenvolvedor codificou não. Para resolver isso o procedimento é simples: exatamente acima de cada classe, método ou variável basta colocar "///" (três barras) e escrever logo a frente sua descrição.

```
5    /// Classe que abriga os dados das cidades e culturas já calibradas, além de
6    /// conter os métodos de manipulação de dados e afins do aplicativo.
7    class DataStuff {
```

Para que seja gerada a documentação, instale o *dartdoc*, com um terminal aberto na raiz o projeto digite o comando "*dartdoc"*, os arquivos gerados serão armazenados na própria raiz do projeto, na pasta "doc".

```
[rai@EveArch AndroidStudioProjects]$ cd chico_dagua/
[rai@EveArch chico_dagua]$ dartdoc
Documenting chico_dagua...
parsing /home/rai/AndroidStudioProjects/chico_dagua/lib/model/session_model.dart...
parsing /home/rai/AndroidStudioProjects/chico_dagua/lib/model/flow_model.dart...
```

-dcdg

A biblioteca *dcdg* (Dart Diagram Class Generator) é a responsável por gerar o diagrama de classes do projeto, mas para isso é necessário instalar-lo e logo após seu uso "compilar" seu resultado, mas vamos por partes.

Sua instalação não se dá via "*pubspec.yaml*", para ele seguiremos um outro caminho. Para início de conversa temos que setar algumas variáveis de ambiente, pois precisamos de alguns executaveis que estão na pasta "/bin" do sdk do *Flutter* e do sdk do *Dart*. O sdk do *Dart* está dentro da pasta do *Flutter*.

As variáveis que necessitamos definir são as constantes na imagem abaixo. Note que são variavéis do Linux, caso esteja usando outro SO faça uma pequena busca, mas os caminhos a partir da pasta "/flutter" são os mesmos.

```
export PATH=$PATH:/home/rai/flutter/.pub-cache/bin
export PATH=$PATH:/home/rai/flutter/bin/cache/dart-sdk/bin
```

Com as variáveis devidamente configuradas em seu terminal, para utilizar o *dcdg* o procedimento é o seguinte, inicialmente instalamos o *dcdg* com o comando "*pub global activate dcdg"* e logo após, estando na raiz do projeto, digite "*pub global run dcdg -o chico_dagua.plantuml*", aqui é usado o parâmetro "-o" para que a saída seja encaminhada para um arquivo, que no caso chamamos de "chico_dagua.plantuml", e que está localizado na raiz do projeto.

```
[rai@EveArch AndroidStudioProjects]$ cd chico_dagua/
[rai@EveArch chico_dagua]$ pub global run dcdg -o chico_dagua.plantuml
```

A nossa saída neste ponto é um arquivo com extensão .plantuml, esse arquivo é fruto do PlantUML, uma ferramenta/linguagem que descreve diagramas da UML, e entre eles o diagrama de classes. Para saber mais sobre a sintaxe do PlantUML acesse o site do projeto.

Uma vez gerado nosso arquivo .plantuml, copiamos seu conteúdo, vamos até o site do projeto PlantUML pois lá temos o "compilador" da linguagem, colamos o conteúdo e temos, finalmente, nossa imagem com o diagrama de classes.

Para a segunda parte do processo, a qual temos em mãos o código do PlantUML, uma outra forma para obter o diagrama (a imagem) é com o auxílio da IDE ***Atom*** e sua extensão ***PlantUML Viewer***. Com ambos devidamente instalados, deve-se apenas abrir o arquivo .plantuml e seguir o caminho "*Packages > PlantUML Viewer > Toggle View*", neste ponto uma nova aba abrirá com o diagrama, e este sendo atualizado em "tempo real" ao ter seu código modificado.