Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Relatório de Projeto

GroLab Mobile Data Viewer 2024

Visualizador de Dados

Gonçalo Miguel Oliveira Mões de Paiva Alves

ESTGV: Prof. Doutor Francisco Ferreira Francisco

Open Grow: Eng. Mestre Daniel Carvalho

Viseu, 2024

Instituto Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Departamento de Informática

Relatório de Projeto

Curso de Licenciatura em

Engenharia Informática

GroLab Mobile Data Viewer 2024

Visualizador de Dados

Ano Letivo 2023/2024

Gonçalo Miguel Oliveira Mões de Paiva Alves

ESTGV: Prof. Doutor Francisco Ferreira Francisco

Open Grow: Eng. Mestre Daniel Carvalho

Viseu, 2024

Índice

Índice de tabelas

Índice de figuras

1. Introdução

O presente relatório foi produzido no decorrer da unidade curricular de Projeto da Licenciatura em Engenharia Informática na Escola Superior de Tecnologia e Gestão de Viseu, pertencente ao Instituto Politécnico de Viseu. O estágio foi realizado na *Open Grow*, durante o período compreendido entre 20/02/2024 e 01/07/2024. A proposta de projeto apresentada pela entidade empregadora teve como objetivo o desenvolvimento de uma aplicação móvel para a obtenção e visualização de dados do *GroLab*. A aplicação foi elaborada com funcionalidades que permitem aos utilizadores consultar diversos dados relativos aos seus dispositivos *GroLab* e visualizar dados posteriores ou em tempo real dos respetivos sensores. O estágio permite gerar uma experiência de trabalho valiosa, assim favorecendo o desenvolvimento das minhas capacidades e de futuras integrações em empresas.

1.1. Enquadramento

O projeto de estágio enquadra-se na área de TI (Tecnologias de Informação), mais precisamente no desenvolvimento de uma aplicação mobile. Para a realização da tarefa foi utilizado o *Flutter*, um *devkit* de interfaces de utilizador (*toolkit* e *framework*), que tem como objetivo permitir ao desenvolvedor a possibilidade de criar apenas uma *codebase*, que pode ser compilada para diferentes plataformas, como *Android* e *iOS*, simplificando o processo de desenvolvimento e manutenção.

1.2. Entidade Acolhedora

A *Open Grow* foi fundada em Viseu no ano 2014 e dedica-se à investigação e criação de soluções de automação para os ambientes de cultivo agrícola. O seu principal objetivo é fornecer tecnologia inovadora e versátil com uma interface extremamente fácil de utilizar, permitindo que qualquer agricultor a possa utilizar. Comprometendo-se a desenvolver sistemas de alta qualidade a um preço acessível. Durante o período do estágio, a empresa proporcionou um ambiente agradável e amigável, no qual todos os colaboradores estavam sempre prontos para ajudar com qualquer dúvida ou problema que surgisse.

1.3. Descrição do Problema

A *GroLab* dispõe de uma aplicação de software (*Windows*) para configuração e visualização de dados em tempo real. No entanto, este software foi inicialmente desenvolvido em 2014, apenas compatível com *Windows*. Tendo em conta que a maioria das pessoas atualmente troca o computador pelo telemóvel, torna-se essencial oferecer uma ferramenta para telemóvel que possibilite a visualização dos dados do *GroLab*. Desta forma, o intuito deste projeto é a criação de uma *App* capaz de comunicar com o *GroLab*, obter dados e apresentá-los ao utilizador.

1.4. Objetivos

Os objetivos iniciais definidos para o projeto consistiram no desenvolvimento de uma aplicação intuitiva, que permitisse visualizar os dados do *GroLab*. Para tal efeito era necessário a adaptação/implementação da *stack* de comunicações para receber dados do *GroLab*, Autenticação de utilizadores, Configurações de acesso ao *GroLab* e *dashboard* para visualização de dados (gráficos).

1.5. Calendarização

Durante o estágio, foram vivenciados diversos momentos de aprendizagem, conforme ilustrado na Figura 1, que apresenta a linha do tempo do estágio. Esses momentos incluíram períodos de recolha de informação, com a finalidade da escolha que tecnologias e abordagens a utilizar, compreensão da forma de comunicação com o *GroLab*, e de seguida a realização do estágio em si. Assim sendo segue-se a identificação das etapas e as respetivas datas ao longo do período de estágio.

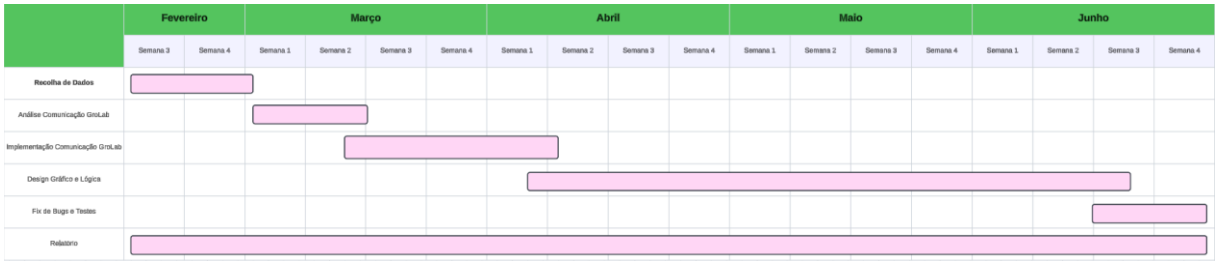

Figura 1 – Calendarização

1.6. Estrutura do relatório

O presente relatório está da seguinte forma:

1. Introdução
2. Estado da Arte
3. Conceção do Projeto
4. Desenvolvimento do Projeto
5. Conclusão
6. Referências

Como se acabou de ler, a introdução abrange o enquadramento do projeto, descreve a entidade acolhedora, descreve o problema que se pretende resolver, os objetivos para o projeto e a calendarização do projeto. De seguida, no capítulo Estado da Arte são referidas as tecnologias que foram utilizadas durante o desenvolvimento do projeto. O terceiro capítulo aborda a conceção do projeto, é descrito a forma como o projeto foi planeado e o necessário para tal. Posteriormente, no capítulo desenvolvimento do projeto, é ilustrado e descrito o funcionamento da aplicação, os ecrãs da mesma e outros aspetos relevantes. Na Conclusão é descrito a reflexão do projeto e do estágio, descrevendo os seus objetivos e algumas das dificuldades superadas. Por fim, temos as referências, onde são citadas todas as fontes consultadas para obter informação, de maneira a possibilitar a escrita do presente relatório.

2. Estado de Arte

Neste capítulo vai ser abordado o estado de arte, mais especificamente as metodologias, tecnologias e ferramentas utilizadas para o desenvolvimento do projeto.

2.1. Tecnologias Utilizadas

Nesta secção é explicado e abordado as diferentes tecnologias utilizadas ao longo da realização do projeto.

Inicialmente foi proposto a utilização de *Xamarin* para o desenvolvimento deste projeto, no entanto, após diversas pesquisas, foi identificado um grande problema. devido ao facto do *Xamarin* encontrar-se em fim de vida, tendo o término do seu suporte a 1 de maio de 2024 (Microsoft, 2024). O sucessor tem de nome *.NET MAUI*, porém ainda se encontra em um estado muito prematuro de desenvolvimento, apresentado falhas e diversos bugs. Após uma análise meticulosa, o *Flutter* destacou-se em relação a outras *frameworks* devido às suas capacidades, afirmação no mercado e por já ter sido utilizado em experiências pessoais anteriores.

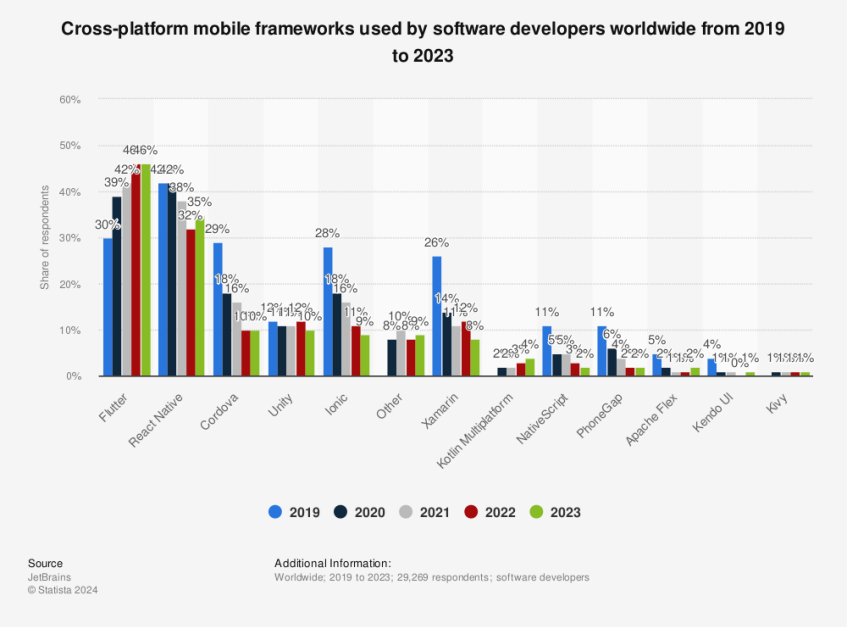

Figura 2 - Comparação Percentagem de Mercado *Frameworks* (2019-2023) (Statista, 2023)

2.1.1. Flutter

Flutter é uma *framework* de código aberto desenvolvido pelo *Google*, utilizado para criar aplicações nativas para *Android*, *iOS*, web e desktop a partir de um único código base. Lançado pela primeira vez em maio de 2017, o *Flutter* tem ganhado popularidade devido à sua capacidade de proporcionar uma experiência de desenvolvimento rápida e eficiente, além de permitir a criação de interfaces de utilizador visualmente atraentes e de alto desempenho.

As principais características do Flutter são:

- ***Hot Reload***: Uma das características mais aclamadas do *Flutter* é o "*Hot Reload*", que permite aos desenvolvedores verem as mudanças no código imediatamente refletidas na aplicação em execução. Isso acelera o processo de desenvolvimento e torna a experimentação e a depuração mais eficazes.
- **UI Nativa e Desempenho Elevado**: Flutter utiliza o motor de *renderização* Skia, o que lhe permite desenhar diretamente na tela e, assim, oferecer uma experiência nativa com desempenho superior. As interfaces de utilizador criadas com *Flutter* são responsivas e podem ser adaptadas para diferentes tamanhos e densidades de tela.
- ***Widget-Based Architecture*:** O *Flutter* é baseado em *widgets*. Tudo na interface do utilizador, desde os controles de layout até aos componentes interativos, é um *widget*. Esta abordagem modular facilita a composição e reutilização de componentes.
- ***Dart Programming Language***: *Flutter* utiliza a linguagem de programação *Dart*, também desenvolvida pelo Google. *Dart* é uma linguagem de propósito geral, fácil de aprender para desenvolvedores vindos de outras linguagens orientadas a objetos como *Java* ou *JavaScript*.
- **Desenvolvimento Multiplataforma:** Com *Flutter*, é possível desenvolver para várias plataformas a partir de um único código base. Isto significa que uma aplicação pode ser lançada simultaneamente para *Android*, *iOS*, *web* e *desktop*, reduzindo o tempo e o esforço necessários para manter versões separadas da mesma aplicação (Flutter, 2024).

Tabela 1 - Comparação entre diferentes framework de desenvolvimento multi-plataforma (Kuppan, 2023)

Característica	Flutter	React Native	Xamarin	.NET MAUI
Linguagem de Programação	Dart	JavaScript	C#	C#
Desenvolvido por	Google	Facebook (Meta)	Microsoft	Microsoft
Plataformas Suportadas	iOS, Android, Web, Desktop (Windows, macOS, Linux)	iOS, Android, Web	iOS, Android, Windows, macOS	iOS, Android, Windows, macOS
Acesso a APIs Nativas	Fácil através de plugins e canais de plataforma	Fácil através de módulos nativos	Completo, acesso direto ao SDK	Completo, acesso direto ao SDK
Desempenho	Alto, próximo ao nativo	Bom, próximo ao nativo	Bom, próximo ao nativo	Bom, próximo ao nativo
Interface do Utilizador	Personalizável com widgets do Flutter	Componentes nativos	Interface nativa através de XAML	Interface nativa através de XAML
Ferramentas de Desenvolvimento	Flutter SDK, Android Studio, VS Code	React Native CLI, Expo, VS Code	Visual Studio, Xamarin Studio	Visual Studio
Comunidade	Grande e ativa	Grande e ativa	Média, em declínio	Pequena, mas crescente
Hot Reload	Sim	Sim	Sim	Sim
Gerenciamento de Estado	Provider, Riverpod, Bloc	Redux, MobX	Xamarin.Forms State Management	.NET MAUI State Management
Atualizações	Rápidas e frequentes	Rápidas e frequentes	Fim de Vida	Moderadas

2.1.2. Dart

Dart é uma linguagem de programação desenvolvida pelo Google, lançada pela primeira vez em 2011. Concebida inicialmente para o desenvolvimento web, a linguagem evoluiu e tornou-se um pilar fundamental para o desenvolvimento de aplicações utilizando o *framework* Flutter. *Dart* é uma linguagem orientada a objetos e de propósito geral, que combina a familiaridade de linguagens como *Java* e *JavaScript* com recursos modernos e eficientes.

Sendo os seus destaques:

- **Simplicidade e Familiaridade:** *Dart* foi projetada para ser fácil de aprender e usar. A sua sintaxe é clara e concisa, semelhante a outras linguagens populares como *Java* e *C#*, o que facilita a adoção por desenvolvedores com experiência em linguagens orientadas a objetos.
- **Performance Elevada**: *Dart* é compilada diretamente para código nativo, o que permite um desempenho extremamente rápido em ambientes móveis e desktop. Além disso, para aplicações web, *Dart* pode ser transpilada para *JavaScript*, garantindo compatibilidade e desempenho otimizados.
- **Sistema de Tipos Estático e Dinâmico**: *Dart* oferece um sistema de tipos flexível que suporta tanto tipagem estática quanto dinâmica. Os desenvolvedores podem optar por declarar tipos explicitamente ou usar a inferência de tipos, combinando a segurança da tipagem estática com a flexibilidade da tipagem dinâmica.
- **Assincronismo e Concorrência**: A linguagem inclui suporte nativo para programação assíncrona, com o uso de *async* e *await*, facilitando o desenvolvimento de aplicações que realizam operações de I/O ou outras tarefas assíncronas de maneira eficiente e legível.
- **Bibliotecas e Pacotes**: *Dart* possui um ecossistema robusto de bibliotecas e pacotes disponíveis através do *Pub*, o repositório oficial de pacotes da linguagem. Isso permite a reutilização de código e a integração de funcionalidades adicionais de maneira fácil e eficiente (Bracha, 2015).

2.1.3. SQLite

SQLite é um sistema de gestão de base de dados relacional (SGBDR) leve e de código aberto. Diferente de outros sistemas de base de dados tradicionais, como *MySQL* ou *PostgreSQL*, que seguem o modelo cliente-servidor, o *SQLite* é embutido diretamente nas aplicações, armazenando os dados em arquivos simples no sistema de ficheiros do dispositivo. Esta arquitetura torna o *SQLite* ideal para aplicações móveis, pequenas aplicações desktop e outros ambientes com recursos limitados.

Os pontos mais importantes sendo:

- **Zero Configuração**: *SQLite* é um sistema "*zero-conf*", o que significa que não requer instalação ou configuração. As aplicações que utilizam *SQLite* podem simplesmente incluir a biblioteca *SQLite* e começar a usar a base de dados sem configurações adicionais.
- **Base de Dados Embutida**: Diferente dos SGBDR tradicionais, o *SQLite* é embutido nas aplicações. Isso significa que a base de dados reside no mesmo ambiente da aplicação, eliminando a necessidade de um servidor de base de dados separado.
- **Armazenamento em Arquivo Único**: Os dados em *SQLite* são armazenados em um único ficheiro de base de dados, facilitando a gestão, backup e transferência dos dados.
- **Tamanho Pequeno**: *SQLite* é extremamente leve, com um binário que normalmente tem menos de 500 KB de tamanho. Isso torna o *SQLite* uma excelente escolha para dispositivos com recursos limitados.
- **Conformidade com SQL**: *SQLite* suporta a maioria dos comandos SQL padrão, incluindo transações, consultas complexas e integrações com outras ferramentas de *SQL*. Isso facilita a migração de aplicações de outros SGBDR para *SQLite*.
- **Multi-Plataforma:** *SQLite* é portátil e funciona em diversas plataformas, incluindo *Windows*, *macOS*, *Linux*, *iOS* e *Android* (Kreibich, 2010).

2.1.4. Grolab

O *GroLab* é um sistema avançado de controle de cultivo que utiliza tecnologia de ponta para monitorizar e automatizar todos os aspectos do cultivo de plantas, desde a iluminação e clima, até à irrigação e dosagem de nutrientes. Ele é projetado para otimizar o crescimento das plantas e maximizar a eficiência do cultivo, libertando o cultivador de tarefas manuais e repetitivas.

O ***GroNode*** é a unidade central de controlo que gere todos os outros módulos do sistema, sendo estes o ***PowerBot***, ***SoilBot***, ***TankBot*** e o ***UserBot***, enviando instruções e recebendo informações sobre os diferentes elementos do cultivo. Este é o ponto de entrada e saída de dados para o exterior. Estes módulos controlam diferentes aspetos do cultivo, como iluminação, clima, irrigação, pH e dosagem de nutrientes, através de sensores que monitorizam constantemente as condições do cultivo, como temperatura, humidade, pH e nível de nutrientes.

Pontos de destaque do *GroLab*:

- **Otimização do crescimento das plantas:** O *GroLab* garante que as plantas recebam as condições ideais de luz, temperatura, humidade e nutrientes em todos os estados de crescimento.
- **Maximização da eficiência:** O *GroLab* automatiza tarefas repetitivas e otimiza o uso de recursos, como água e energia, resultando em economia de custos e maior produtividade.
- **Monitoramento e controle remotos:** O *GroLab* permite que o utilizador monitore e controle o cultivo de qualquer lugar, a qualquer hora, através do seu computador.
- **Análise de dados:** O *GroLab* coleta e armazena dados sobre o cultivo, permitindo ao usuário analisar o desempenho das plantas e tomar decisões informadas para melhorar o cultivo.

Existem diferentes versões do *GroLab* para atender às necessidades de diferentes tipos de cultivadores (OpenGrow, 2024).

Figura 3 - Diversos Módulos *GroLab*

2.1.5. Modbus

O *Modbus* é um protocolo de comunicação desenvolvido pela *Modicon* (atualmente *Schneider Electric*) em 1979, inicialmente para ser utilizado com controladores lógicos programáveis (*P.L.C.*). Tornou-se um padrão de facto na automação industrial, permitindo a comunicação entre diversos dispositivos ligados à mesma rede. Aqui estão alguns pontos chave sobre o *Modbus*:

- **Simplicidade**: O *Modbus* é simples e fácil de implementar, o que contribuiu para a sua ampla adoção.
- **Flexibilidade**: Pode ser usado em vários tipos de redes e meios de comunicação, incluindo RS-232, RS-485, Ethernet e TCP/IP.
- **Interoperabilidade**: Permite a comunicação entre dispositivos de diferentes fabricantes, desde que todos sigam o protocolo *Modbus*.

Este tem também vários modos de operação:

- ***Modbus* RTU (Remote Terminal Unit)**: Utiliza comunicação série (RS-232 ou RS-485) e é conhecido pela sua eficiência, pois transmite dados em formato binário.

- ***Modbus* ASCII**: Também usa comunicação série, mas transmite dados em formato ASCII, o que facilita a leitura e depuração dos dados transmitidos.
- ***Modbus* TCP/IP**: Funciona sobre redes Ethernet e usa o protocolo TCP/IP para a comunicação. É ideal para redes locais e sistemas modernos de automação.

A estrutura de uma mensagem *Modbus* é composta por:

- **Endereço do Dispositivo**: Identifica o dispositivo escravo (slave) na rede.
- **Código de Função**: Define a ação a ser executada (por exemplo, leitura ou escrita de dados).
- **Dados**: Contém a informação a ser lida ou escrita, conforme especificado pelo código de função.
- ***CRC* (*Cyclic Redundancy Check*)**: Utilizado para verificação de erros na comunicação («Modbus», 2024).

Importante salientar, que toda a comunicação efetuada entre o *GroNode* e o software de visualização de dados é realizada através de uma versão modificada deste protocolo, tomando base o *Modbus* TCP/IP, seguindo essencialmente o mesmo padrão de funcionamento, porém adaptado para as necessidades do *GroLab*. Por motivos de segurança e confiabilidade, a pedido da EA, não será explicado de forma concreta a forma de funcionamento deste (valores por exemplo de bytes de autenticação, etc), pois seria uma exposição severa da lógica de

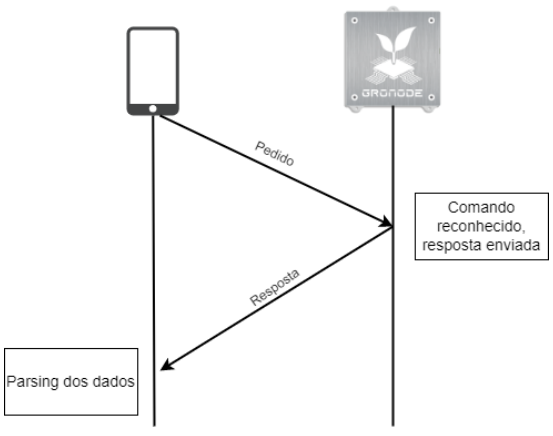

Figura 4 - Pedido Genérico

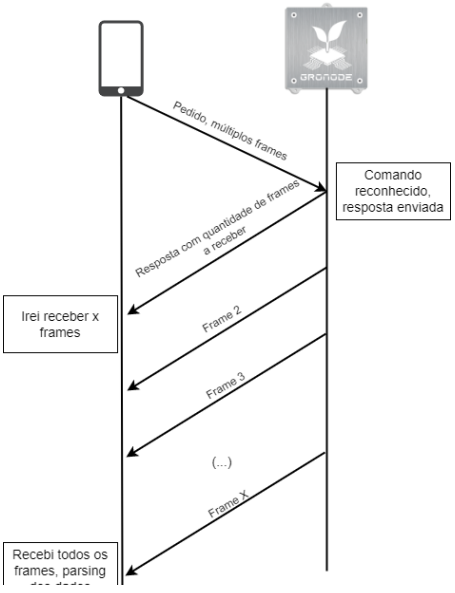

Figura 5 - Pedido Genérico com várias respostas

3. Conceção do Projeto

Neste capítulo vai ser abordado a conceção do projeto.

O projeto completo apresentado pela entidade acolhedora está dividido em duas partes:

- ***Frontend***: Interfaces gráficas, onde o utilizador facilmente consegue interagir com os dados recebidos enviados pelo *GroLab*. Como gráficos, *dashboard*, entre outros.
- ***Backend***: Responsável pela comunicação com *GroLab*, desde autenticação, envio e receção de dados entre este e o dispositivo móvel.

3.1. Visualização de Dados

O objetivo desta aplicação é permitir a um utilizador autenticar-se, e conseguir analisar de forma conjunta todos os dados obtidos pelos vários dispositivos *GroLab*. Para tal necessita de existir um meio de comunicação entre o dispositivo móvel e o *GroLab*, tal como uma interface intuitiva e simples, que permite facilmente a consulta destes dados, e a possibilidade de filtragem caso seja necessária. Nestes dados incluem não só valores de sensores presentes, mas também dados sobre os próprios dispositivos *GroLab*, áreas, cultivos, horários e alarmes.

3.2. Modelação

A secção seguinte vai abordar e explicar a modelação de dados realizada.

3.2.1. Modelação de requisitos funcionais - Casos de uso

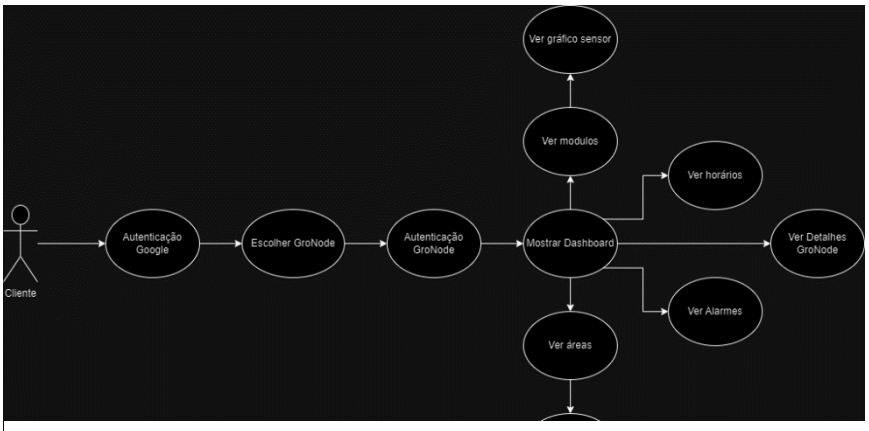

Figura 6 - Casos de Uso

3.2.2. Modelação de dados

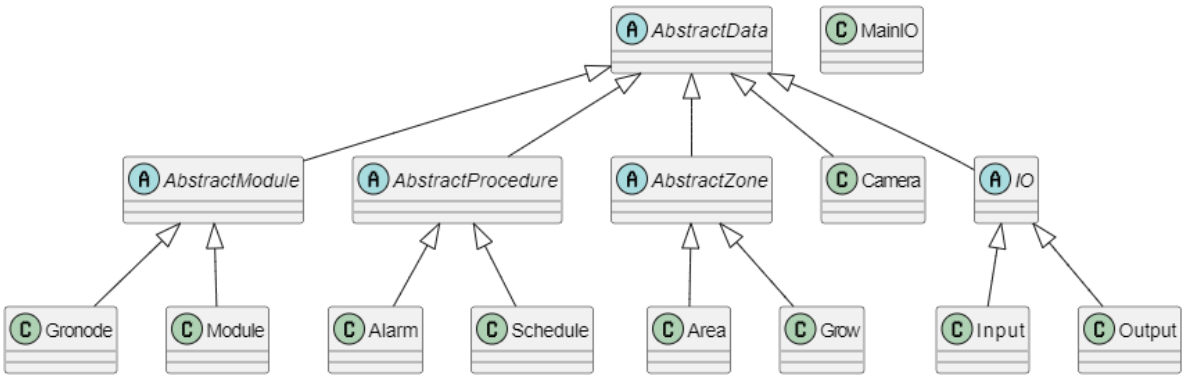

Figura 7 - Diagrama de Classes

O diagrama de classes ilustra a hierarquia e as relações entre várias classes dentro do sistema. No topo da hierarquia está `**AbstractData**`, uma classe base abstrata da qual todas as outras classes derivam. Esta classe serve como a fundação para toda a estrutura de dados do sistema, contém campos genéricos como o ID e o Nome.

Derivando de `**AbstractData**`, temos quatro subclasses abstratas: `**AbstractModule**`, `**AbstractProcedure**`, `**AbstractZone**` e `**IO**`. Cada uma destas subclasses abstratas especializa a funcionalidade da classe base. `**AbstractModule**` serve como a base para todos os módulos, enquanto `**AbstractProcedure**` é a base para os procedimentos. `**AbstractZone**` é usada para definir diferentes zonas, e `**IO**` abrange as entradas e saídas.

As classes concretas derivadas destas subclasses abstratas implementam comportamentos específicos. `**Alarm**` e `**Schedule**`, por exemplo, derivam de `**AbstractProcedure**`, representando um alarme e um horário, respectivamente. `**Area**` e `**Grow**` derivam de `**AbstractZone**`, representando as áreas e os cultivos. `**Gronode**` e `**Module**` são os diferentes módulos e o GroNode, derivados de `**AbstractModule**`. `**Input**` e `**Output**`, por outro lado, derivam de `**IO**`, representando os sensores de entradas e saídas de dados, respectivamente. `**Camera**` é uma classe concreta derivada diretamente de `**AbstractData**`, representando as câmaras.

3.2.3. Base de Dados

Na figura 6, está representado o diagrama da base de dados, esta base de dados é exatamente a mesma utilizada no atual software de gestão para *Windows*.

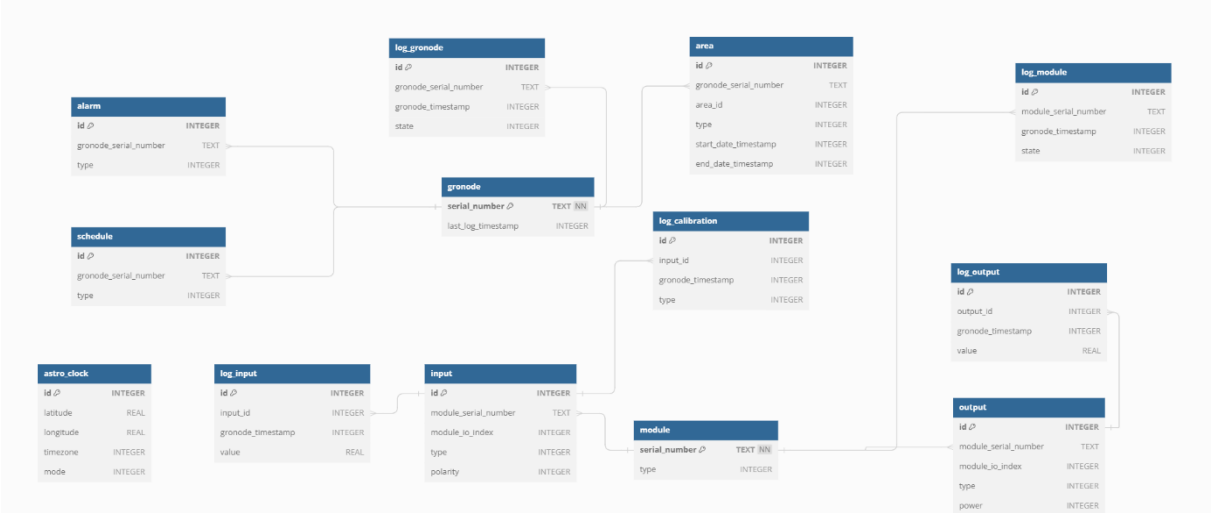

Figura 8 - Diagrama Base de Dados

A função desta é para armazenar os dados recolhidos pelo *GroNode*, para posterior amostragem. Por exemplo durante 1 semana, este recolhe dados. No final da semana o utilizador pode sincronizar a sua base de dados local com a do *GroNode*, permitindo visualizar os valores obtidos no decorrer dessa semana, não existindo a necessidade de estar ligado continuamente ao dispositivo.

4. Desenvolvimento do Projeto

Neste capítulo vai ser abordado o funcionamento da aplicação e os diversos processos necessários para o bom funcionamento.

4.1. Funcionamento da Aplicação

A aplicação dispõe de vários ecrãs para a interação do utilizador com o sistema. Inicialmente é realizada a autenticação na aplicação, local ou através da conta google do utilizador, a utilização desta em relação à autenticação local é vantajosa pois permite a sincronização da base de dados local em qualquer dispositivo móvel que utilize. De seguida, o utilizador pode ligar-se diretamente através de IP ou *hostname* ao *GroNode*, ou caso não tenha acesso a estes dados pode procurar na sua rede por todos os *GroNodes* disponíveis. Após esta ligação, uma ligação *socket* entre os dois é estabelecida, permitindo a comunicação exclusiva entre o visualizador e o *GroNode*. Agora ocorre ao visualizador de dados executar pedidos de dados ao *GroNode*, que estará sempre à escuta por estes. Todos os dados recebidos são em byte sendo necessário um *parser* de dados específico para cada pedido.

```
case ModuleType.tankBot:
  // PH
  var TBPh = InputDataLogDB(
      index: 0, type: InputType.PH_PROBE, polarity: 0x00, value: 0.0);
  TBPh.value = ByteData.sublistView(message)
      .getFloat32(parseDataIndex, Endian.little);
  parseDataIndex += 4;
  TBPh.type = getInputTypeFromValue(message[parseDataIndex++]);
  if (TBPh.type != InputType.INPUT_NOT_CONNECTED &&
      TBPh.type != InputType.INPUT_NOT_EXISTS) {
    newModuleLog.inputs.add(TBPh);
  }
```

Figura 9 - Exemplo de *parsing* de um sensor e seu respetivo valor para adição na base de dados local

Após a sincronização dos dados relativos às áreas, alarmes, horários, cultivos, módulos e caso existam registo dos sensores, é mostrado ao utilizador um *dashboard* com um resumo de dados relevantes do seu *GroLab*. A partir daqui o utilizador pode abrir diversos ecrãs, enquanto por trás os *sockets* comunicam em uma *thread* isolada para manterem a ligação e a sincronia dos dados mais recentes.

No ecrã Módulos, o utilizador pode consultar os diversos módulos atualmente ligados ou anteriormente ligados ao *GroNode*. Em cada um pode consultar dados relativos a este, sendo um dos dados os sensores que este encontra-se ligado. Caso pretenda o utilizador pode visualizar os dados obtidos de casa sensor em tempo real através de um gráfico ou definindo um intervalo de tempo, caso tenha registos de valores passados. No ecrã Áreas podemos consultar dados relativos a estas, tal como os seus cultivos caso existam. Novamente um cultivo pode ter sensores de entrada de saída de dados associados, e nestes podemos ver através de uma demonstração gráfica os valores lidos. No ecrã Horários encontram-se os horários gravados no *GroNode*, estes estão associados a uma saída, e têm como objetivo definir atividades recorrentes. Podem ser aplicadas a uma área e/ou a um cultivo ou até mesmo não serem aplicados a nenhum destes. No ecrã Alarmes são mostrados os diversos alarmes existentes, tal como os seus detalhes: valores limites, decisões, consequências de certos valores lidos, estado de sensores, etc. No ecrã Informações do meu *Gronode*, são mostrados alguns mais dados avançados sobre o dispositivo de cultivo. Como número de série, versão do *firmware*, entre outros. Por últimos um ecrã com algumas opções úteis como a alteração da unidade de temperatura, alteração do idioma, *reset* dos dados da aplicação, entre outras.

4.1.1. Abertura da Aplicação

Quando o utilizador abre a aplicação são inicialmente verificados alguns dados relativos ao seu dispositivo móvel.

Figura 10 - *Splash Screen*

Qual o idioma utilizado e se é a primeira vez que executa a app neste dispositivo, enquanto estas verificações executam é mostrado um simples *splash screen* com o logo da aplicação.

Figura 11 - Autenticação Local

Caso o utilizador já tenha utilizado a aplicação, os dados de login serão preenchidos automaticamente, e se se autenticou com a sua conta *Google* o ecrã de login é totalmente ultrapassado e autenticação na aplicação é feita de forma automática. Se nunca utilizou pode autenticar-se com as credenciais de predefinição do *GroLab* ou através da sua conta *Google*.

4.1.2. Autenticação no *GroNode*

Neste ecrã o utilizador pode escolher entre três opções para autenticar-se no *GroNode*, caso conheça o seu número de série ou endereço IP ou *hostname*, caso encontra-se em uma rede exterior ou se não conhecer nenhum destes dados é possível realizar *ping sweep* [1] na porta utilizada pelo *GroNode* em todos em endereços da rede. Caso o *GroNode* esteja presente irá retornar uma resposta positiva e será identificado como um possível *GroNode* para estabelecer ligação.

Figura 5 - Ligação por número de série

Figura 4 - Ligação por IP Direto

Figura 14 – Localizar *GroNodes* na Rede

De seguida é aberto um *socket* e é enviado o comando de autenticação ao *GroNode*, contendo os dados de autenticação do *GroNode*. Em caso de autenticação com sucesso é retornado alguns dados básicos do *GroNode*, como a versão, nome, versão do hardware, entre outros.

[1] *Ping sweep (ICMP sweep):*
Método para descobrir hosts ativos em uma rede, enviando solicitações *ICMP Echo* (ping) para vários endereços IP e verificando quais respondem.

4.1.3. Sincronização de Dados

Primeiramente é verificado se existe alguma base de dados local ou na *cloud* do utilizador. Caso exista na *cloud* esta é sincronizada automaticamente para o dispositivo do utilizador, caso não exista na *cloud* esta é criada localmente e na próxima autenticação é carregada para a *cloud*. Por último, caso já existam quer na *cloud*, quer localmente o utilizador é questionado quais os dados que deseja manter, os locais ou os da *cloud* ou se não pretender fazer nada. Se escolher a opção local os dados presentes na *cloud* serão sobrepostos com estes e vive-versa no caso de escolher a *cloud*. De seguida, são sincronizados os dados presentes no *GroNode*, como módulos, áreas, horários e alarmes. Por último, se estiverem presentes na memória do *GroNode*, o utilizador é questionado se pretender obter os registos catalogados. Se o assim pretender, estes dados são adicionados à base de dados local, para consulta futura.

Figura 15 – Alerta base de dados local ou na nuvem

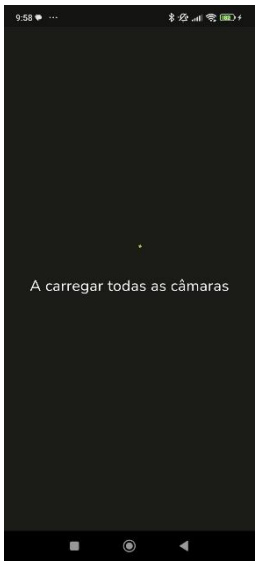

Figura 16 – Menu de *Loading*

Figura 17 – Alerta sincronização dados do *GroNode*

Após executados todos estes comandos, uma *thread* isolada será criada, onde uma função irá pedir a atualização dos dados, a cada 10 segundos, ao *GroNode*. Com esta função assíncrona é possível manter o *socket* de ligação aberto, e receber constantes atualizações de dados, por exemplo, a ativação de um horário ou um disparo de um alarme.

4.1.4. *Dashboard*

Este é o ponto de partida da aplicação, neste ecrã temos alguns dados imediatos. Como os totais de números de módulos, áreas, horários e alarmes. É mostrado também, o nome do *GroNode* ligado, tal como o relógio interno do dispositivo pois este pode funcionar com um fuso horário diferente da região do utilizador. Caso existam horários ativos, serão mostrados três, sendo o fator de diferenciação o tempo restante para o seu termino, por fim são mostradas as diferentes áreas de cultivo, nas quais o utilizador pode aceder para consultar com mais detalhe.

Figura 18 - *Dashboard*

Figura 19 - Dashboard em outro dispositivo móvel (modo claro)

4.1.5. Side Drawer

Este *widget* é o responsável por todas as ligações da aplicação, através dele o utilizador pode navegar entre os diversos ecrãs.

Figura 20 - Side Drawer

4.1.6. Módulos

Neste ecrã o utilizador pode consultar todos os módulos atualmente ligados ou que já se tenha ligado posteriormente e atualmente encontram-se em estado *offline*. Está subdivido em 4 opções:

- *PowerBot*
- *SoilBot*
- *TankBot*
- Outros

Cada opção mostra uma imagem representativa do módulo, nome e o seu estado caso encontre-se *offline*. Importante salientar, que a imagem altera consoante a versão do módulo, por exemplo um *TankBot* por ser um *TankBot* normal ou um *TankBot Plus*, como podemos ver na imagem em baixo.

Figura 21 - Lista de Módulos

O separador Outros é dedicado caso o utilizador pretender conectar os seus próprios módulos ou ligar mais módulos listados acima, pois o *GroNode* tem a capacidade de guardar 4 módulos de cada categoria.

4.1.6.1 Consultar módulo

Após o utilizador escolher o módulo que pretender obter mais informações, é mostrado o módulo selecionado tal como os respetivos sensores de entrada e saídas ligados a este, em formato de lista. Neste momento a função assíncrona criada na autenticação é alterada para obter os valores dos sensores do módulo a cada 3 segundos, permitindo mostrar de forma conjunta a atualização quase imediata de todos os valores dos sensores em simultâneo. Se pretender ainda ver com mais detalhe um sensor, o utilizador pode escolher um deles.

Figura 262 - Entradas de um Módulo

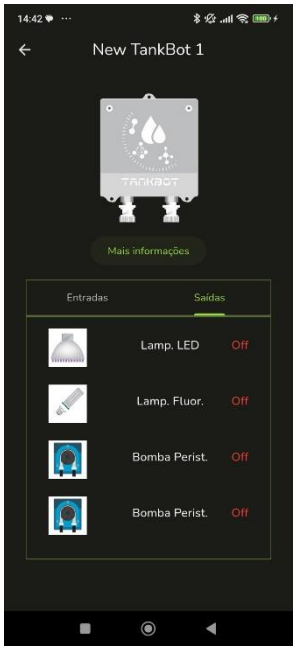

Figura 23 - Saídas de um Módulo

4.1.6.2 Visualização gráfica de valores de um sensor

Neste ecrã é apresentado ao utilizador um gráfico, onde serão mostrados os valores do sensor escolhido, com diversas opções de intervalos de tempo. Consoante a decisão do utilizador os dados serão mostrados na ordem desejada. Na escolha em "Tempo Real", os valores do sensor serão obtidos e mostrados em tempo real, desde o momento da última recolha de dados. Se definir por exemplo "6 Horas", caso existam na base de dados local, serão

mostrados os valores das últimas 6 horas e futuros valores obtidos em tempo real. Por último, o utilizador pode ainda definir um período de tempo.

Figura 247 - Gráfico em tempo real

Figura 25 - Gráfico período definido

4.1.7. Áreas

No separador das áreas o utilizador pode consultar todos as áreas registadas e os seus respetivos cultivos. É apresentado um *widget* por cada área existente. Caso a área contenha cultivos é mostrada uma imagem apelativa ao tipo de cultivo, tal como a data de criação e o nome definido, se não existirem cultivos a área estará vazia. Se pretender o utilizador pode consultar cada área com mais detalhe tocando na pretendida.

Figura 26 - Lista das Áreas

4.1.7.1 Consultar Área e Cultivo

Após o utilizador escolher a área que pretender obter mais informações, é mostrado o os respetivos cultivos desta, tal como os respetivos sensores de entrada e saídas associados ao cultivo, em formato de lista. Pode existir apenas um cultivo ou nem existir, ou até mesmo um cultivo não ter qualquer sensor associado a este.

Figura 27 - Exemplo de um cultivo vazio

Figura 28 - Exemplo de um cultivo com entradas

4.1.7.2 Visualização gráfica de valores de um sensor (Área)

Usufruindo das capacidades do Flutter e da estrutura de dados utilizada, é possível a reutilização do ecrã de visualização gráfica de valores de um sensor, dos módulos, permitindo a fácil integração desse ecrã. (Ver Ponto **4.1.5.2** "Visualização gráfica de valores de um sensor" para mais informações)

4.1.8. Horários

Neste ecrã é apresentado ao utilizador os seus horários registados no *GroNode*. Nestes consegue visualizar o tipo de horário, o atuador que irá ativar quando a hora de início definida é atingida. Caso o horário não esteja atualmente ativo, será mostrada uma barra vermelha com o tempo total de funcionamento do horário, se estiver ativo a barra irá mostrar a percentagem decorrida do horário, tal como o tempo decorrido e tempo total. Também se for aplicado são mostradas as respetivas áreas e/ou cultivos associados ao horário.

Figura 29 - Lista de horários

4.1.9. Alarmes

Nesta interface podemos consultar os alarmes definidos pelo utilizador ou pelo próprio *GroNode*. Um alarme pode estar associado a um horário, por exemplo quando começa um horário de iluminação pode existir um alarme que irá alterar o estado de uma lâmpada *LED* para **ON**.

Um alarme tem um formato semelhante a uma condição SE/ENTÃO, no mundo da programação, podendo ser representado dessa forma para ajudar a compreensão. No exemplo abaixo temos por exemplo:

"Se valor do sensor de temperatura da água é superior a 150 ºC então é ligado a lâmpada fluorescente até o valor do sensor de temperatura da água ser inferior a 150 ºC"

Figura 30 - Lista de alarmes

Figura 31 - Exemplo de outro tipo de alarmes

4.1.10. Câmaras

No ecrã das câmaras é apresentado ao utilizador as câmaras guardadas no *GroNode*. Porém caso o utilizador não tenha quais quer câmaras guardadas, pode conectar-se na mesma a qualquer câmara caso tenho o seu *username, password*, endereço e protocolo. Através do pacote VLC existe a capacidade de suporte de *livefeed* de dados, como *streams*, vídeos em direto, ou nesta situação uma transmissão em direto CCTV, permitindo a existência desta funcionalidade.

Figura 32 - Lista de câmaras guardadas

Figura 383 - Ligação direta a uma câmara

Figura 34 - *LiveFeed* de uma câmara

4.1.11. Informações do meu *GroNode*

Este pequeno ecrã apenas contém dados relevantes do *GroNode* como o seu nome, endereço *IP*, endereço *MAC*, porta, *Hostname*, Versão *Hardware*, *Serial Number* e Data de Produção. Tem como função ser um ecrã mera mente informativo sobre as características do dispositivo.

Figura 35 - Informações GroNode

4.1.12. Definições

Neste menu temos algumas opções que oferecem algumas alterações necessárias caso o utilizador as pretenda, tal como algumas informações.

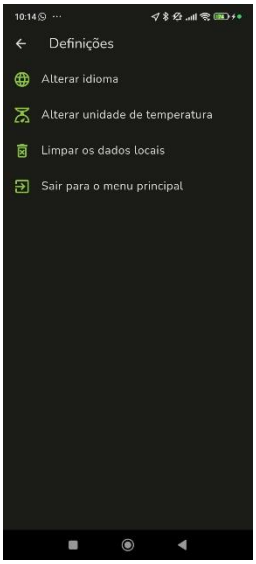

Figura 36 - Menu de definições

4.1.12.1 Alterar Idioma

Neste menu é possível a alteração do idioma da aplicação. Após a escolha de umas das possíveis opções, o idioma é alterado de imediato.

Figura 37 - Menu alteração idioma

Figura 98 - Informações *GroNode* em espanhol

4.1.12.2 Alterar unidade de temperatura

No menu alterar unidade de temperatura, o utilizador consegue alterar a unidade de temperatura utilizada em toda a aplicação.

Figura 3910 - Menu alteração de unidade de temperatura

Figura 4011 - Unidade alterada

4.1.12.3 Limpar dados da aplicação

Esta opção permite que o utilizador elimine todos os dados presentes na aplicação. Estes dados são: base de dados local, definições guardadas como unidade de temperatura ou idioma e a chave de autenticação na sua conta Google.

4.1.12.4 Sair para o ecrã principal

Esta opção simplesmente retorna o utilizador de volta ao menu inicial, desconectando-se do *GroNode* atualmente ligado e limpa todos os dados na memória temporária da aplicação.

5. Conclusão

Com a implementação integral das funcionalidades pretendidas, o objetivo principal deste estágio foi alcançado com sucesso.

A experiência foi enriquecedora tanto a nível pessoal como profissional. Adquiri novos conhecimentos em áreas como comunicação, sistemas embebidos e desenvolvimento mobile, graças ao ambiente acolhedor e colaborativo da Open Grow.

Inicialmente, a complexidade do sistema representou um desafio. No entanto, o apoio dos colaboradores foi fundamental para a compreensão do funcionamento da comunicação Modbus e do GroNode. A comunicação constante com o cliente, através de "updates" diários e discussões regulares, garantiu o alinhamento e o progresso contínuo do desenvolvimento da aplicação.

Embora o projeto ainda esteja em desenvolvimento, a base de comunicação estabelecida permite a expansão futura das funcionalidades do GroLab, como a interação do utilizador e o controlo de dispositivos. Atualmente, a aplicação funciona como um visualizador de dados, mas o "stack" de comunicação implementado facilita a integração de novas funcionalidades através da adaptação do "parsing" de dados.

6. Referências

Bracha, G. (2015). *The Dart programming language*. Addison-Wesley Professional. https://books.google.com/books?hl=pt-PT&lr=&id=UHAlCwAAQBAJ&oi=fnd&pg=PT13&dq=The+Dart+Programming+Language+Gilad+Bracha&ots=Pq_Tg_GaKQ&sig=4olQAdIiBI7qs3kKo1DPXB1VUsc

Flutter. (2024). *FAQ*. docs.flutter.dev. https://docs.flutter.dev/resources/faq

Kreibich, J. (2010). *Using SQLite*. O'Reilly Media, Inc.

Kuppan, M. (2023). *.NET MAUI vs. Flutter vs. React Native: A Comprehensive Comparison*. KANINI. https://kanini.com/blog/net-maui-vs-flutter-vs-react-native/

Microsoft. (2024). *Xamarin official support policy | .NET*. Microsoft. https://dotnet.microsoft.com/en-us/platform/support/policy/xamarin

Modbus. (2024). Em *Wikipédia, a enciclopédia livre*. https://pt.wikipedia.org/w/index.php?title=Modbus&oldid=67436932

OpenGrow. (2024). GroLab™ Software. *GroLab™ | Controlador de Cultivos - Automatização Agrícola*. https://opengrow.pt/pt/software-grolab/

Statista. (2023). *Cross-platform mobile frameworks used by global developers 2023*. Statista. https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/